Social Choice for Partial Preferences Using Imputation

by

John A. Doucette

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

Abstract

Within the field of multiagent systems, the area of computational social choice considers the problems arising when decisions must be made collectively by a group of agents. Usually such systems collect a ranking of the alternatives from each member of the group in turn, and aggregate these individual rankings to arrive at a collective decision. However, when there are many alternatives to consider, individual agents may be unwilling, or unable, to rank all of them, leading to decisions that must be made on the basis of incomplete information. While earlier approaches attempt to work with the provided rankings by making assumptions about the nature of the missing information, this can lead to undesirable outcomes when the assumptions do not hold, and is ill-suited to certain problem domains. In this thesis, we propose a new approach that uses machine learning algorithms (both conventional and purpose-built) to generate plausible completions of each agent's rankings on the basis of the partial rankings the agent provided (imputations), in a way that reflects the agents' true preferences. We show that the combination of existing social choice functions with certain classes of imputation algorithms, which forms the core of our proposed solution, is equivalent to a form of social choice. Our system then undergoes an extensive empirical validation under 40 different test conditions, involving more than 50,000 group decision problems generated from real-world electoral data, and is found to outperform existing competitors significantly, leading to better group decisions overall. Detailed empirical findings are also used to characterize the behaviour of the system, and illustrate the circumstances in which it is most advantageous. A general testbed for comparing solutions using real-world and artificial data (Prefmine) is then described, in conjunction with results that justify its design decisions. We move on to propose a new machine learning algorithm intended specifically to learn and impute the preferences of agents, and validate its effectiveness. This Markov-Tree approach is demonstrated to be superior to imputation using conventional machine learning, and has a simple interpretation that characterizes the problems on which it will perform well. Later chapters contain an axiomatic validation of both of our new approaches, as well as techniques for mitigating their manipulability. The thesis concludes with a discussion of the applicability of its contributions, both for multiagent systems and for settings involving human elections. In all, we reveal an interesting connection between machine learning and computational social choice, and introduce a testbed which facilitates future research efforts on computational social choice for partial preferences, by allowing empirical comparisons between competing approaches to be conducted easily, accurately, and quickly. Perhaps most importantly, we offer an important and effective new direction for enabling group decision making when preferences are not completely specified, using imputation methods.

Acknowledgements

I thank my supervisor, Professor Robin Cohen, for her advice, guidance, and knowledge, which were instrumental to this thesis, and invaluable for my academic career besides. Robin's concern for her students is exceptional, and serves as an example for how an adviser's interest can enhance the success of her students.

I thank the members of my committee. Professors Kate Larson and Daniel Lizotte provided valuable feedback and criticism throughout my doctoral studies, as well as essential knowledge and instruction. Professors Paul Thagard and Toby Walsh offered compelling questions and suggestions on the final manuscript that were greatly appreciated. Professor Marc Kilgour also provided deep insights into the thesis.

The efforts of university staff members helped me to navigate the complex university bureaucracy, and remained familiar faces amid an ever changing population of students and faculty. I especially thank Wendy Rush, Margaret Towell, Neoma Power, and Jessica Miranda for their assistance.

Throughout my doctoral studies, visits, outings, and correspondence with friends and family, both near and far, have helped me to remain focused and engaged with my work. Foremost, I thank my spouse, Catherine Holloway, for her support and companionship throughout our academic journey. I also thank my family, especially Sally, Glenn, Zoe and Clara for their support, and my grandfather Tony Erskine for his correspondence and understanding. My close friends Elliot Snow-Kropla, Michael Todd, Jake Summers, and Colin Conrad provided the occasional escape from academia, and the accompanying shift in perspective, that was a great source of relaxation during my doctoral studies.

Closer to home, Alan Tsang, Hadi Hosseini, Cecylia Bocovich, and Tariq Elahi became collaborators as well as friends, broadening the scope of my studies and providing the intellectually stimulating conversations that make the academic environment so compelling in the first place. John Champaign, Lachlan Dufton, Graham Pinhey, Dean Shaft and Shengying Pan kept our shared office a lively and thought provoking place. Noel Sardana, Vijay Menon, Michael Cormier, Dan Ricoskie, Rhiannon Rose, and Adam Hartfiel cultivated a similar atmosphere in the AI Lab at large. Friends and classmates Sarah Kaiser, Erinn Atwater, Yuval Sanders and Steven Casagrande commiserated or celebrated as our shared times in academia demanded.

Finally, I acknowledge the generous financial support of the Vanier Canada Graduate Scholarship program, the Natural Science and Engineering Research Council of Canada, the David R. Cheriton Scholarship program, the Ontario Graduate Scholarship program, the GO-Bell Scholarship program, and the University of Waterloo.

Dedication

I dedicate this thesis to my wife and partner in life, Catherine Holloway. Without her love, support and friendship my doctoral studies would not have been possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

I consider it completely unimportant who in the party will vote, or how; but what is extraordinarily important is this—–who will count the votes, and how.

Boris Bazhanov [1992], attributed to Joesph Stalin

[1]

We will make every vote count.
We are committed to ensuring that 2015 will be the last federal election conducted under the first-past-the-post voting system.

Liberal Party of Canada [2015]

Fundamentally, this thesis is about voting, as studied from the perspective of artificial intelligence. To say more than this requires some shared context between the reader and the author, so the first order of business must be to circumscribe the fields of study involved, at a high level.

Artificial intelligence (AI) is the subfield of computer science concerned with making programs or systems that behave intelligently, typically to the benefit of a human user or

[1]Translated. Original text: "Знаете, товарищи, — говорит Сталин, — что я думаю по этому поводу: я считаю, что совершенно неважно, кто и как будет в партии голосовать; но вот что чрезвычайно важно, это — кто и как будет считать голоса"

users. These programs are called "Agents". A somewhat broader definition of an agent is given by Russell and Norvig [Russell and Norvig, 1995], which can also encompass intelligent *human* actors. We adopt this definition throughout this thesis:

> An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors. A human agent has eyes, ears, and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. A robotic agent substitutes cameras and infrared range finders for the sensors and various motors for the effectors. A software agent has encoded bitstrings as its percepts and actions.

Agents can work alone, or in groups, and group interactions bring with them a wide variety of new and interesting problems. When the agents involved are human beings, the problems posed by these interactions fall under the purview of the social sciences. In contrast, when they involve software or robotic agents (acting on behalf of human users), the AI sub-field of multiagent systems is used. A mixture of human and software agents is also possible, and the line between multiagent systems and the social sciences (especially economics) is not always clear. Shoham and Leyton-Brown [2008] suggest a fairly broad definition:

> Multiagent systems are those systems that include multiple autonomous entities with either diverging information or diverging interests, or both.

Their definition is well suited to the problems and systems discussed in this thesis.

Example applications of multiagent systems research are quite varied, and include both competitive and cooperative groups of agents. Many different problems fall under the domain of automated resource allocation or matching, where a set of agents must be assigned a set of resources or items, to maximize the collective well-being of the agents. The United States National Kidney Exchange program uses multiagent systems research to automatically find assignments of donors (those willing to give a kidney) to recipients (those in need of one) [Abraham et al., 2007]. This system allows donors who are poor biological matches for their preferred recipient to swap recipients with others in similar situations, leading to improved outcomes for all.

Multiagent systems are also used to defend important targets from attackers, finding optimal strategies to protect airports, or prevent the poaching of endangered animals [Jiang et al., 2014; Pita et al., 2008]. In this domain, different models of attacking groups can be compared, and the resources required to optimally counter the threats posed by these groups can be computed, allowing for improved security, reduced costs, or both.

As a final example application, multiagent systems research can be used to coordinate a team of robotic soccer players that have been designed by different researchers, and that operate autonomously [Genter et al., 2015]. The agents involved were judged both by the final score of their team, and by their "sportsmanship", a quantified measure of how well they played as a team member (e.g. passing the ball to teammates, not colliding with friendly players).

The model of a "rational" agent has long been used in a number of different problem domains in economics [Foley, 2004] as an approximation of human behaviour. One such domain is the study of group decision making, called social choice [Arrow et al., 2010]. Social choice problems involve decisions (choices) made collectively by a group (society) of agents. Although agents may have dramatically different views about what constitutes a favourable decision, all agents are subject to the outcome selected by the group. For example, agents might decide whether or not to build a public works project like a swimming pool. Even agents who vote against the construction may be required to pay for a share of the construction. Similarly, in a federal election, Canadians collectively decide which party or parties should govern our country in the coming years. Although many members of the electorate may disagree with the outcome of the election, all must abide by the decision.

Within multiagent systems, the study of collective or group decision making is consequently referred to as "computational social choice" (COMSOC). The forthcoming Handbook of Computational Social Choice [Brandt et al., 2015b] defines the field as comprising two related research areas:

> First, researchers seek to apply computational paradigms and techniques to provide a better analysis of social choice mechanisms, and to construct new ones. Leveraging the theory of computer science, we see applications of computational complexity theory and approximation algorithms to social choice. Subfields of artificial intelligence such as machine learning, reasoning with uncertainty, knowledge representation, search, and constraint reasoning have also been applied to the same end.
>
> Second, researchers are studying the application of social choice theory to computational environments. For example, it has been suggested that social choice theory can provide tools for making joint decisions in multiagent systems, which are populated by heterogeneous, possibly selfish, software agents. Moreover, it is finding applications in group recommendation systems, information retrieval,

and crowdsourcing. While it is difficult to change a political voting system, such low-stake environments allow the designer to freely switch between choice mechanisms, and therefore provide an ideal testbed for ideas coming from social choice theory.

This thesis primarily concerns the first area, though naturally the new social choice techniques that are developed could find applications in the second.

An early example of computational social choice research included the use of computational complexity theory to study and circumvent impossibility results relating to strategic voting [Bartholdi III and Orlin, 1991; Bartholdi III et al., 1989]. More recent projects include the computational analysis of combinations of voting systems [Walsh and Xia, 2012; Nina Narodytska et al., 2012], and the use of techniques from optimization and statistics to decide elections with incomplete information generated under different models [Xia and Conitzer, 2008; Lu and Boutilier, 2011b].

A concrete example will aid the reader's appreciation of the difficulties that arise when making group decisions, and also allow the introduction of a visual representation of votes common throughout the remainder of the thesis. The preferences of a group of n **identical** agents are represented by a "vote vector", adopting Borda's notation, reproduced from Black et al. [1958]. The vertical arrow shown in Figure 1.1 below depicts an ordering over a set of four alternatives. Alternative a is ranked first (highest; best), while alternative d is ranked last (lowest; worst). The **n** placed below the vector indicates that there are n agents who order the four candidates in this way. The preferences of other agents may be represented with adjacent vote vectors showing other orderings, along with the number of agents expressing these orderings.

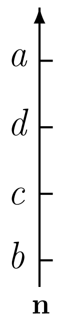


Figure 1.1: An example vote vector.

As a full example, consider a "preference profile", depicting the preferences of a group of agents deciding between alternatives a through d. Each vote vector in Figure 1.2 shows the preferences for a group of agents. The 15 leftmost voters rank alternative a highest, then d, then c, and finally b. In contrast, the 10 voters represented by the arrow second from the right think alternative c is best, then b, then d, and finally rank a last of all.

A common, and perhaps natural, way to reach a group decision given a preference profile of this kind would be the **Plurality** system, familiar to the reader through exposure to North American politics, committee meetings, or popular talent contests like "Canadian Idol" [2]. In **Plurality**, each voter assigns a single point to their most preferred candidate (i.e. the one ranked highest in their preferences). This might be done with a show of hands, the casting of a ballot, or a phone call. The candidate receiving the greatest number of points is declared the winner. In the preference profile above, note that although a receives the largest number of votes when **Plurality** is used, a substantial majority of the electorate (68%) prefer *every* other candidate to a. This suggests a is a poor choice to be the winner of the election: most voters think a is the worst choice possible.

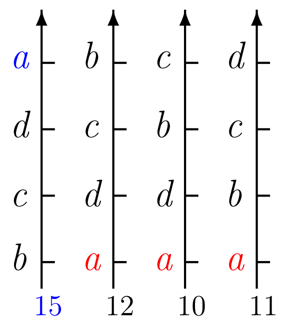


Figure 1.2: An example preference profile illustrating the difficulties inherent in making principled group decisions.

If a is thought to be a poor or unfair choice, then the system must pick another winner instead. Supporters of candidate c might note that c appears in the highest *average* position across the ballots, and this is the argument for legitimacy underlying the **Borda** system. Supporters of d would notice that a strict majority of voters prefer d to c, and indeed, a majority prefer d to each of the other candidates, when they are considered one at a time. This argument underlies the **Condorcet** system. Although the **Condorcet** argument is compelling, not every election has a winner that satisfies this condition. The reader should now have a taste of the difficulties inherent in making principled group decisions, and these are discussed more formally in later sections.

[2] Provided that viewers are limited to one vote each.

Machine learning (ML) is another subfield of AI research. While the subfield of multiagent systems focuses on the interactions between agents, machine learning research concerns the automatic construction of a predictive model of some environment or process by a computational agent, using data or observations taken from that environment. There are many machine learning techniques, and many different problem areas, but a common theme involves phrasing the act of learning as an optimization problem, wherein salient features of the observations are mapped to different labels or actions to minimize error or maximize reward. Examples applications of machine learning at large include automated speech and language processing (e.g. [Yu et al., 2012]), the automated recommendation of films and television shows to users based on their past ratings of other content (e.g. [Zhou et al., 2008]), and learning the parameters of a probabilistic model of user rankings selected from a known family (e.g. [Azari Soufiani et al., 2013]). Some of these applications will be discussed in more detail later in the thesis.

1.1 Thesis Statement

The definitions above provide the context necessary to state the fundamental question addressed within this thesis: **Can existing techniques from machine learning be used in a novel way in order to provide improved computational social choice solutions and hence, to better make group decisions in multiagent systems?** Answering this question reduces to answering three related ones, sometimes in several different ways, or by considering several competing approaches.

The **first** sub-question is simply, how *exactly* should existing machine learning techniques be applied to problems in group decision making? Chapters 3 and 4 provide an answer to this question. Existing ML techniques can be applied to help make better group decisions via voting *when agents have incomplete information*. An example that illustrates the difficulties inherent in these problems is the preference profile shown in Figure 1.3.

The set of preferences represented by the vote vector second from the right is incomplete, and does not specify the relative ordering of candidates a, c and d. This is denoted with the '?' symbol in vector. A natural way to treat this missing information is to ignore it in the calculation of the winner. If the **Borda** method is applied to these ballots, and missing information is ignored, a will be found to appear in the highest average position. However, much like how different voting systems lead to more or less principled decisions, different treatments of the missing preference information can lead to more or less principled decisions. In particular, there are reasonable ways of treating the missing information from this example under which c is the winner (imputing missing information with that of similar

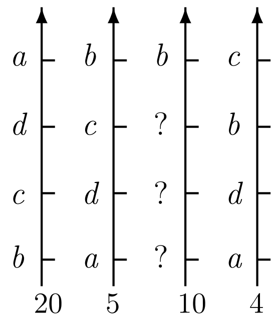


Figure 1.3: A preference profile illustrating the difficulties of group decision making with incomplete information.

voters) and under which d is the winner (there exists a completion under which d wins by the largest margin). Deciding how to treat missing information can impact the quality of group decisions to much the same extent as deciding which voting system to use, and is a challenging question to address.

This problem is described in detail in Chapter 3, where the shortcomings of existing approaches are presented using examples. Chapter 4 then proposes a new approach, based on using conventional machine learning algorithms intended for use in automatic classification, to *impute* missing information. The new approach is justified philosophically by showing that an idealized learning algorithm will cause decisions made by the system to match those in an idealized election. This philosophical justification is supported at length by validations in subsequent chapters using real-world data.

The **second** sub-question concerns whether the proposed system is *useful*. In particular, does the proposed system actually work? Does it make group decisions that are fairer, more reasonable, or otherwise "better" than existing systems meant for the same purpose? When decisions are made with incomplete information, exactly what constitutes the "right" decision is not always clear, or even knowable. Chapter 5 answers these questions affirmatively using a novel experiment design based on ablation, and repeated trials, applied to a number of large real-world datasets from the Preflib repository [Mattei and Walsh, 2013]. Chapter 6 builds on this initial experiment with a robust experimental testbed, allowing for a wide range of empirical questions to be answered using the same basic experimental design. Later chapters evaluate refinements of the initial technique using this testbed, and it could be used much more broadly by other social choice researchers in the future. Chapter 7 proposes a refinement using machine learning algorithms for predicting *sequences* rather than classification.

The **third** and final sub-question is: Which machine learning algorithms are "fairest" for use in group decision making? This is a deceptively simple question, and cannot be answered empirically without first specifying what it means for a learning algorithm to be "fair". Chapter 4 co-opts theoretical results from social choice proper by showing that many machine learning algorithms can be mapped directly to voting systems, and studied in that context. Building on longstanding results from the study of fair voting systems, the question is answered with the non-sequitur "None.". Chapter 8 builds on this idea by proposing a refined set of axioms specifically for use with learning algorithms, and also examines the axiomatic properties of combinations of learning algorithms and voting rules. Chapter 9 assesses machine learning algorithms for their propensity to induce voters to lie by casting strategic ballots rather than stating their true opinions, and proposes some techniques to allow voting systems to resist this behaviour.

Surrounding these components, Chapter 2 presents relevant background material from computational social choice and machine learning required for the reader to follow the remaining chapters. Chapter 10 ends the thesis by providing a discussion of the broader implications of the work, and what conclusions can be drawn, as well as avenues for future work, building upon the results presented here. The chapter also shows an example application domain for imputation-based social choice, in which the proposed approach offers marked advantages. Finally, the chapter contextualizes the results from other chapters alongside related work from the field, although more detailed comparisons with related works can be found throughout the thesis in the appropriate chapters.

Overall, the thesis offers a new way to approach social choice with incomplete information. It advances the state of the art in terms of empirical performance, theoretically connects machine learning to social choice in a surprising way, and provides a comprehensive testbed system, providing value to researchers in the computational social choice community. Further, its techniques are broadly applicable to problems in multiagent systems, and to low-stakes human electoral contests, where they facilitate better decision making with minimal human efforts. Finally, the thesis may offer value to researchers in the social sciences, as an extension of the idea of voting correctly [Lau and Redlawsk, 1997], or because of the theoretical connection between social choice and machine learning.

Chapter 2

Background

Today's posterior distribution is tomorrow's prior

Dennis Lindley [1970]

Before proceeding, it may be useful for readers to review relevant background material, to familiarize themselves with the fields of computational social choice and machine learning in greater detail than was provided in the introduction. These two fields are discussed at some length over the course of this chapter, including important work related to the general topics addressed in subsequent chapters. Although more detail is provided for complex topics in later chapters, introductory topics and some notation are defined only here.

2.1 What is Social Choice?

To function in everyday life, a person or an agent must make decisions. One must decide whether to got to get out of bed, or to press the snooze button; whether to walk to work or to drive; and whether to pack a lunch or to buy one. In addition, most of us also make daily judgements: Does an employee's work merit a raise or special recognition? Has a ball rolled out of bounds, or is it still in play? Is this thesis well written, or is it difficult to understand?

In the examples given above, the judgements and decisions are generally made on an individual basis: individuals decide for themselves whether to get out of bed or to press the snooze button. However, people often make decisions as a *group*, rather than alone.

Friends deciding where to eat a meal all have their own opinions as to which location would be best, but nonetheless, all would prefer to eat together than to eat alone. Members of a hiring committee may have different opinions about which candidate would be the best to fill a new role, but ultimately they must hire only one person — they cannot each hire their favourite. On a larger scale, voters in a national election may have radically different views on which party should form the government, but all must actually live with the government selected (barring a revolution). All of these are examples of situations where a *social choice* or *group decision* (rather than an individual choice or decision) must be made.

Social choice then, is the field of study concerned with *group decision making*. While such studies are obvious intimately concerned with the fields of political science and economics, of late they have been of increasing interest to computer scientists as well, as will be explained later in this section. The set of problems considered by social choice is any situation where a group of entities (people, or models of people, in the social sciences, but often abstract rational or computational intelligences in computer science) with differing individual preferences must reach a collective decision or judgement.

Although at first glace social choices may not seem more complex than individual choices, there are in fact many problems that arise when making decisions that a group must abide by collectively. Returning to the familiar example of selecting a restaurant to eat at as a group, one can imagine many different systems for reaching a group decision, all of which have their own advantages. A simple system might involve a form of *dictatorship*, where each participant may name any restaurant of their liking, and where the group's decision is based on a random selection of one such opinion. Although simple, this solution could lead to poor outcomes. For example, the dictator's preference might be for sushi when no one else eats fish. Although all the individuals have stated their individual choices, nearly no one is made happy by the group decision.

For this reason, most social choice systems require more detailed information about the preferences of the agents involved. One general framework, which is the focus of this thesis and to which we shall return in greater detail and formality in subsequent sections, is voting based on ranked preferences. In this framework, the voting system is operated by an abstract *centre* (i.e. a central person, agent, or system). The centre presents the agents with a list of *alternatives* or *candidates*: the different decisions or judgements they can make. The agents then each provide to the centre a *ranking* or *ordering* of the alternatives, and the centre selects the group decision on the basis of these orderings. For example, friends deciding where to eat might decide to pick from among a finite set of restaurants, and would each indicate which restaurant they thought was best, second best, third best, and so on. The centre could then select a restaurant that everyone thought was acceptable, if not their most preferred, by any number of sensible systems; several are discussed later in this chapter.

Throughout this work, we will operate within the above framework. It is worth mentioning that this is not the only way to make social choices. An immediate drawback is that it does not allow voters to express the raw intensity of their preferences. That is, while one can rank sushi restaurants last on their ballot, one cannot indicate whether this is because they would rather eat elsewhere, but would tolerate it, or because they have a deadly fish allergy and would die upon entering the building. One reason for preferring a system that does not extract this sort of information is that such facts are difficult to pin down the value of quantitatively. If one adopts subjective, individual, scales then saying one likes sushi "-1,000" might mean a deadly allergy for one person, and or simply great disgust for another. Further, even if a uniform scale is defined that agents share, then someone must define it, and implicitly, must define value relationships for the agents that may not be universal. For example, if a designer says that a liking sushi "-1,000" points means it will kill the agent to eat it, and liking it "-100" points means the agent thinks sushi is disgusting, the designer is implicitly saying that disliking something is about one tenth as bad as dying. These sorts of judgements seem dubious in general, and only become more complex as domains contain more varied alternatives. Consequently, the thesis is not concerned with more complex frameworks here (more formally, it assumes only subjective, and not objective, utilities, can exist, so that only relative rankings of alternatives can have external validity)[1].

Central to the conventional social choice framework is the formal idea of a *ranking*. Although the term 'ranking' will be used somewhat loosely throughout the thesis, it will always refer to some form of ordering over the set of alternatives. Formally, a *total* or *linear ordering* $\succ$[2] is a binary relation defined on members of a set $C = \{c_1, ..., c_k\}$ that is transitive (i.e. $c_x \succ c_y \wedge c_y \succ c_z \leftrightarrow c_x \succ c_z$), antisymmetric (i.e. $c_i \succ c_j \leftrightarrow c_j \not\succ c_i$ unless $c_i = c_j$), and reflexive (i.e. $c_i \succ c_i$), as well as being defined for every member of C with respect to every other member (i.e. total). Less formally, these conditions correspond to the idea that a total ordering of the candidates ranks all candidates into an ordered list. Naturally every candidate needs to come either before or after every other one (antisymmetry and totality), a candidate has the same ranking as itself (reflexivity), and

[1]However, the tools and techniques outlined in the thesis could be adapted to work with cardinal utilities, by using regression techniques instead of classification techniques, and in some domains (like medical decision making [Dolan et al., 2005; Miyamoto and Eraker, 1984; Nord, 1994; Gerard and Mooney, 1993; Grosse, 2008]), interpersonal comparisons of utility are used effectively, though not always without controversy.

[2]$\succ$ may be read as "is preferred to", although this semantic meaning doesn't quite hold when the reflexive property is taken into account.

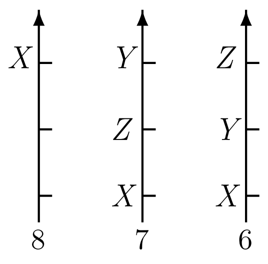


Figure 2.1: Borda's example preference profile.

the order is self-consistent so if a candidate comes before some other candidate, it must also come before everything that follows that candidate (transitivity). For now, 'ranking' will be used to refer to a total ordering over the candidates, but in other parts of the thesis, it may refer to a partial ordering, in which not every candidate needs to be ranked relative to every other one (i.e. dropping the totality requirement). The term ranking is often used synonymously with *preference* in the thesis (e.g. "a voter's preference over the candidates" or "a voter's ranking of the alternatives"), following convention. A set of rankings is often referred to as a *preference profile*.

2.1.1 A Brief History of Social Choice

[3] Systems for making social choices are ancient and myriad, but the *study* of such systems in a formal way did not begin until more recently. Some of the earliest attempts are made known to us through the work of Black et al. [1958], who recovered and popularized the early work of Borda [de Borda, 1781] and Condorcet [de Caritat, 1785]. Although earlier formal work on voting exists (see [Hägele and Pukelsheim, 2001]), the efforts of Borda and Condorcet are widely considered the first serious foray into the theory of social choice.

Borda was first to consider this topic in his 1770 presentation to the French academy of Sciences entitled ' *Sur la Forme des Élections* ', which was later published[4] as a paper in 1781 [de Borda, 1781]. The paper describes Borda's rule, a member of the now widely used family of election systems known as positional scoring rules described later in this section. Borda's line of reasoning began with the recognition that merely asking each voter

[3]Some portions of this section are based on text which appeared in the author's comprehensive examination document.

[4]Apparently through the efforts of Condorcet, as a strawman with which to contrast his own work. [Black et al., 1958, Introduction]

in an election for their most preferred candidate (a system denoted **Plurality**, and widely used in Borda's time as well as our own) could produce undesirable outcomes. He gives an example of a election with 3 candidates, shown in Figure 2.1[5]. Each vector depicts an ordering over the three candidates X, Y, and Z. The number at the bottom of a vector denotes the number of electors casting their ballots thus. For instance, the leftmost vector denotes 8 voters who have ranked X first, and who have no particular preference over the other candidates. The middle vector denotes 7 voters who rank Y before Z, and Z before X. Such orders were assumed to be transitive. The paradox in the example ballots lies in the fact that under the **Plurality** rule, candidate X is elected, even though a majority of the electorate prefer any alternative to X. Borda held that the correct treatment would be to elect one of Y or Z instead in this case. His proposed rule was to treat each vote as stating that a candidate has defeated everyone below it, and to assign a candidate p points for each such victory, and a base of q points for appearing in last place in the ranking[6]. This leads to a family of (equivalent) voting rules. A **Borda** rule is any voting rule such that a candidate receives $np + q$ points for appearing ahead of n other candidates on a given ballot, for some positive value of p and non-negative value of q. All rules make identical decisions, independent of what values p and q are adopted, so for the remainder of the thesis, $p = 1$ and $q = 0$ are used. The candidate to receive the largest number of points across all rankings is the winner. This amounts to selecting the candidate that appears in the highest average position across all the rankings[7].

Borda's work was quickly followed by that of Condorcet [Black et al., 1958][de Caritat, 1785]. Condorcet was interested in constructing a theory of voting derived from the theory of probabilities. Initially, Condorcet considers the problem of a jury deciding upon the

[5]reproduced from [Black et al., 1958], with Borda's original notation, but the names of the candidates changed.

[6]For example, with $p = 1$ and $q = 0$, candidate X would receive a score of $8 \times 2 + 7 \times 0 + 6 \times 0$ under the rankings in Figure 2.1.

[7]As an aside, Borda's justification for this choice of function, where the differences between the scores assigned to any two adjacent positions in the scoring vector are identical, provides an early glimpse of a justification for the use of ordinal preferences (i.e. asking each voter for a ranking, rather than an intensity for each candidate), later utilized independently by Arrow [Arrow, 2012]. An ordinal preference relation is just one based on a binary relation over the set of candidates, exactly as defined above. Ordinal preferences are contrasted with real-valued or utilitarian preference relations, which map each candidate to a *utility* corresponding to the intensity with which a given voter prefers a given candidates, and where a ballot takes the form of a specification of the utility a voter would derive from each candidate. Since the only information given when the centre is presented with an ordinal ballot $X \succ Y \succ Z$ is that X is preferred to Y, and Y preferred to Z, and nothing about the relative magnitude of that preference, Borda proposes the adoption of something akin to a uniform prior over the relative differences in the individual's perceived desire for the different candidates to win. Precisely the same concept is utilized as part of Arrow's later arguments in favour of using ordinal preferences.

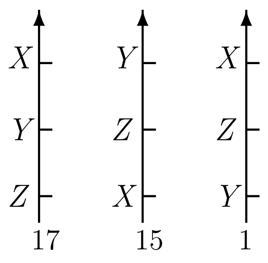


Figure 2.2: Condorcet's example preference profile.

innocence or guilt of some party. Individual members of the jury may decide the case rightly or wrongly, but, provided that the probability they decide rightly exceeds 0.5, adding more members to the jury will produce an increase in the probability that the jury *collectively* arrives at the correct decision[8]. In this case, the correct decision corresponds to a truth about the external world. For example, in a trial, the accused either truly is, or truly is not, guilty, so there is a correct answer to the question of their guilt. Supposing that the probability of a juror giving the correct answer is v, and the incorrect answer is e (and $v + e = 1$)[9]. Then, observing h jurors in favour of conviction and k against, the likelihood that a guilty verdict is incorrect is given by $\frac{v^k e^h}{v^h e^k + v^k e^h}$. Moving from the realm of juries into the realm of elections, Condorcet first notes the same problem as Borda with respect to the **Plurality** rule, and refines this by showing that X loses in a runoff election to *every* other candidate in the election. This property would become known as being a *Condorcet Loser*. Following on this, Condorcet suggested that one should view an election as a set of binary decisions over each pair of candidates. For instance, in an election with three candidates, $C = \{X, Y, Z\}$, there are really three questions being considered. Respectively, these questions consist of whether $X \succ Y$, $Y \succ Z$, and $X \succ Z$, where $\succ$ is used in this case to denote the aggregate views of the electorate in each binary contest. If each question is treated separately, then the problem looks very much like that of the jury described above. The electorate's votes can simply be interpreted as approximations of the truth or falsehood of propositions about the pairwise orderings of the candidates. That is, in much the same way as the question of the accused's guilt in a trial has a correct answer, the question of whether $X \succ Y$ is assumed to have a correct answer. Condorcet assumed again that individual preferences take the form of linear orders over the candidates, so one

[8] Condorcet did not consider the possibility that the jurors' individual decisions might be correlated, but this area has since been developed more richly, e.g. [Conitzer, 2013; Dietrich and Spiekermann, 2013; Tsang et al., 2015]

[9] And that the jurors' probabilities are independently and identically distributed.

can infer an individual's opinion on any pairwise comparison using transitivity. To select a winner with this approach, the propositions are treated as independent events, and one multiplies together the likelihoods derived for each via the jury procedure. Unfortunately this approach does not yield sensible results. Condorcet constructs an example, which is illustrated using Borda's notation in Figure 2.2. In this example the proposition that $X \succ Y$ is true is inferred be true with likelihood $\frac{v^3}{v^3+e^3}$, since the margin of victory for X over Y is 3 votes in total (18 rank $X \succ Y$, 15 rank $Y \succ X$). Similarly, the likelihood that $X \succ Z$ is computed to be $\frac{v^3}{v^3+e^3}$. The joint likelihood of X being the 'correct' candidate in every pairwise contest then, is $\frac{v^6}{v^6+2v^3e^3+e^6}$. The same computation for Y yields a likelihood of being the 'correct' candidate of $\frac{v^{31}e^3}{v^{34}+v^{31}e^3+v^3e^{31}+e^{34}}$. However, recall that the probability v is not known apriori, and Condorcet has only assumed it exceeds 0.5. If $v \sim 1$, then X has a higher total likelihood of being the 'correct' winner. On the other hand, if $v \sim 0.5$, Y's likelihood is higher[10]. The selection of the correct winner will thus depend on how well the electorate approximates the 'correct' knowledge of which candidates are better than each other (v), which is unknown to the centre.

At this point, Condorcet abandons the probabilistic approach, and asserts using 'straightforward reasoning' that X should be chosen, as a majority of voters support $X \succ Y$ and $X \succ Z$, which matters more than the margin of victory. This concept in turn became known as the *Condorcet Winner* — a candidate which wins all pairwise contests against others in runoff elections. Interestingly, the winner selected if v is close to 0.5 will correspond to the Borda winner instead [Black et al., 1958, introduction], which Condorcet had dismissed at the beginning of his argument. Condorcet would later claim that his method was to be preferred instead on the basis of the *independence of irrelevant alternatives* axiom which Arrow formalized much later (discussed at some length later in this thesis), which his own method satisfies but which Borda's does not.

2.1.2 Voting Rules Used in this Thesis

This thesis makes use of a number of different voting rules, which are presented formally here for reference. There are a great many other voting rules that are not presented. Some of these may be mentioned elsewhere in the text, but will be accompanied by a reference if further clarification is needed. The systems are presented somewhat informally here, and more formal structures are constructed later in the thesis when greater detail is required, as in the theoretical analysis of Chapter 8.

[10]Incidentally, the same principle underlies ensemble methods used in machine learning [Freund and Schapire, 1997; Breiman, 1996]

The first voting rule presented is **Plurality**. The **Plurality** system is likely to be familiar to readers, as it continues to be very widely used, and is often the system used in informal contexts when a group decides to put an issue to a vote. Within the framework for voting discussed earlier in this chapter, **Plurality** operates by having each voter submit a ranking of the candidates. A voter's most preferred (i.e. top-ranked) candidate receives a single point. The candidate that receives the most points is the winner. Formally, given a set of rankings B over a set of candidates C,

$$\text{Plurality}(B) = \arg\max_{c \in C} \sum_{\succ_i \in B} I(c \succ_i c' \quad \forall c' \in C \setminus c)$$

where I is a indicator function (1 if the argument is true, 0 otherwise). As outlined in the previous section, **Plurality** can make problematic selections if a decision is being made over a set of more than two candidates.

An alternative way to represent **Plurality** is via a $1 \times |C|$ scoring vector, $s = \{1, 0, ..., 0\}$. The behaviour of the rule can then be written:

$$\text{Plurality}(B) = \arg\max_{c \in C} \sum_{\succ_i \in B} s_{\succ_i, c}$$

where $s_{\succ_i, c}$ is the component of s corresponding to the location of candidate c in ranking $\succ_i$. This approach allows for easy representation of several other rules, by using different scoring vectors, but identical formulations. For instance **Borda**'s rule, discussed in the previous section is identical, but with a scoring vector $s_{borda} = \{|C|, ..., 1\}$. Rules of this form are called positional scoring rules, since they give different scores to candidates depending on their positions in each ranking. There are two other positional scoring rules not previously mentioned in the thesis, but used widely in later chapters. The first is **Veto**, which uses the scoring vector $s_{veto} = \{1, ..., 1, 0\}$. Effectively, this allows each voter to vote *against* a single candidate, but not in favour of any particular candidate (the opposite of **Plurality**). The second is **K-Approval**, which is a family of positional scoring rules, where the first k elements of s are set to 1, and the remainder are set to 0. Effectively this gives each voter the ability to pick their k most preferred candidates, and give one point to each. The thesis will generally be concerned with $\lceil \frac{|C|}{2} \rceil$**-Approval**, where voters sort the candidates into equally (or nearly-equally) sized groups, those that are "above average" (awarded one point each), and those that are "below average" (awarded no points).

The only other voting rule used widely in the thesis is the **Copeland** voting rule, which is not a positional scoring rule. **Copeland** is based on Condorcet's idea of pairwise contents. Informally, the rule holds a pairwise contest between each pair of candidates. The winner of the contest receives one point, while the loser receives zero. The handling of ties can have important implications [Faliszewski et al., 2008]. The version used in this thesis is called Copeland$_0$, in which no points are awarded to either player in the event of a tie, the other common variant being Copeland$_{\frac{1}{2}}$, where each participant in a tie gets half a point. Obviously **Copeland** will output the Condorcet winner when one exists, but as a Condorcet extension, the rule can also select a winner in some other circumstances. However, it is very prone to ties when no Condorcet winner exists.

2.1.3 Axiomatic Social Choice and Impossibility Results

Following Condorcet, there was little work on social choice until the emergence of axiomatic social choice and the impossibility results of the next section. The efforts of Laplace, Nanson, Galton, and Dodgson are notable exceptions [Black et al., 1958]. The revitalization of the field, in its modern form, appears to have been spurred by increasing interest in the problem of corporate governance and committees in general. Like Borda and Condorcet before them, the scholars of this era would realize the problems inherent in plurality voting, but they would also go further than earlier authors toward resolving the issue. The two most important authors from this era are Black and Arrow.

Black's interest in social choice came in part from a desire to create a 'pure science of politics' [Black et al., 1958, introduction]. His work 'The Theory of Committees and Elections' [Black et al., 1958] constitutes significant progress toward this goal. The theory supposes that a committee (a group of electors) will consider a series of binary issues, perhaps mediated by a chair. Black supposed that a series of binary issues would be sufficient to represent an issue with more than two alternatives, by treating one of the alternatives as a 'bill' or 'motion' put to the electorate. Each of the other alternatives may be viewed as an amendment to such a bill, which can either be passed or rejected. If passed, it replaces the original bill, and may be subject to further amendments (i.e. binary contests with the remaining candidates). Eventually, when no other amendments are put forward, the bill is voted on in its current state, which can be viewed as a binary contest with just one more alternative (i.e. the status quo). This process is sometimes called *sequential plurality voting*.

Black first noticed that, in the case where the set of alternatives can be *ordered*, and where there are no external factors to consider, the procedure above will actually produce a reasonable result, independent of the order in which the proposals are considered. For example, imagine a committee deciding where to set the minimum wage. Individual committee members will each have a *most preferred* value for the minimum wage. The closer

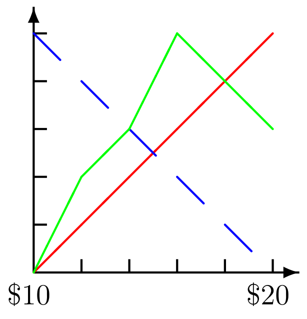


Figure 2.3: Example Single-Peaked Preferences

a proposed value is to their preferred value, the more willing they are to accept it, and vice versa. If we plot the preferences of voters across the candidates, assigning a value of $|C|$ to the voter's first preference, and $|C| - k$ to their $(k-1)^{th}$ preference, we would thus expect to see something like the graph in Figure 2.3. The dashed blue line shows a voter who prefers a minimum wage that is as low as possible. The red line shows a voter with the opposite preference ordering. The lighter green line shows someone with more complex preferences, preferring a value in the middle, and in general preferring a value higher than their preferred one to a value less than their preferred one. Notice that, when plotted against increasing minimum wage, all three preference functions are convex. Such preference profiles are called *single-peaked*.

Black observed that, when a preference profile is single-peaked, the most preferred value of the vote with the median peak will defeat all alternative candidates in pairwise plurality elections, making them a Condorcet winner. The median peaked vote is the one about which the set of votes would be equally partitioned if it were sorted according to the location of the most preferred candidate on each ballot (i.e. the peak on each ballot) in the ordering under which the preferences are single-peaked. In the preceding example, this is the voter represented by the green line.

It is easy to see why the first preference of the median peak ballot (C_m) will defeat all other candidates. Let $\pi_\wedge$ be the ordering over C such that the preferences are single peaked, and suppose that C_m faces a candidate such that $\pi_\wedge(C_m) > \pi_\wedge(C^*)$ (that is, a candidate C^* to the left of C_m under $\pi_\wedge$). All voters with peaks to the right of C_m have preference relations such that $C_m \succ C^*$, by definition of single-peakedness. The number of such voters is $\geq \frac{|C|}{2} - 1$. Since the median peak ballot also has $C_m \succ C^*$, it follows that C_m defeats C^* in a binary contest under plurality, and a similar argument can be made for candidates such that $\pi_\wedge(C_m) < \pi_\wedge(C^*)$. Unfortunately Black also discovered, 'with something akin to physical sickness' [Black et al., 1958, introduction], that when the preference profile is not single-peaked, the winner selected via sequential plurality voting will be dependent on the order in which the alternatives are considered, which would seem to yield unfair results.

This in turn leads us to Arrow's theorem, and the beginnings of axiomatic social choice. Arrow independently considered the problem of decision making in committees, and arrived at conclusions identical to those of Black, only to read of Black's results shortly afterwards [Black et al., 1958, introduction]. A natural subsequent question was whether there was an alternative voting system to the sequential plurality supposed by Black, under which sensible winners might be selected. In his doctoral thesis *Social Choice and Individual Values* [Arrow, 2012], Arrow outlines four axioms which he believes any reasonable voting system must satisfy, given below:

- **Non-dictatorship**: The voting system must not function as a dictatorship — the outcome must depend on more than a single elector's opinion.
- **Universality**: There must be no apriori restriction on the preferences a voter may specify. Note that this prevents us from requiring preferences to be single-peaked.
- **Independence of Irrelevant Alternatives (IIA)**: Whether C_1 defeats C_2 in the election should not depend on whether or not a third candidate C_3 exists (i.e. no chance of vote splitting).
- **Unanimity**: If all the voters prefer C_1 to C_2, then C_2 cannot defeat C_1 in the election.

Arrow proved that there exists no voting system which can satisfy all four criteria simultaneously. A brief form of the argument is provided by Geanakoplos's modern proof [Geanakoplos, 2005]. Since the four criteria appear to be vital for any fair voting system[11], this poses a significant challenge to the design of such systems. There are several approaches to getting around Arrow's theorem. First, for many domains universality may not be required. As in the example used by Black, some issues may really have single-peaked preferences. Alternatively though, some criticisms of the theory include Black's contention [Black, 1969] that the conditions suggested by Arrow may not be good criteria on which to judge a voting system. A system in which only motions receiving unanimous support are adopted essentially satisfies Arrow's conditions, but is highly dysfunctional in

[11]Though some authors dispute this, see e.g. [Samuelson, 1977]

practice. Similarly, if one allows for the possibility of intransitive social outcomes (presumably with some other mechanism for resolving them), then Arrow's theorem need not apply. The axiomatic study of social choice systems is discussed in greater detail in Chapter 8.

An important theorem very similar in form to Arrow's is the Gibbard-Satterthwaite Theorem [Gibbard, 1973; Satterthwaite, 1975]. Many voting systems suffer from the problem of strategic voting, or *manipulation*. Ideally, voting systems are supposed to aggregate the rankings of the group to reach a consensus decision. However, under **Plurality**, for example, voters can often obtain outcomes that are favourable to them and are (perhaps) less reflective of the group's preferences, by submitting an insincere ranking to the centre. For example, consider the ballots in Figure 2.1 near the start of the chapter. Under **Plurality**, any two of the six voters casting ballots that rank C first could instead submit an insincere ballot ranking B first. This would cause B to win the election. However, it violates the spirit in which the decision was being made (i.e. that the candidate that is the favourite of the largest number should win). A system in which no voter can benefit by submitting an insincere ballot is said to be *strategy-proof*. The Gibbard-Satterthwaite Theorem adds an axiom to the set that any reasonable voting system should satisfy:

1. Strategy-Proofness: No voter can benefit by reporting an untruthful ranking to the centre.

The theorem shows that a deterministic voting system (i.e. one that does not rely on randomness) that always picks a unique winner (i.e. no ties) cannot satisfy Universality and Strategy-Proofness unless it is also Dictatorial. As dictatorial voting systems are not very representative of the group's preferences, circumventions of this result are the subject of a great deal of study. This topic is covered in much greater detail in Chapter 9.

2.1.4 Measuring Similarities Between Rankings

A recurring topic in the thesis is the measurement of how similar two rankings are. These problems arise, for instance, when two competing voting systems have output rankings over the same set of alternatives, and a quantitative measurement is required to determine how much they differ from one another. The primary application of this measurement in the thesis is in assessing the performance of different voting rules empirically, when a collective ranking of candidates output by a particular system can be compared to a "correct" ranking by measuring the distance between them.

Perhaps the simplest such measurement is the Kendall Correlation [Kendall, 1938], denoted τ, and sometimes called Kendall's Tau. The Kendall correlation is proportionate to the smallest number of pairwise swaps of adjacent candidates required to transform one ranking into another. A pairwise swap of adjacent candidates consists of finding two candidates that are positioned one after another in a ranking, and swapping their positions. The count of such swaps is normalized by the maximum number possible, and scaled to lie between -1 (one sequence is the other backwards), and 1 (the two sequences are identical). More formally, the Kendall Correlation is given by:

$$\tau(\succ_1, \succ_2) = \frac{1}{|C|(|C|-1)} \sum_{c \in C} \sum_{c' \in C \setminus C} (I((c \succ_1 c' \wedge c \succ_2 c') \vee (c' \succ_1 c \wedge c' \succ_2 c))$$
$$-I((c \succ_1 c' \wedge c' \succ_2 c) \vee (c' \succ_1 c \wedge c \succ_2 c')))$$

The Kendall Correlation is a useful estimate of how closely two rankings are correlated, but it is not the only such measurement. Spearman's Footrule distance [Diaconis and Graham, 1977; Spearman, 1906] is a common alternative, which counts the distance between the position of each candidate in ranking $\succ_1$ and its position in $\succ_2$, rather than the overall number of swaps that might be required to transform one order into the other. Formally, this is given by

$$\delta_{SF}(\succ_1, \succ_2) = \sum_{i=1}^{|C|} |\text{Pos}(\succ_1, c_i) - \text{Pos}(\succ_2, c_i)|$$

where in this case, $\text{Pos}(\succ_i, c_j)$ denotes the number of candidates that follow candidate c_j in ranking $\succ_i$.

Other distance measures of this kind weight the candidates based on some measure of their importance when computing the similarity of the rankings. For example, one possible weighting for the Kendall Correlation would be:

$$\tau(\succ_1, \succ_2) = \frac{2}{|C|(|C|-1)(|C|+1)} \sum_{c \in C} \text{Pos}(\succ_1, c) \times$$
$$\sum_{c' \in C \setminus C} (I((c \succ_1 c' \wedge c \succ_2 c') \vee (c' \succ_1 c \wedge c' \succ_2 c))$$
$$-I((c \succ_1 c' \wedge c' \succ_2 c) \vee (c' \succ_1 c \wedge c \succ_2 c')))$$

under which candidates that are ranked higher in ranking $\succ_1$ contribute more to the correlation measure than other candidates do. Many other weightings are also possible.

2.1.5 Artificial Preference Distributions

The thesis will occasionally make use of two families of artificial preference distributions. These are described briefly here, and expanded upon later as needed.

An artificial preference distribution is a statistical process for generating the preferences of voters[12]. A very simple distribution (not used in the thesis) would be the uniform distribution, under which a ranking of the candidates is sampled uniformly at random from the set of all possible rankings. For instance, if there were three candidates (c_1, c_2, and c_3), then there are 6 possible total orderings of the candidates, and a uniform distribution would assign each of them to a given voter with probability $\frac{1}{6}$.

A widely used artificial preference distribution is the Mallows model [Mallows, 1957]. The Mallows model assumes that there is an objectively "correct" total ordering over the candidates, $\succ^*$, and that voters' preferences are "noisy" impressions of this correct ordering. The probability of assigning a ranking $\succ$ to a particular voter is proportionate to $e^{-\phi\Delta(\succ,\succ^*)}$, where Δ is the Kendall distance (not Kendall correlation), computing the number of pairwise adjacent swaps needed to convert $\succ$ to $\succ^*$. ϕ is called the *dispersion parameter*, and $0 < e^{-\phi} \leq 1$. At $e^{-\phi} = 1$, the Mallows model selects rankings uniformly at random. At $e^{-\phi} \sim 0$, the model will output only $\succ^*$ with high probability. In between, as ϕ decreases, and $e^{-\phi}$ increases, rankings that are increasingly far from $\succ^*$ will be output with higher and higher probability. In this respect, ϕ acts very much like the standard deviation σ of a Gaussian distribution, while $\succ^*$ acts very much like the mean.

The other artificial preference distribution used in this thesis is the Random Utility Model, or RUM [Plackett, 1975; Luce, 1959; Azari et al., 2012]. In this family of distributions, each candidate is assigned a "true" utility with respect to the group, that is, the real utility the candidate would provide collectively to the group, which is a well defined quantity for some problem domains. Individual agents are assigned individual utilities for a given candidate sampled from a probability distribution in the exponential family (e.g. Gaussians) with mean equal to the candidate's true utility, and a candidate-specific variance. Thus, each voter has a "noisy" impression of the true social utility of each candidate. Voters rank candidates in order of the individual voters' impressions of the candidates' utilities. The family of distributions is quite varied, but in general if two candidates have similar true utilities, and have large enough variances in the distributions of their individual utilities, voters will be more likely to disagree on which candidate is best. In contrast, if one candidate has a much larger true utility than another, and the variances are relatively small, nearly all of the electorate will agree on their relative order.

[12]For example, to create an artificial election for testing purposes.

2.2 What is Machine Learning?

As outlined in Chapter 1, this thesis is concerned with the application of techniques from machine learning to work in the area of computational social choice. Having covered the background material for computational social choice, we now move on to provide an introductory view of machine learning in general, and the specific machine learning tools that will be most relevant to the content of subsequent chapters.

2.2.1 The Basics

Although there is debate about its precise scope, at the broadest possible level, machine learning is the area of artificial intelligence research that is concerned with developing algorithms that can autonomously learn. This is the definition adopted by the Encyclopedia Britannica [Hosch, 2016]. More precisely, machine learning might be described as the study of algorithms that learn by experiencing or interacting with the world [Russell and Norvig, 1995]. Naturally, this is a very broad area, including applications that range from training robots to move or behave in a certain way via punishment and reward signals, through to simply finding (i.e. learning) statistical patterns in a database.

This thesis is concerned with problems that fall within the machine learning topic of supervised learning, in which some external signal is available to provide an algorithm with feedback on whether the patterns it has learned are more or less correct. Supervised learning stands in contrast to unsupervised learning, where no feedback is provided, and the algorithm simply looks for "interesting" patterns in the data. The analogous distinction in human learning would be between searching for an object while a peer indicates whether you are "hot" or "cold" (supervised learning), versus trying to find an object based only on a vague description of it (unsupervised learning). An example supervised learning application is classification. In classification, an agent or algorithm is given some example objects or data points, and told which class each belongs to. The algorithm then constructs a theory describing how the properties of the objects map to membership in different classes. The quality of the theory is known more or less exactly, because the algorithm's supervisor has told it the correct label for each object. In contrast, an unsupervised machine learning task like clustering involves having an algorithm find a set of k groups of objects such that the objects inside a group are similar to one another. The algorithm does not know whether it has discovered a "correct" partitioning of the objects, because no one has told it what a correct partitioning would look like precisely. The specific problem of interest in this thesis is in fact the classification problem mentioned above. A more formal description of classification will now be presented.

In a classification problem, a classification algorithm is given a set of *exemplars* $\mathbb{X}$, and a corresponding set of *labels* $\mathbb{Y}$, such that $|\mathbb{X}| = |\mathbb{Y}|$ (i.e. each label is paired with one exemplar). Typically an exemplar $x_i \in \mathbb{X}$ is a numerical vector of a constant length (i.e. x_i is the same length as x_j for all $x_i, x_j \in \mathbb{X}$), so that $\mathbb{X}$ takes the form of a matrix. Each row of the matrix describes a particular exemplar, and each column is called a *feature*. A classification algorithm's goal is to find a mapping f, from exemplars to labels, such that $f(x_i) = y_i$. Ideally such a mapping is general, and so can be applied to predict the labels of other exemplars that the algorithm has not seen before.

A concrete example might consist of a pedagogical classification task. For example, suppose one were teaching an online class to thousands of students. It is very easy to gather information about the students' interactions with course content through their web browsers. On the other hand, it is quite hard to predict in advance which students are going to fail the course[13]. Nonetheless, after the course has been completed, it is known which students passed and which students failed. A interested instructor might then want to develop a way to predict in advance whether a student will pass or fail, based on their behaviours early in the course. To accomplish this, the instructor might then gather statistics regarding the browsing behaviours of each student, and create a matrix of them, which would serve as the exemplar matrix $\mathbb{X}$. The label vector $\mathbb{Y}$ would have value 1 if the corresponding student passed the course, and 0 otherwise. An example of such a problem is shown in Table 2.1.

In the example data, the instructor has gathered four features for each of four students. For example, the instructor has measured that the student corresponding to exemplar x_1 spent 5 hours reading the course textbook during the first week of the class, and 6 hours on Facebook, a popular social media website the instructor suspects students are using when they ought to be studying. In contrast, the student corresponding to exemplar x_2 spent no time at all reading the course textbook, but 8 hours working on homework, and 4 hours watching the course lectures. Both students passed the course, indicated by their labels y_1 and y_2 having value 1. The *classification problem* here is to find a mapping from the times students spent doing various tasks (their features) to whether or not they eventually passed the course (their labels). Collectively, $\mathbb{X}$ and $\mathbb{Y}$ are said to be the *data* for this classification *problem instance.* If the instructor gathered the same measurements (i.e. the same features) from a different class of students, the classification problem would be the same, but the problem instance (and corresponding data) would be different, since the students in the other class are unlikely to have exactly the same values.

A *classification algorithm* for the purpose of this thesis is any function that accepts

[13]This is often surprisingly hard when teaching in person, let alone online!

	Time Reading Textbook	Time Doing Homework	Time on Facebook	Time Watching Lectures
x_1	5	5	6	0
x_2	0	8	2	4
x_3	2	3	8	3
x_4	12	14	0	3

	label (1=pass, 0=fail)
y_1	1
y_2	1
y_3	0
y_4	0

Table 2.1: Data for an example classification problem. The matrix of exemplars is shown on the left, with four named features corresponding to the time (in hours) that each student spent doing different tasks during the first week of an online course. The vector of labels is shown on the right.

the data for a classification problem instance, and outputs a predictive *model*. A model is function f that accepts an exemplar as input, and produces a label as output. An effective classification algorithm should output a model such that $f(x_i) = y_i$ all or most of the time. Even better, if a new exemplar is created after the model has been learned (x^*), then ideally the model should be able to deduce its correct label as well (i.e. $f(x^*) = y^*$).

The term "model" is used because in practice classification algorithms usually do not learn the true underlying pattern that caused certain labels to be assigned to certain exemplars. Instead, they learn some approximation of this pattern, which could be viewed as a model of the true process. For example, an algorithm applied to the data in Table 2.1 might learn two rules to describe the behaviour of students: first, that students who spend more than 6 hours a week on Facebook will fail the course, and second that students who spend more than 8 hours a week doing homework will fail the course. The model then consists of a function:

$$f(x_i) = \begin{cases} 1 & \text{if } x_{i,3} \leq 6 \wedge x_{i,2} \leq 8 \\ 0 & \text{otherwise} \end{cases}$$

where $x_{i,j}$ corresponds to the value of the feature in the j^{th} column of exemplar x_i.

The patterns described by this model are present in the exemplars that were provided, but the model is probably too simple to be an accurate representation of reality. An exceptionally skilled student might be able to finish their homework very quickly, and then choose to spend many hours on Facebook, for instance. This pattern of behaviour is not captured by the simple model that was learned. Nonetheless, models can be useful without being fully correct. The simple model described above would certainly be able to correctly identify many students in need of early intervention, and could form the basis of a simple automated system to alert the instructor when students are at risk of falling behind.

In this example, the classification problem involved selecting between only two classes (students who passed and students who failed). More complex problems might involve finding models that map from a vector of features to many different classes. For example, the instructor might wish to learn a model that predicts not only whether students passed or failed, but what their grades will be (i.e. A, B, C, D, or F). In this case the instructor might gather the same features as before, but the label vector would now have 5 possible values, perhaps 0, 1, 2, 3, 4, corresponding to the students' grades.

2.2.2 Relevant Learning Algorithms

There are a great many classification algorithms available for use. One reason for this is that different algorithms may learn models with very different forms, and some forms may be better suited to certain types of tasks. For example, some algorithms like the C4.5 decision tree learner [Quinlan, 2014, 1986] are designed to learn models that are (comparatively) easy for humans to understand. Their behaviours can be simplified down to a fairly simple rules, and parameters allow the user to force the algorithm to prioritize simpler or more complex rules. In contrast, the models produced by algorithms that generate artificial neural networks [Hinton et al., 2006] tend to be opaque and very difficult for humans to understand. However, opaque models can often provide more accurate predictions than more easily interpretable ones.

Another reason for the wide range of learning algorithms may be the No Free Lunch Theorems [Wolpert, 1996; Wolpert and Macready, 1997], which are discussed in greater detail later in the thesis. The core statement of the no Free Lunch Theorems amounts to the claim that there is no single "best" classification algorithm. An algorithm can only perform better than other algorithms in certain types of classification problems by performing worse in other types. Despite this, there are certainly algorithms that most practitioners would agree are objectively worse than others. This is because the sorts of problems humans are interested in solving have some commonalities, and so an algorithm might work better on these sorts of classification problems, and worse on very odd kinds of problems that are not especially interesting.

In this chapter, several algorithms are presented at a high level, with descriptions of the kinds of models they learn, and the general approach that is taken to learning them. In later chapters more details are provided for specific models as they come up. The interested reader is referred to Russell and Norvig [1995] part vi for a more detailed treatment, or to the papers cited during the discussion of each method.

Perhaps the simplest method for solving classification problems, though also a very inaccurate one, is to simply count the number of labels for each class and ignore the features entirely. The model that is output is either $f(x) = 1$ or $f(x) = 0$, depending on which class appeared more often in the input. Besides being ineffective on many problems, this approach has a rather uninteresting interpretation. Since the model contains no information about, for instance, which features are more or less predictive, it does not tell the user anything about the data. This model is not used in the thesis, but is included as a simple example.

Another comparatively simple model is the venerable logistic regression algorithm, originally due to Cox [1958], building on the earlier approach of Fisher [1936]. The approach is intended to solve only binary classification problems (i.e. problems with exactly two classes). Logistic regression assigns values of 0 and 1 to the labels that are used for the classes, and then outputs a model of the form $f(x) = \frac{1}{1+e^{(-xw+b)}}$, where w is a column vector of weights. The model will thus take the form of a multi-dimensional sigmoid function. Each feature x_j in x is multiplied by a corresponding weight w_j, and the resulting products are summed. A bias term b is added. When the sum is higher, the model outputs a value closer to 1. When the sum is lower, the model outputs a value closer to 0. The user is left to interpret the outputs, but often output values greater than 0.5 are assumed to predict membership in class 1, and values less than 0.5 to predict membership in class 0. Some example sigmoid curves are shown for a problem with only one feature in Figure 2.4. A much more detailed description of logistic regression appears near the end of Chapter 4.

A more complex classification algorithm is the Support Vector Machine learner, originally proposed by Cortes and Vapnik [1995]. Like logistic regression, support vector machines are intended to learn solutions only to binary classification problems. A support vector machine learns a model based around the idea of a “maximum-margin separating hyperplane”. Essentially, if each exemplar is taken to be a point in a space defined by the features, various planes could be drawn through the space to partition it into two pieces (a plane in more than two dimensions being called a “hyperplane”). If a plane can be drawn that puts all the members of one class on one side of the plane, and all the members of the other class on the opposite side, then the plane is a reasonable predictive model (predictions about the class of a new point can be made based on which side of the plane it falls on). However, if such a plane exists in a real-valued space, there will also be an

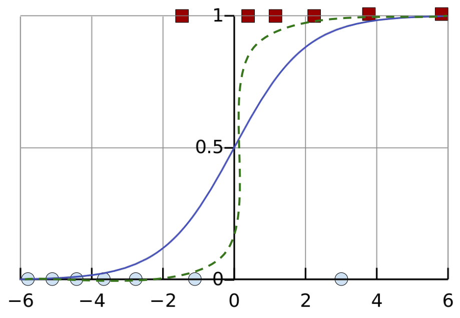


Figure 2.4: Some example logistic regression models. The horizontal axis corresponds to the values of a feature, while the vertical axis corresponds to the value of the labels. Points indicate examplars. A curve with a larger positive weight will be steeper than one with a smaller positive weight

infinite number of variations on the plane which still separate the two classes. The support vector machine will attempt to pick the plane that sits equidistant between points of the two classes, maximizing the "margin" between the observed distribution of the points of each class and the plane itself, according to the equation:

$$\underset{w}{\operatorname{argmin}}\, \lambda||w||^2 \frac{1}{|\mathbb{X}|} \sum_{x_i \in \mathbb{X}} \max(0, 1 - w y_i x_i)$$

where w is a vector of weights, one per feature of exemplar x_i (plus a bias term), label y_i is one of two possible values (1 or -1), and λ is a tuned parameter that controls the tradeoff between the size of the margin and the number of points that end up on the wrong side of the plane. The margin is maximized in the belief that future exemplars would have to be very dissimilar from their true class to end up on the wrong side of such a plane, and putting the plane equidistant between the two classes minimizes the risk of this for either class. Finding such a plane amounts to an optimization problem where, as in logistic regression, a vector of weights w and a bias term b are found. This process is depicted in Figure 2.5.

Additionally, support vector machines are capable of learning more complex partitioning surfaces by recourse to the so-called "kernel trick". Via the kernel trick, exemplars are projected from points in the original feature space (where there may not exist a hyperplane that separates them neatly into two classes) into a larger feature space where such a

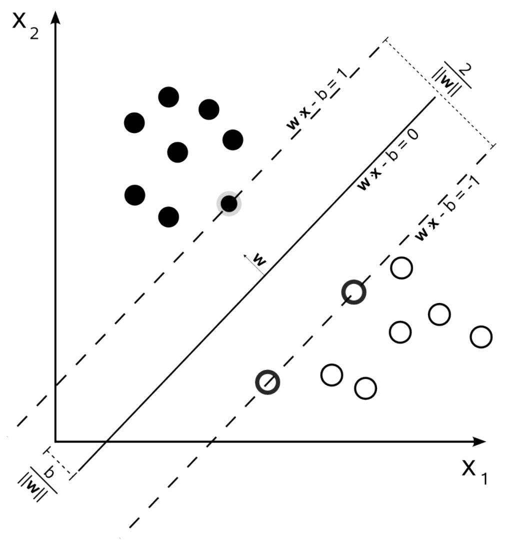


Figure 2.5: A depiction of the optimization problem solved by a support vector machine classification algorithm. The colours of different point correspond to their classes. The learned plan lies equidistant between the nearest points from each class. The figure was released into the public domain for the purpose of illustrating support vector machines, and the creator is known only by a pseudonym "Cyc". It is reproduced here from https://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png

separation might exist. As a simple example, reproduced from Russell and Norvig [1995], consider a problem where points are distributed such that the points of one class form a ring around the points of another class (see Figure 2.6). Obviously there is no way to draw a line (i.e. a 1-dimensional plane) that separates the two classes. However, if a third feature is added corresponding to the distance of each point from the origin, then such a plane is quite easy to draw. Support vector machines can optimize efficiently over points embedded in a variety of feature spaces that are more complex than the original space, because they only use the products of pairs of points in their optimization, which are readily replaced by more complex functions over the pairs that correspond to projection into larger feature spaces (i.e. kernels)[14]. In contrast, other approaches may become much slower when operating over a larger feature space. Common kernels include polynomial combinations of the original features, and the radial basis function or RBF kernel, which computes the distances between pairs of points using Gaussian functions, and which corresponds to projecting points into an "infinite dimensional" feature space in the sense that the distance is described by an infinite series. Support vector machines are used in Chapter 6, but are not described in greater detail there. A reader requiring more details should refer to one or more of the references [Russell and Norvig, 1995; Cortes and Vapnik, 1995].

The final classification algorithm described here is the Naive Bayes algorithm. This algorithm outputs a conditional probability distribution for $P(y|x)$, under the assumption that the values of the features are conditionally independent of each other, given the label y, which is often clearly incorrect. By Bayes Theorem, the probability of y given x is proportionate to

$$P(y|x) \propto \prod_{x_j \in x} P(x_j|y)$$

This leads to the "Naive" name, because the model cannot properly encode the effect of interactions between features on the label. For example, suppose that if a student spends a lot of time reading the textbook, and views the lecture, the probability of them doing well is much higher than if they do either of these things alone. Since Naive Bayes assumes

$$P(y = \text{pass}|\text{read book, saw lecture}) = P(y = \text{pass}|\text{read book})P(y = \text{pass}|\text{saw lecture})$$

the model cannot encode this kind of non-linear relationship. Despite this, the Naive Bayes algorithm has many advantages. Models can be produced after a single pass through the data, whereas the optimization process for logistic regression and support vector machine

[14]The precise runtimes may vary, but will take between $O(kl)$ and $O(knl)$, where k is the number of iterations (typically small), l is the number of exemplars, and n is the number of features [Chang and Lin, 2011].

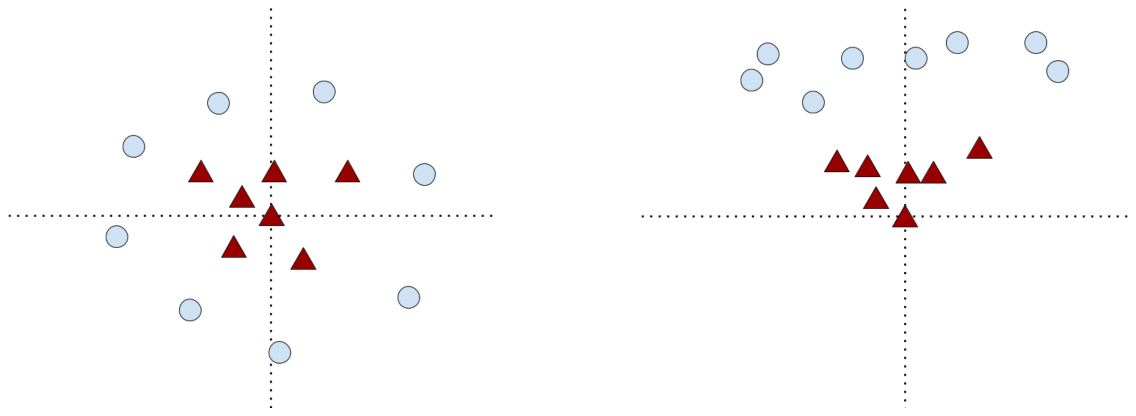


Figure 2.6: An example showing the power of the kernel trick. In the original feature space the points belonging to the two classes cannot be separated with a plane (left). After projecting them into a space with an additional dimension based on the distance of points from the origin, the points are easily separated (right). Note that the figure on the right shows only two of the three dimensions involved (the horizontal axis from the left figure, and the new feature based on distance from the origin).

learning may (and often does) require many such passes (depending on the number of iterations needed to solve the optimization problems). The model is also easy to understand. $P(y = \text{pass}|\text{saw lecture})$ indicates an estimate of the probability that a student passes the course, given that they saw the lecture, for example. If this value is higher, then the instructor might reasonably infer that students who do not attend the lecture should be encouraged to seek additional help, or to study more vigorously. Estimates of the probabilities used in a Naive Bayes model are typically obtained by counting observations of the data. For example, if 4 students are observed who saw the lecture and failed, and 5 students are observed who saw the lecture and passed, a typical Naive Bayes learner would output a value of $P(y = \text{pass}|\text{saw lecture}) = \frac{5}{9}$, or perhaps $\frac{6}{11}$ if common smoothing techniques (see [Russell and Norvig, 1995]) have been used. This makes it very easy to extend Naive Bayes to solve classification problems with more than two classes. The resulting model simply includes estimates for the probabilities of each class. Naive Bayes is not used directly in the thesis, but a Bayesian algorithm is used in Chapter 7, which is similar in principle, but makes less extreme assumptions about the probability distributions it is modelling.

2.2.3 Multiclass Classification

Although several of the classification algorithms described earlier in this chapter were not intended to solve classification problems with more than two classes, most of the classification tasks considered in this thesis involve many classes, often between 7 and 20, because the classes correspond to the set of candidates in an election. A classification algorithm intended for use with binary classification problems can be converted into an algorithm that can address problems involving more than two classes, and there are a number of different strategies for doing so. Perhaps the most common approach, and the one adopted by this thesis in most cases, is "one-versus-all" or OVA classification [Rifkin and Klautau, 2004]. In OVA classification, a classification problem with k different classes (i.e. unique labels) is converted into k separate binary classification problems. In the i^{th} such problem, members of class i are re-labelled to have label 1, and all other classes are re-labelled with value 0. A binary classification algorithm is then used to solve each sub-problem in turn, yielding k separate models, each of which has learned to differentiate one class from the rest. To make a prediction about the class of a new, previously unseen, exemplar, the exemplar is provided to each model, which will output a label (1 if the model predicts that the new exemplar belongs to its target class, 0 otherwise). If only one classifier (i.e. model) predicts a value of 1, then the new exemplar is readily labelled with the class associated with that model. Otherwise, various other techniques can be used to pick a label. Models like the support vector machine and logistic regression learners are capable of outputting a "confidence" value between 0 and 1, rather than just a binary label. A common approach is to pick the class corresponding to the classifier with the highest confidence.

Although Rifkin and Klautau [2004] argues compellingly for the use of OVA classification over competing methods, these other methods will be mentioned briefly for completeness. The natural alternative to OVA is "one-versus-one" classification. In this scheme, a separate model is trained for every pair of classes, and various schemes are used to combine the outputs of these models. For example, Knerr et al. [1990] suggests that outputs be combined by applying a logical AND to the output of every model that involved a given label. If all such models agree that the exemplar belongs to a given class, it is then assigned to that class. Alternatively, Freund and Schapire [1997] suggests a similar scheme where the exemplar is assigned to the label with the largest fraction of classifiers agreeing with the assignment. More complex schemes also exist (e.g. [Platt et al., 1999; Dietterich and Bakiri, 1995]). A one-versus-one approach is used in the experiments near the end of Chapter 6, but nowhere else in this thesis.

2.2.4 Feature Selection

While techniques like the support vector machine classification algorithm are able to operate efficiently over very large feature spaces, other algorithms are not. Feature selection techniques can be applied to reduce the size of a large feature space to a smaller one that contains only the most predictive features. As with classification itself, there are many different feature selection techniques. This thesis makes use of only two techniques.

Principal components analysis (PCA) [Dunteman, 1989] finds linear combinations of the original features that effectively encode (most of) the same information in a lower dimensional space. It is widely applied as both an analysis and compression technique in addition to its application in feature selection. PCA is used only briefly, near the end of Chapter 6.

The other feature selection technique used in this thesis is Information Gain (IG)-based feature selection [Quinlan, 1986; Yang and Pedersen, 1997]. This approach ranks features by the decrease in the entropy of the labels that would result from partitioning the exemplars about different values of the feature. For example, suppose that all students who saw a particular lecture passed the course, and all who did not see it failed. If the students are partitioned based on who did or did not see this particular lecture, then the resulting subsets will have no entropy at all (their labels will be homogeneous). In contrast, the full population of students might have quite a lot of entropy (their labels might be very heterogeneous). The reduction in entropy determines the importance of the feature corresponding to attendance at the lecture in question. IG-based feature selection is described more formally near the end of Chapter 6, but is used throughout the thesis.

2.3 Summary

This chapter provided a brief survey of topics relevant to the thesis, to ensure the reader has a basic understanding of the background material. In subsequent chapters, these topics will be expanded upon in greater detail as needed. The reader may be referred back to this chapter for reference however.

Chapter 3

Problem Statement

There are known knowns. These are things we know that we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns – the ones we don't know we don't know.

Donald Rumsfeld [2002]

What he forgot to add was the crucial fourth term: the "unknown knowns". These are the things that we don't know that we know.

Slavoj Žižek

[2004]

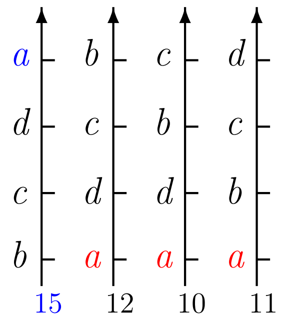
While Chapter 1 provided an overview of the thesis, and some examples outlining the problems to be addressed, the core problem of the thesis has been described only informally, through examples and prose. Taking as given that the reader is now familiar with the background material discussed in Chapter 2, the problem can now be presented in greater detail, and expressed formally. This chapter mixes formal problem descriptions with concrete examples, and concludes with a discussion of current approaches. A more detailed motivational example is used throughout, and described early in the chapter. Subsequently, Chapter 4 proposes a new approach to solving the problem.

3.1 Social Choice with Partial Preferences

In Chapter 1, the core problem addressed by this Thesis was "**Can existing techniques from machine learning be used to better make group decisions in multiagent systems?**". To answer this question concretely, it must first be carefully framed. How precisely might machine learning techniques be able to improve group decision making? What part (or parts) of group decision making could benefit most from their application? A good candidate area is the problem of making group decisions with limited information.

As discussed in Chapter 2, voting is a process by which a group of autonomous agents can coordinate their actions. Ranked ballot voting systems like the **Borda Count** or **Single Transferable Vote** function[1] by eliciting preferences from each voter, and aggregating these preferences to produce a collective outcome. The availability of complete preference information allows these systems to avoid making "poor" or "unfair" group decisions according to various more precise definitions of those terms. For example, **Single Transferable Vote** will never pick a Condorcet loser as the winner of an election (i.e. a candidate that a strict majority of voters believes is the worst possible), while systems like **Plurality** can easily pick such candidates[2].

Although ranked ballot voting systems are powerful, and "fairer" according to many criteria (e.g. never picking a Condorcet loser), much of this power comes from the extra information that ranked ballot systems have access to. In the example preference profile from Chapter 1 reproduced below, a simple rule like **Plurality** picks Condorcet loser a as the winner (since a is the most preferred candidate of 15 voters) because it does not know (or request) information about the candidate each voter liked least. In contrast, any rule that does request such information can (at least in principle) avoid picking a.



[1] A ranked ballot system which simulates a number of rounds of **Plurality**-like contests, dropping the weakest candidate in each round, and reassigning their supporters to other candidates.

[2] See the example in Chapter 1.

Since the power of ranked ballot voting systems comes in part from having access to complete information about voters' preferences, the power of such systems could be significantly reduced if voters cannot or will not provide the necessary preference information. An obvious example is that if voters reported only their top preferences in the above profile, then exactly the same set of problems that plague **Plurality** will affect many other voting systems. For example, if only first preferences are specified, then the principle underlying **Borda Count** becomes nonsensical: the candidate with the highest average position is simply the one with the most first place preferences, exactly the winner **Plurality** would select.

There are many ways in which the available information could be reduced. In a robotic swarm, some agents may have unreliable sensor information, or could experience limitations on their communication with the entity responsible for aggregating their preferences. In a group recommendation system like Netflix, users may not have enough information about the various alternatives to rank them. In a hiring committee, human agents might decline to publicly announce their preferences, to avoid offending their peers. Similarly, if preference information is a valuable secret, as might be the case when aggregating the preferences of corporate entities or strategically motivated software agents, voters may attempt to reveal as little information to the system as possible. Consequently, in many practical applications of ranked ballot voting systems, the preferences provided by voters will be incomplete.

3.1.1 Motivating Example

Although it may be apparent that extreme information deprivation (such as limiting agents to reporting only their first preferences) may introduce challenges, it is perhaps less obvious that the removal of smaller amounts of information can have a significant impact on the outcome of an election. To demonstrate this, consider a whimsical scenario taking place on the surface of Mars. Robotic swarms, comprised of many small, weak machines, are retrieving mineral samples on behalf of different prospectors. Four small swarms meet, but each is too small to retrieve an adequate sample on its own, and the groups must work together to prospect four potential mining sites, sharing the mineral rights between their owners.

The largest swarm, belonging to Space Mining Inc., acts on behalf of a prospector interested primarily in distant and dangerous location a, while MarsMetal Corp. prefers the closer but less geologically promising sites like b and c. Ares and Boone Prospectors operate smaller swarms, and also prefer to minimize risk. The three risk-averse companies each prefer the safe sites closest to their headquarters to sites further away (See Figure 3.1).

To facilitate prospecting, the companies have agreed on a coordination protocol to allow smaller swarms to cooperate. Each member of the swarm will submit a preference (ranked ballot) summarizing the prospecting preferences of its company to a voting system. The swarms collectively agree to prospect the site selected by the voting system. The prospectors agree to use the Borda Count system, under the belief that it will select compromise missions for their swarms, rather than recklessly risking robots on the hunch of another company.

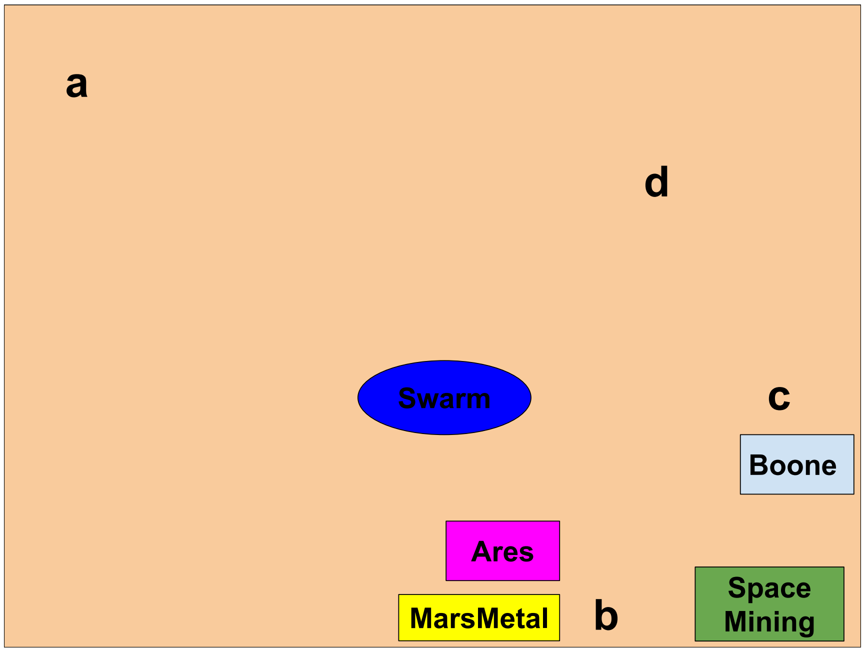


Figure 3.1: A map depicting the agents and alternatives involved in Martian Swarm Example.

Suppose that the preference profile for the various members of the swarm are as given in Figure 3.2, with the leftmost preferences belonging to the 20 robots of Space Mining Inc., the 5 second from the left belonging to the robots of MarsMetal, the 10 third from the left to Ares, and the remaining 4 to Boone Prospectors. Location a is not a Condorcet Loser in this profile, but nontheless, a is a divisive choice because of the high risks involved. If the **Borda Count** system is applied with a scoring vector of $\vec{s} = \{3, 2, 1, 0\}$, then alternative c is selected as the winner, as summarized in Table 3.1. This is not unreasonable, as c does appear in the highest average position across the preferences of the companies, weighted by the resources invested in this particular swarm.

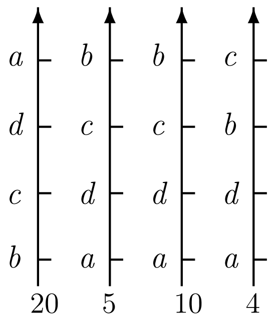


Figure 3.2: A preference profile from the Martian Swarm example, used to illustrate the difficulties of group decision making with incomplete information.

a	$3 \cdot 20 = 60$
b	$3 \cdot 15 + 2 \cdot 4 = 53$
c	$1 \cdot 20 + 2 \cdot 15 + 3 \cdot 4 = \mathbf{62}$
d	$2 \cdot 20 + 1 \cdot 19 = 59$

Table 3.1: The Borda Count scores assigned to each alternative in the Martian Swarm Example.

Suppose however, that the situation were different, and that the prospectors at Ares are a new entry to the market. So far, the company only has good knowledge of location b, and is unsure about the relative risks involved in exploring the other locations. They program their robots to communicate this information to the voting system, trusting that the robots will be used in a safe and reasonable fashion based on the information they are able to provide. The resulting preference profile appears in Figure 3.3.

How should the voting system treat the missing information transmitted by Ares' robots? As a natural first step, the system might assign 0 points to each unranked candidate on each of the 10 ballots submitted by Ares' robots, effectively using the scoring vector $s = \{3, 0, 0, 0\}$ for the incomplete ballots. This amounts to using only the information that was explicitly given: b is the most preferred alternative of the Ares company. The Borda scores resulting from this decision are shown in Table 3.2. Several things have changed. Now a has the highest total score. Further, while previously the overall ranking of the alternatives was $c \succ a \succ d \succ b$, when the missing information is treated this way, the Borda Count instead ranks the alternatives $a \succ b \succ d \succ c$. b has become the second highest ranked alternative, despite being ranked last when all preferences were known. c, the winner when all preferences were known, is now ranked last.

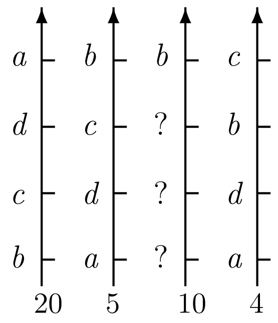


Figure 3.3: An *incomplete* preference profile from the Martian Swarm example, after the preferences of Ares are ablated.

a	$3 \cdot 20 = \mathbf{60}$
b	$3 \cdot 15 + 2 \cdot 4 = 53$
c	$1 \cdot 20 + 2 \cdot 5 + 3 \cdot 4 = 42$
d	$2 \cdot 20 + 1 \cdot 9 = 49$

Table 3.2: The Borda Count scores assigned to each alternative in the Martian Swarm Example, if Ares transmits incomplete information, and only transmitted information is used in scoring.

This shift in the outcome of the vote is not merely the result of preferences being unspecified by Ares, but also of the decision that the voting system made about how to *treat* this missing information. Suppose that instead of assigning no points, the system assigns all three unranked candidates the average of the points they could have received under a randomly selected ranking. On each of Ares' ballots, one candidate would have gotten two points, another would have received one point and the third no points, if information on the three unranked candidates were included. This suggests each of the three candidates should be assigned $\frac{2+1+0}{3} = 1$ point per Ares ballot, effectively using the scoring vector $s = \{3, 1, 1, 1\}$. Table 3.3 shows the Borda scores obtained if this process is followed instead. Again, a is the winning alternative, but now the ordering is $a \succ d \succ b \succ c$, inverting the overall ranks of d and b.

a	$3 \cdot 20 + 1 \cdot 10 = \mathbf{70}$
b	$3 \cdot 15 + 2 \cdot 4 = 53$
c	$1 \cdot 30 + 2 \cdot 5 + 3 \cdot 4 = 52$
d	$2 \cdot 20 + 1 \cdot 19 = 59$

Table 3.3: The Borda Count scores assigned to each alternative in the Martian Swarm Example, if Ares transmits incomplete information, and an average score is assigned to missing entries.

3.2 Formal Problem Statement

If the treatment of missing information can affect the outcome selected, then this element of a voting system's design must be carefully considered. In formally describing this aspect of decision making, there are both modelling and evaluation decisions that determine the exact nature of how missing information ought to be treated. In this section, a formal model of group decision making under incomplete information is specified, with justification for the modelling decisions following.

An election with incomplete information is modelled by a set of agents $A = \{a_1, ..., a_n\}$ of total size n, deciding among a set of outcomes $O = \{o_1, ..., o_m\}$. Each voter has a *true* preference over O, denoted by $\succ_i \in \succ$, and an *expressed* preference (ballot) $b_i \in B$, over the set of candidates $C = \{c_1, ..., c_k\}$. Additionally, if $x \; b_i \; y$, then $x \succ_i y$ for all $x, y \in C$ and $a_i \in A$ (i.e. voters report their preferences honestly).

A preference is a binary relation over C, that is reflexive, transitive and antisymmetric, but not necessarily total (i.e. a partial order). The set of all permissible preference relations over a set C is denoted by $\mathcal{P}(C)$. $x \succ_i y$ is written to indicate that x precedes y in a true preference order $\succ_i$. $x \sim_i y \leftrightarrow x \not\succ_i y \wedge y \not\succ_i x$ is written to indicate that the order $\succ_i$ does not define a relationship between x and y. Analogously, for expressed preferences $x \; b_i \; y$ is written to indicate that x precedes y, and $x \sim_i^b y$ to indicate that no relation between x and y was expressed. This notation is summarized in Table 3.4 for reference.

This model contains a non-standard component. It defines explicitly separate *true* and *expressed* preferences for voters. In social choice it is typically assumed [Black et al., 1958] that voters express exactly their true preferences. A notable exception to this is when considering strategic voting behaviours like 'manipulation', where one of the voters expresses preferences that differ from their true preferences [Gibbard, 1973; Satterthwaite, 1975]. This formalism will allow clear expression of different models of incomplete *expressed* preferences. The assumption that voters do not cast strategic ballots is preserved via the restriction that a voter's *true* preferences must be a consistent extension of their *expressed* preferences[3]. That is, if $x\ b_i\ y$, then $x \succ_i y$ for all $x, y \in C$ and $a_i \in A$. This does not model the case where voters are *misinformed* (i.e. where their expressed preferences explicitly contradict their true preferences), only the cases where voters are *uninformed* (i.e. where their expressed preferences contain a subset of the information encoded in their true preferences) Some of the proposed voting systems for this model of election, discussed later in the thesis, are able to function even when voters are misinformed, however.

The reason that a separate outcome set is used is to allow the model to capture both elections to pick a single winner, and elections to rank the entire set of candidates. In the latter case, $O = \mathcal{L}(C)$, where $\mathcal{L}(C)$ is the set of all linear orders over C.

An election is *decided* by a social choice function[4]. A social choice function $S : \mathcal{P}(C)^n \rightarrow O$ maps from a preference profile (i.e. a set of preferences for every member of a group or community) to an outcome. Throughout, only social choice functions that operate identically for different values of n are considered, and that break ties by drawing lots (i.e. uniformly at random).

3.2.1 Formalization of the Motivating Example

Formalizing the motivating example from the previous section will provide a concrete instantiation of the abstract model described above. In this formal model, there exists a set of agents

$$A = \{a_1, ..., a_{39}\}$$

where a_1 to a_{20} correspond to the robots of Space Mining Inc., a_{21} to a_{25} correspond to those of MarsMetal Corp, a_{26} to a_{35} to those of Ares, and a_{36} to a_{39} to those of Boone Prospectors. The set of alternatives is given by

$$C = \{a, b, c, d\}$$

and is identical to the set of possible outcomes $O = C$ if the robots will visit only one of the locations. If instead the robots will visit more than one location, and the vote is to determine the order they will be visited, then $O = \mathcal{L}(C)$ instead.

The robot's true preferences are summarized in Table 3.5, while their revealed preferences are summarized in Table 3.6.

[3]i.e. Voters may lie by omission, but do not contradict their true preferences.

[4]The term is used in this model to refer to both social choice and social welfare functions.

A	A set of agents
a_i	An element of the set of agents
n	Total number of agents; $\|A\|$
O	A set of outcomes
o_i	An element of the set of outcomes
m	The total number of outcomes
C	The set of alternatives
c_i	An element of the set of alternatives
k	The total number of alternatives
$\succ_i$	The *true* preferences of agent i; A partial ordering over C
b_i	The *expressed* preferences of agent i; A partial ordering over C
$\succ$	The set of true preferences
B	The set of expressed preferences
$\mathcal{L}(C)$	The set of linear orders over C
$\mathcal{P}(C)$	The set of partial orders over C
S	A social choice function

Table 3.4: Notation used in the modelling of an election.

$\succ_1$ to $\succ_{20}$	$a \succ d \succ c \succ b$
$\succ_{21}$ to $\succ_{25}$	$b \succ c \succ d \succ a$
$\succ_{26}$ to $\succ_{35}$	$b \succ c \succ d \succ a$
$\succ_{36}$ to $\succ_{39}$	$c \succ b \succ d \succ a$

Table 3.5: A formalized version of the true preferences in the motivational example.

b_1 to b_{20}	$a \succ d \succ c \succ b$
b_{21} to b_{25}	$b \succ c \succ d \succ a$
b_{26} to b_{35}	$b \succ c \sim d \sim a$
b_{36} to b_{39}	$c \succ b \succ d \succ a$

Table 3.6: A formalized version of the revealed preferences in the motivational example.

The Borda Count maps from the expressed preferences of the voters (i.e. $B = \{b_1, ..., b_{39}\}$) to either C or $\mathcal{L}(C)$, again depending on whether the robots are interesting in selecting only one location to visit, or in ordering the entire set of locations, to visit them one after another. If $O = C$, then the **Borda** voting system with scoring vector $s = \{3, 2, 1, 0\}$ is defined by:

$$\text{BordaScore}(B, c) = \sum_{b_i \in B} s(\text{Pos}(c, b_i))$$

$$\text{Borda}(B, C) = \operatorname*{argmax}_{c \in C} \text{BordaScore}(B, c)$$

where $\text{Pos}(c_i, b_i)$ returns the position of c on ballot b_i (i.e. the number of candidates c' such that $c' \succ c$ in b_i, or $|C| - 1$ if c is not defined on b_i) and $s(i)$ returns the value of the scoring vector at position i. Otherwise, if $O = \mathcal{L}(C)$, then the Borda Count is instead given by:

$$\text{BordaRank}(B, C) = \succ_l \in \mathcal{L}(C) s.t. c \succ_l c' \to \text{BordaScore}(B, c) > \text{BordaScore}(B, c')$$

The two **Borda** definitions above are consistent with the example voting systems that ignored missing information entirely, and did not assign any of the missing points. If a definition that assigned average points were desired, a different definition of the Pos function could be provided:

$$\text{Pos}(c, b_i) = \sum_{c' \in C \setminus c} I(c' b_i c) + \frac{I(c' \sim_i^b c)}{2}$$

where $I(x)$ is an indicator variable, with value 1 iff x is true.

3.2.2 The Definition of a "Good" Decision

The motivating example in the preceding sections illustrates that the way in which missing information is treated can have a significant impact on the decision reached by a voting system. *Different* decisions were reached when different extensions of the **Borda** system were used. If the purpose of a good voting system is to make decisions that reflect the will of the electorate, then some of these decisions must be better (i.e. more reflective) than others. This section provides further justification for using separate true and revealed preferences, and philosophical motivation for performance measurements used to assess the quality of decisions made in subsequent sections. The next section contains a discussion of existing systems, and why they may make poor decisions, as assessed by these measures.

Philosophically, the approach to voting proposed in this thesis is an outgrowth or extension of an older idea within political science: *Voting Correctly.* Classical models of democratic behaviour from the enlightenment era (e.g. [Mill, 1999; Rousseau, 1920; Hobbes, 1900]) assume the existence of a rational electorate, where individual voters are civic-minded, politically engaged, and able to determine the policies that benefit them maximally. If the agents are rational, and the system sound (e.g. if the system picked a Condorcet winner), then the selected choice must be truly reflective of the will of the people. In practice however, most voters are not very politically engaged. Indeed, attempts to measure the level of knowledge and engagement of human voters on political issues reveal a great disparity between the theorized model of human behaviour, and the reality of human disinterest in political matters. Lau and Redlawsk [Lau and Redlawsk, 1997] describe the state of research in the 1997:

> Five decades of behavioural research in political science have left no doubt, however, that only a tiny minority of the citizens in any democracy actually live up to these ideals [i.e. of rational, fully informed voters]. Interest in politics is generally weak, discussion is rare, political knowledge on the average is pitifully low, and few people actually participate in politics beyond voting (e.g [Berelson et al., 1954; Campbell et al., 1966; Converse, 1964; Carpini and Keeter, 1997]). And what good is even voting if for so many it is based on so little information?

Within political science, Lau and Redlawsk describe a division in how the gap between theory and practice ought to be interpreted, with some experts characterizing democratic decisions as having false legitimacy (because the reality theories of legitimacy are predicated on does not exist), and others revising the theories to better match the data. Lau and Redlawsk's *Voting Correctly* is a new approach to addressing the gap, that instead attempts (in a sense) to revise the data to better match the theory. The authors claim that, in fact, despite being so ill informed, a large majority of voters are casting "correct" ballots, providing reasonable legitimacy to democratic decisions.

For the purpose of this thesis, it is the notion of "correctness" used by proponents of Voting Correctly that matters most. There are many reasonable ways to define correctness. A technocratic approach to correctness would assume that a correct vote reflects some objective relationship between the voter's beliefs, social class, aspirations, or values, and some similar set of measurements about the candidates. In contrast, an purely egalitarian approach to correctness might assume that a vote is *always* correct, as it reflects the intention of the voter, no matter how misinformed or misplaced. Lau and Redlawsk suggest instead a *normative* definition of correctness: a correct vote is one that an individual voter considers correct ex-post, when they are provided with full information in an easily digestible format. Stated more concisely, a correct vote is the one a voter *would have cast with complete information*, as judged by the voter themselves.

Lau and Redlawsk use this notion of correctness to empirically assess the extent to which voters cast correct ballots on political issues [Lau and Redlawsk, 1997]. Voters were provided with a simulated election, and noisy or confusing sources of information about the candidates. After gathering information from these noisy sources, voters cast ballots under the **Plurality** system. They were then given a "cheat sheet", summarizing the positions of the candidates on various issues in a concise, easily comparable, format. After considering the summary, voters were asked to assess whether they had voted correctly (i.e. whether they would change their vote). Approximately 30% of voters admit to making the wrong choice, a figure since replicated on a much larger array of electoral groups [Lau et al., 2014]. This provides compelling evidence for the idea that voters may express preferences for candidates that differ from their *true* preferences (i.e. the preferences they would express if they had better information).

Following Lau and Redlawsk, the definition of a "good" group decision, in the context of this thesis, is one that is "correct" in the normative sense proposed above. A decision made with voters' expressed preferences is a good decision only to the extent that it approximates the decision voters would have made, had they voted with complete information (i.e. the decision voters would have made if their expressed preferences were exactly equal to their true preferences). Formally, this idea can be expressed in conjunction with different distance pseudometrics applied to the space of outcomes (which can be either individual winners, or orderings over the set of alternatives). Suppose that a function $\delta : O \times O \rightarrow \mathbb{R}$ defines distances between elements of the outcome set of an election, with the usual properties of a pseudometric [Burago et al., 2001]:

- $\delta(o_i, o_j) > 0$
- $\delta(o_i, o_i) = 0$
- $\delta(o_i, o_j) = \delta(o_j, o_i)$
- $\delta(o_i, o_j) \leq \delta(o_i, o_k) + \delta(o_k, o_j)$

Then, in the context of this thesis, a good decision minimizes $\delta(S(B), S(\succ))$ (i.e. the distance between the outcome under voters' expressed preferences, and the outcome under their true preferences).

Example: Continuing with the motivational example from earlier in the chapter, if the outcome space was the set of candidates (i.e. the robots only plan to explore one location; $O = C$), then the distance over the set of points in the outcome space could be a function of ordering of the candidates under the scores assigned to different outcomes by the BordaScore function, *under the voters' true preferences*:

$$\delta_{\succ}(o_i, o_j) = |\text{BordaScore}(\succ, o_i) - \text{BordaScore}(\succ, o_j)|$$

The quality of a decision made using ballots B would then be given by:

$$\delta_{\succ}(\text{Borda}(B), \text{Borda}(\succ))$$

The totals from the earlier example show that if Borda is used, a is selected as the winner using B, but c as the winner using $\succ$. c beat a by two points in the election with complete information (see Table 3.1), so the decision is characterized as being two points from optimal. Alternatively, if a variant of Borda that picked b as the winner were used, it would be 9 points from optimal.

Alternatively, if the robots intended to visit all the locations, and are voting to determine the order in which they should be visited (i.e. $O = \mathcal{L}(C)$), then instead a pseudometric defined over linear orders of candidates is used. For example, one might elect to use Kendall's Tau correlation, which returns a real value between -1 and 1 indicating the similarity of two ordinal sequences, by measuring the number of adjacent pairwise swaps needed to convert one sequence into the other. In this case, the pseudometric is defined as a transform of the correlation to ensure a positive distance value:

$$\delta(o_i, o_j) = -(\tau(o_i, o_j) - 1)$$

Recall that the outcome when all voters expressed preferences that were identical to their true preferences was $c \succ a \succ d \succ b$ (see Table 3.1), and, when the expressed preferences of Ares were reduced to just reporting b as a first preference, the outcome was $a \succ b \succ d \succ c$ (see Table 3.2). Four swaps are needed to convert the latter sequence to the former ($b \leftrightarrow d$; $c \leftrightarrow b$; $c \leftrightarrow d$; $c \leftrightarrow a$). There are a total of $\frac{4 \cdot 3}{2} = 6$ possible inversions of the candidates, so the Kendall Correlation is $\tau = \frac{2-4}{6} = -\frac{1}{3}$. Consequently,

$$\delta(\text{BordaRank}(B), \text{BordaRank}(\succ)) = -(-\frac{1}{3} - 1) = \frac{4}{3}$$

In contrast, if the Borda extension that assigns average scores to missing candidates is used, then the order $a \succ d \succ b \succ c$ is selected (see Table 3.3). Converting this order to the one reference order produced by the voters' true preferences requires just 3 swaps ($c \leftrightarrow b$; $c \leftrightarrow d$; $c \leftrightarrow a$), so it has a Kendall Correlation of 0. It follows that, under this extension:

$$\delta(\text{BordaRank}(B), \text{BordaRank}(\succ)) = -(0 - 1) = 1$$

3.2.3 Problem Statement

Informally, the topic addressed by this thesis was **Can existing techniques from machine learning be used to make better group decisions in multiagent systems?** The content of the preceding two subsections provide enough information to make this statement more formally.

The topic addressed by this thesis is the application of existing techniques from machine learning to make group decisions that are close (in the sense of a problem-specific pseudometric) to the decisions voters *would have made* using a voting system, if their true preferences matched their expressed preferences. This is inspired by the philosophy of Voting Correctly, and could be interpreted as constructing systems to help groups vote more correctly, using the normative definition of correctness discussed in the previous section.

3.3 Existing Approaches

In the first section of this chapter, a motivating example showed why members of a multiagent system might need to vote with a ranked-ballot system; why some members might be unable to cast a complete ballot; and that decisions made when some ballots are incomplete can be quite different from the decisions voters might prefer. The chapter's second section then presented a formal model of an election with incomplete information, and an approach to measuring the quality of decisions made with incomplete information. In this section, two existing approaches to voting with incomplete information are discussed, along with examples of situations where existing approaches make decisions that could be improved.

The existing approaches considered here were constructed with different philosophies, and should not be expected to perform well when assessed according to the measures proposed in this thesis. This in no way invalidates the existing approaches, which perform well with respect to the goals of their designers. Instead, it highlights a different problem area within group decision making, and shows the need for new techniques to address these problems.

3.3.1 Maximum Likelihood

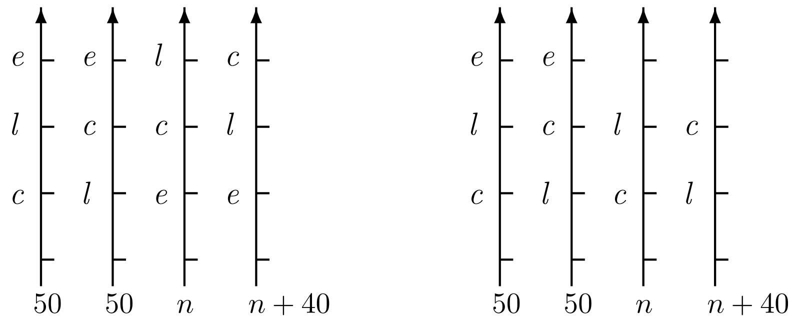
If a voting system has no external information about the true preferences of voters, and the expressed preferences are incomplete, then a natural approach to aggregating the given information is to assume that all completions of the ballots are equally likely. Philosophically, such a system is fair in the sense of being unbiased (i.e. it treats all candidates as equally likely to appear ahead of each other).

Xia and Conitzer [Xia and Conitzer, 2011] propose such a system, based on the idea that votes are independent observations of a *globally* correct ranking. Voters' observations are modelled as partial orders derived from the globally correct ranking via a stochastic process. The set of stochastic processes considered assumes that, $P(c_i \succ c_j | o^*)$ is independent for every pair c_i, c_j (except with respect to the requirement for a transitive ordering overall), where o^* is the globally correct outcome.

The technique's performance is validated theoretically, in terms of consistency with different axiomatic properties, and computational complexity of inference in different models.

The technique is interesting in that it provides many general extensions of existing voting systems to domains with partial preferences. However, the voting rules proposed ignore missing data entirely when calculating the likelihood that $c_i \succ c_j$ in the global ranking. As an example, depicted in Figure 3.4, consider the proposed extension based on the weighted majority graph. In this graph vertices are candidates, and directed edges are weighted according to the margin by which voters that ranked two candidates prefer the source to the destination. Consider an election between two mainstream alternatives (l and c) and one lesser-known extremist party (e). Suppose that the extremist party has 100 dedicated supporters who rank $e \succ l \sim c$. l and c each have $n \pm 40$ ill-informed supporters who express only $l \succ c$, or $c \succ l$, and express no information about e, save perhaps some small minority (say 20 each).

Consider the pairwise contest between e and l. Since 100 voters rank $e \succ l$, and only 20 rank $l \succ e$, the assumption that the remaining votes are distributed uniformly at random means that e prevails over l by a margin of 80. By similar argument, e prevails over c by



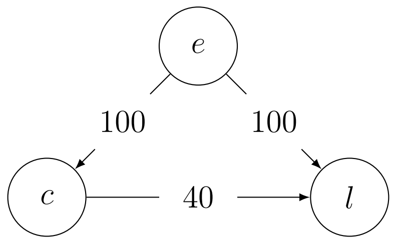


Figure 3.4: An example where the maximum likelihood approach to voting selects a Condorcet loser as the winner of the election. The upper portion of the figure shows the true and expressed preferences of the voters, while the lower portion show the weighted majority graph for the election.

a margin of 80. Since c and l have a similar number of total supporters, one will prevail over the other, but by a margin of at most 80. e must be declared the winner, since it dominates both other candidates by the widest observed margins. However, by making the assumption of pairwise independence, the system has ignored the observed pattern that everyone ranking one of c or l first ranks e last. If the true preferences of the electorate at large rank e last en mass, then a very poor decision has been made (the worst possible: a Condorcet loser).

The above example explicitly violates the independence assumption of Xia and Conitzer's model, so it is not surprising that a poor decision was made. However, it does illustrate the need for a new approach when pairwise independence does not hold.

3.3.2 Minimax Regret

Lu and Boutilier [Lu and Boutilier, 2011b] adopt a similar problem model to the one outlined in this chapter, implicitly assuming different revealed and true preferences. However, rather than trying to pick the most correct alternative, their approach focuses on picking the one with the least potential to be the worst alternative, given the information voters provided. The technique was designed to be used to incrementally and strategically elicit further information from voters, improving the lower bound on decision quality by the maximum amount possible with each elicitation.

The proposed approach is called "minimax regret" (MMR). The regret associated with a decision is the distance between the outcome selected and the outcome selected in reality, according to some distance pseudometric (e.g. the distance between two candidate's Borda scores). This is much the same as in the model proposed in this chapter. For each candidate c, MMR computes the adversarial preference profile that is both consistent with the expressed preference profile, and where c is the greatest distance possible from the winner. MMR declares the winner to be the candidate which is closest to winning under its own adversarial preference profile. More formally, given a social choice function S, distance metric d, and expressed preferences B:

$$\text{MMR}(B, C) = \operatorname*{argmin}_{c \in C} \operatorname*{argmax}_{\pi \in \Pi} \delta_\pi(S(\pi), c)$$

where Π is the set of all ballots that are extensions of B, and δ_π is a distance function in terms of some $\pi \in \Pi$, not $\succ$.

Lu and Boutilier [Lu and Boutilier, 2011b] show both that the technique is computationally tractable for some voting rules (e.g. Borda), but not others (e.g. Copeland) where the set of possible profiles must be enumerated in the worst case, and is exponentially large in the number of candidates and agents. They also evaluate MMR empirically by showing the change in the regret for the selected candidate decreases rapidly as additional information is elicited, during simulated elections on the Dublin North dataset [Mattei and Walsh, 2013].

Although MMR's conservative nature provides a clear prioritization for eliciting new information from voters, it can lead the technique to make choices that are cautious, but improbable. For example, consider the motivational example used throughout this chapter. If the ballots belonging to Ares in Figure 3.3 were completed $b \succ c \succ d \succ a$, then a would be defeated by just 2 points. The completion $b \succ d \succ c \succ a$ would allow d to defeat a by 9 points. Therefore, the maximum regret for picking candidate a is 9 points. However, if they are completed as $b \succ a \succ d \succ c$, then alternative a will beat alternative c by 38 points, so the maximum regret for candidate c is at least 38 points. Therefore candidate a will be the winner picked by MMR by quite a wide margin. Nonetheless, although a is a safer choice, it is incorrect in the example, in the sense that a is in fact a Condorcet loser under voters' true preferences. Further, it should be apparent that a is *likely* to be incorrect by looking at the preferences of similar voters (e.g. those of MarsMetal, which also rank b first).

3.4 Summary

This chapter described a more formal version of the problem addressed in this thesis, and provided a detailed motivational example. A discussion of what constitutes a good decision, including both philosophical justification and a formal definition in terms of pseudometrics, facilitated a more careful description of the goal of the thesis: finding voting systems that can make decisions that are close (under a problem-specific pseudometric) to those that would be made with complete information. The final portion of the chapter discussed two existing approaches to similar problems, explained the different modelling assumptions made, and provided short examples of problems where violations of the existing approaches' assumptions led to bad (i.e. far from optimal) decisions. The next chapter proposes a new approach to solve these problems based on existing machine learning techniques.

Chapter 4

An Imputation-Based Approach

That the sun will not rise tomorrow is no less intelligible a proposition, and implies no more contradiction, than the affirmation, that it will rise. We should in vain, therefore, attempt to demonstrate its falsehood. Were it demonstratively false, it would imply a contradiction, and could never be distinctly conceived by the mind.

David Hume [1739]

Our passional nature not only lawfully may, but must, decide an option between propositions, whenever it is a genuine option that cannot by its nature be decided on intellectual grounds; for to say under such circumstances, “Do not decide, but leave the question open,” is itself a passional decision—just like deciding yes or not—and is attended with the same risk of losing truth.

William James [1899]

In Chapter 3, a formal problem was posed: how can we design systems to make group decisions with incomplete preferences, such that the decision made is as “close” as possible to the “correct” decision? Closeness was assigned a problem specific meaning, defined in terms of a distance pseudometric tailored to the goals of the agents making the decision. Correctness was assigned the normative definition proposed by Political Scientists working on the problem of "Voting Correctly" [Lau and Redlawsk, 1997] . Several existing approaches to making group decisions with incomplete preferences were considered, and some situations in which these systems make decisions that are "far" from "correct", under the definitions adopted by this thesis, were highlighted.

This chapter proposes a new approach to solving social choice problems with incomplete information, based on the application of conventional machine learning techniques. The first section demonstrates how machine learning can be used to solve the motivational example of the previous chapter, and make better decisions in the example problems that were used to illustrate weaker areas for existing systems. In addition, it discusses the inherent weaknesses of a machine learning approach, and justifies the approach in spite of this. In the second section, the approach is presented more formally, and is shown to be a sort of social choice function in its own right, an important result returned to in Chapter 8. The final section of the chapter discusses the various implementation details that need to be considered for a realization of the general framework proposed in the second section. A realization of the approach is implemented and validated subsequently in Chapter 5.

4.1 Solving the motivational example

Chapter 3 used a recurring motivational example to illustrate the difficulty of making good group decisions with incomplete information. Although various techniques were proposed for deciding where the robot swarm should prospect, none of the suggested solutions picked the outcome produced by the voters' true (rather than expressed) preferences. The true and expressed preferences from the example are reproduced in Figure 4.1. In the example, the **Borda Count** voting system was applied to the true preferences of the voters, where it selected the election of candidate c as the winning outcome. However, various reasonable extensions of the Borda system to accommodate partial preferences were shown to pick other candidates as the winner. For example, the extension where missing candidates on a ballot are ignored (assigned no points) selects a as the winner. a was also the candidate selected by the alternative minimax regret approach [Lu and Boutilier, 2011b].

Although these techniques for making decisions with partial information are intended to be fair (i.e. to treat candidates equitably), they are in some sense perhaps too fair. In many domains, like the motivational example, voters' true preferences are not distributed uniformly at random, and not all outcomes are equally likely. Assumptions about the

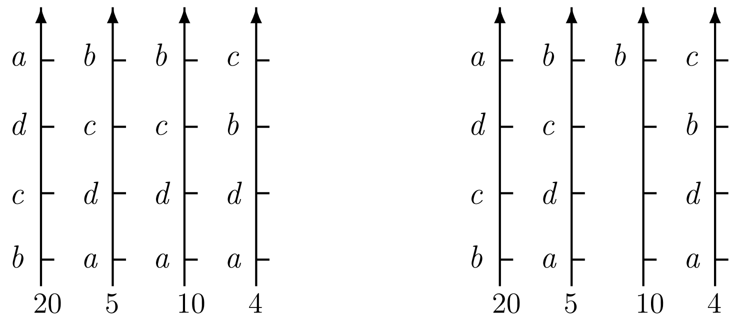


Figure 4.1: A reproduction of true (left) and expressed (right) preferences from the motivational example of Chapter 3.

symmetry of missing information[1] are still assumptions, and in fact, many problem domains do have asymmetric structures that can be exploited. For instance, in the Martian swarm example, voters' preferences were derived from a combination of the locations and priorities of their associated companies. Knowing this, and knowing that location b is nearest to location c, location d is slightly further, and location a furthest, both in terms of geography and risk, one might reasonably infer that voters ranking b first will tend to rank c second, and so on. Indeed, this ranking corresponds directly to the true preferences of the voters.

While a voting system that simply imposed such an ordering on voters' preferences might perform well in this problem domain, it would not adapt well if the environment changes. For instance, what if mineral deposits are confirmed at location a, making further prospecting there less risky than location d, despite the slightly greater distance. Voters' preferences would change, but the system would still assume a was a worse choice than d for risk-averse voters. Instead, to be fair and adaptive, a voting system that exploits structure in voters' preferences should *infer* that structure from the preferences themselves.

As a simple example, suppose voters have submitted top-orders for their preferences, and the missing information is imputed with the most popular suffix[2] of voters' specified preferences (e.g. in Figure 4.1, imputing the 10 ballots with missing information via with the ordering over the unranked candidates c, d, and a that is most popular among the other voters that ranked b first). This technique makes a very strong assumption on the structure of voters' true preferences: those with incomplete information have identical preferences to those who voted most similarly to them. However, compared with the symmetry assump-

[1]As with the maximum likelihood approach to voting [Xia and Conitzer, 2011], for instance.

[2]i.e. impute voters' preferences with the preference of the most common complete ballot that is consistent with the preferences they have specified.

a	$3 \cdot 20 + 1 \cdot 10 = 70$
b	$3 \cdot 15 + 2 \cdot 4 = 53$
c	$1 \cdot 20 + 2 \cdot 5 + 3 \cdot 4 = 42$
d	$2 \cdot 30 + 1 \cdot 19 = 79$

Table 4.1: The Borda Count scores assigned to each alternative in the true preference profile from Figure 4.2.

tions of other approaches (that voters' preferences are drawn uniformly at random [Xia and Conitzer, 2011] or that worst case completions are likely completions [Lu and Boutilier, 2011b]), this does not seem outrageous. Throughout the example this approach will be referred to as **Imputation Plurality**[3].

In Figure 4.1, the incomplete ballots all start with prefix b (they rank b first). There are only 5 other ballots ranking b first in the voters' expressed preferences, and all of them also rank c second, d third, and a fourth. If the expressed preferences are imputed in this fashion, the true preference profile is recovered. Therefore, imputing the expressed profile, and applying the standard **Borda Count** to the completed preferences would yield the same decision as applying **Borda Count** to the true preferences of voters. The combination of a very simple learning algorithm and a standard voting rule yields the correct decision.

The motivating example was constructed to be easily solved with this approach. It is not difficult to construct preference profiles where the proposed technique makes an incorrect decision, and in fact, even arbitrarily incorrect decisions. For example, consider the true and expressed preference profiles in Figure 4.2. Here the true preferences (left) of the 10 voters who only expressed a first preference are radically different from those of the 5 voters who also rank b first. The 10 voters who expressed incomplete preferences rank d second, a third, and c last, while the 5 voters who expressed complete ones use the an entirely different ranking for their true preferences. Table 4.1 shows the **Borda Count** scores for each candidate under this true preference profile. The combination of **Imputation Plurality** followed by **Borda Count** will reproduce the true preferences from the original example (Figure 4.1), under which c is declared the winner. However, c has the lowest Borda score in the true preferences of Figure 4.2, and is therefore the least correct choice under any reasonable distance pseudometric.

The reason for this failure is that the patterns in the true preference profile did not

[3]This is not the same as the **Plurality** social choice function. Rather, it is an imputation approach that is *analogous* to the social choice function of the same name.

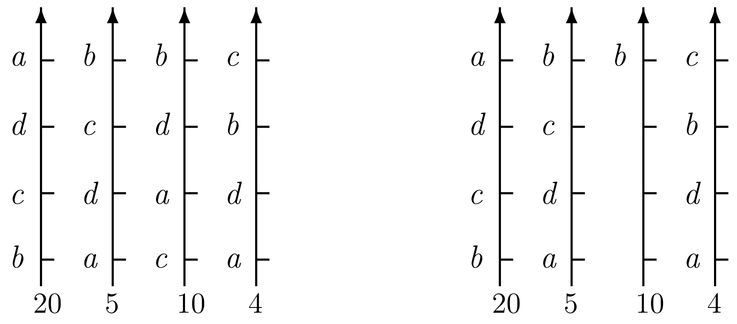


Figure 4.2: Example true (left) and expressed (right) preference profiles where the combination of **Imputation Plurality** and **Borda Count** makes an incorrect decision.

match the assumptions of the (simple) learning algorithm. Voters with similar first preferences did not in fact have similar second, third, or fourth preferences. This is a fundamental limitation of any technique that makes assumptions about the way missing preference information is distributed, but note that it is not inherently better or worse than assuming preferences are distributed at random; it is simply suited to resolving a different set of problems than a technique with with different or fewer assumptions about the nature of the data. This thesis adopts a pragmatic approach to the question of whether assumptions are reasonable or valid: the assumptions made by a decision making system are reasonable and valid insofar as they make better decisions in the real world. If real world electoral problems have relatively homogeneous preferences, then techniques like **Imputation Plurality** would be expected to produce more correct outcomes than if they have highly heterogeneous preferences, for example.

4.1.1 Free Lunches in Social Choice?

The No Free Lunch (NFL) Theorem [Wolpert, 1996] provides a more formal perspective for discussing the validity of different modelling assumptions when constructing machine learning algorithms (as well as algorithms for other AI purposes, like optimization and local search more generally). The NFL Theorem basically concludes that the quality of an algorithm depends entirely on how likely it is to face problems that match its own biases, versus problems that do not [Wolpert, 2012]. Stated more strongly, the essential feature is that, averaged across the set of all possible problems, all supervised learning algorithms [Wolpert, 1996] have identical performance, under any performance measure one might choose. For example, in supervised machine learning, a problem consists of some true function (not fully known, and thus to be learned) mapping inputs to outputs, and some revealed (incomplete) mappings from a subset of inputs to outputs. A learning algorithm generates a hypothesis for the true function on the basis of the revealed mappings. However, the set of all problems includes a set of problems where identical revealed mappings are presented to the algorithm, but *different* true functions are used. Regardless of how the algorithm picks the mapping it will use, it will be better on some problems and worse on others. The NFL theorem says that average performance will actually be *identical* across different algorithms, provided that problems are sampled uniformly at random.

The astute reader may spot a close analogy to the problem considered in this thesis. A "true" mapping from voters (inputs) to preferences (outputs) defines the true preference profile. The revealed preference profile is a partial mapping: for any pair of candidates in $C \times C$ and any voter, a relationship might be defined in the revealed preference profile, or not. A social choice function operating on partial preferences, when averaged over the set of all possible problems, then also ought to perform the same on average. For example, consider again the true and expressed preferences in Figures 4.1 and 4.2. The two different preference profiles are both extensions of the same expressed preference profile. If both true profiles are equally likely to appear, then no social choice method can hope to perform better than any other, on average, when given this set of expressed preferences: A method that picks d as the winner may be as correct as one that picks c as the winner, because there exist true preferences consistent with the expressed preferences under which either d or c wins.

However, different social choice methods make different assumptions about the probability that certain true preference profiles are present, conditioned on the observed expressed preferences. **Imputation Plurality** assumes the true profile is going to be very homogeneous, on average. That is, it assumes that incomplete preferences are just truncated versions of similar preferences. In contrast, **MMR** implicitly assumes that misleading problems, where true preferences are radically different from expressed ones, are common enough to be the focus of its decision making. Clearly on problems that are not so misleading, the imputation-based approaches might be expected to perform better.

4.1.2 The Relationship Between Preference Learning and Group Decision Making

Social choice with incomplete information is highly similar to the related field of preference learning. Preference learning tries to infer an individual's preferences, perhaps to make a product recommendation (e.g. [Zhou et al., 2008; Kamishima, 2003]). In domains like this, performance often focuses on making reasonable recommendations. For example, one might want to infer which movies a customer would like to watch next on the basis of movies they have enjoyed in the past, and perhaps also on the basis of what other viewers like them have enjoyed. In a domain like this one, a reasonable performance measure might include the *fraction* of top results that interest the user [Herlocker et al., 2004]. Further, guesses at the preferences of *individual* users matter a great deal. If a_i prefers c_1 to c_2, then imputing the opposite ordering does not constitute good performance.

In contrast, when making social choices, one cares much more about the *aggregate* results than about the imputation of individual voters' preferences. In particular a voter with preferences $\succ_i$ ranks $c_1 \succ_i c_2$ and another voter with preferences $\succ_j$ ranks $c_2 \succ_j c_1$ but the opposite set of relationships are imputed for those two voters, the mistakes will (for most social choice rules) cancel out in the aggregate, since one mistake benefits c_1 and hurts c_2, while the other benefits c_2 and hurts c_1. Social choice also tends to be more concerned with the accuracy of the entire sequence of alternatives. While imputing a single poor recommendation at the top of a sequence in a recommender system would not be especially bad, provided that reasonable recommendations were still highly ranked on the whole, in social choice such a mistake could be catastrophic.

Although techniques from preference learning can indeed to be used for social choice, this thesis is concerned explicitly with the learning of preferences for applications in social choice, and so will prefer to select techniques that are better tailored to this goal. Often this will mean selecting techniques that have some theoretical grounding for social choice applications.

4.2 A General System

The previous section showed how a very simple learning approach could be used to resolve the motivational example from Chapter 3. Although it was able to resolve the example, there were still problems on which this technique made incorrect decisions. The section wrapped up with a discussion of the No Free Lunch theorem, explaining that no approach to social choice with incomplete information can perform better than others, except on specific subsets of the space of possible decision problems.

4.2.1 Informal Description

Machine Learning-, or Imputation-based approaches to the problem of social choice with incomplete information are a promising avenue for creating a method with a bias toward voters having homogeneous opinions, or at least opinions with consistent patterns. There are good reasons to believe that many elections will have homogeneous preferences. For example, in political contests, candidates are often arrayed on a left to right political spectrum, and the preferences of voters tend to reflect this. There are typically few voters who would rank both extreme left- and extreme right-wing parties high on their ballots. In problems like a hiring committee, although voters might have individual preferences, there is likely to be an overall trend among the candidates, with some objectively worse than others for the position. In problems like controlling a robotic swarm on Mars, individual opinions will often be a product of common environmental factors and attitudes to risk, which again, lead to a very small range of true preferences. Indeed, single-peaked preference domains, a nearly degenerate case of homogeneous preferences where *all* voters rank candidates relative to their opinions on a single issue, like the left-right political spectrum (i.e. voters view the candidates as embedded in a one-dimensional space), was described as "*the* canonical setting for models of political institutions" by prominent political scientists as recently as 2008 [Gailmard, 2008].

In this section, a general framework is proposed for using machine learning algorithms to address social choice problems. The framework is parameterized by a pair (S, M) consisting of a voting rule S and a learning algorithm M. The selection of effective combinations of voting rules and learning algorithms is considered in detail in later chapters. The system is described at an higher (i.e. more abstract) level first, before more specific versions are shown later in the section.

The overall system is depicted in Figure 4.3. Given a set of incomplete expressed preferences, the system applies a machine learning algorithm M to learn a model m that predicts true preferences (i.e. complete preferences) from expressed ones (i.e. incomplete preferences). The resulting model is then used to generate a guess at the true preferences of the voters, completing the expressed preferences. This guess is then passed to a social choice function S, which selects an outcome on that basis.

It should be apparent that if M learns the correct mapping from incomplete to complete preferences, then the guess at the true preferences will be correct. For example, if **Imputation Plurality** were given the expressed ballots in Figure 4.1 (right), it would output a learned model that maps from the ballot $b \succ a \sim c \sim d$ to $b \succ c \succ d \succ a$. Since this mapping is correct, ballots imputed by this model will be identical to the true preferences of voters, and the social choice function will make the correct decision. Likewise, if the mapping learned by M is close to correct, then the guessed preferences will be a close approximation of the true preferences. This is the intuitive justification behind the system. As outlined above, if there is no clear relationship between revealed and true preferences, this technique may fail to produce a good approximation of the voters' true preferences.

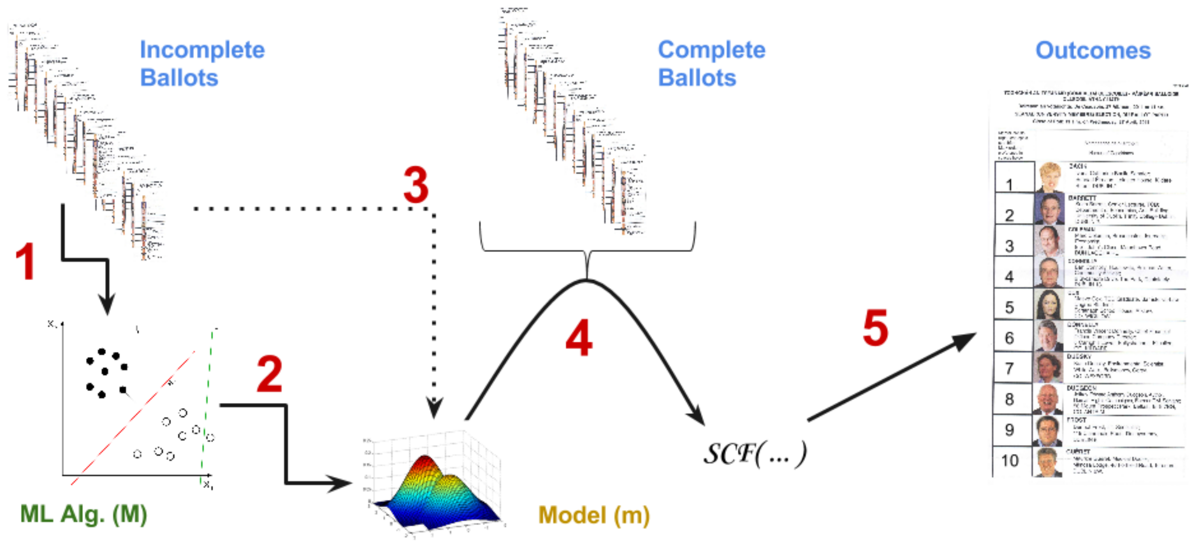


Figure 4.3: A graphical depiction of a general approach to combining social choice and machine learning. Incomplete expressed preferences (ballots) are passed to a machine learning algorithm (1), which outputs a model (2) capable of predicting true preferences from expressed preferences. The incomplete ballots are then passed to the model (3), producing a set of complete ballots. Complete ballots are passed to a social choice or social welfare function (4), which outputs a decision (5). To the extent that the model is a true mapping from expressed to true preferences, the complete ballots will match the true preferences of the voters, leading to a decision that closely approximates the correct decision.

Even if there is a relationship, if M is not well suited to discovering relationships of the type that are present, the guessed preferences may be far from correct. The choice of M will be central to the success of the approach.

4.2.2 Example

As a concrete example[4], ballots might be completed using Imputation via Classification. Imputation is the process of replacing missing data (i.e. "missingness") with a carefully selected guess at the missing value [Schafer, 1999]. For instance, if a person's age is missing from an otherwise complete questionnaire, a very simple imputation technique would replace the missing age with the average age of the other questionnaire-takers. A more sophisticated technique would be to use the age of another questionnaire-taker with similar demographic characteristics, especially those known to be correlated with age.

Classification algorithms operate over a data matrix X and label vector Y, with an equal number of rows. The algorithm finds a model (a function) m such that $m(X)$ is a vector of labels, and a cost function $G(m, m(X), Y)$ is minimized. The set of possible functions from which m may be selected is called the *hypothesis space*, and is restricted in a way intended to ensure that m captures general patterns in the data that will allow it to accurately predict the labels of previously unseen datapoints in future. G often has some related component based on the structure of m as well as the number of differences (errors) between $m(X)$ and Y. For instance, G might penalize functions which are not very "smooth", meaning those that are highly sensitive to small changes in the values of x.

This example will assume that ballots are top-ordered, for simplicity. The votes are indexed such that $B_{i,j}$ is the j^{th} most preferred candidate of voter i . If voter i has specified only j preferences, then $B_{i,k} = \emptyset, \forall\, k > j$. A given ballot $b_i \in B$ thus represents a total ordering over an arbitrary subset of the set of the candidates, C. The elements of C that are not on the ballot are of lower rank in the voter's preferences than those candidates that were ranked. Finally, the vote matrix formed by the first j preferences of each ballot is denoted with B_j. For instance, B_1 denotes the first preferences of every ballot, while B_2 denotes the first *and* second preferences of every ballot.

To impute the missing information in B, a classification system might begin by extracting B_2, the first and second preferences of every voter's ballot. Some ballots may state only a single preference, while others state two. Taking the subset of B_2 which is complete ($B_{c_2} = \{b \in B_2 |\ |b| = 2\}$), a classification algorithm M is used to train a classifier

[4]This example is based on text from the author's published work [Doucette et al., 2015].

$m_2 = M(B_{c_2})$, which predicts the second preference of each ballot from their first preferences. The data matrix X used for classification is some function Φ of the first preferences on every ballot in B_{c_2}. The label vector Y is the second preference of every ballot in B_{c_2}. Once m_2 has been computed, it is used to impute $B_2 \setminus B_{c_2}$, generating a complete ballot matrix of two columns. This imputed ballot matrix is called B'_2.

The process is then extended to the next set of preferences in B. The ballot matrix of the first, second and third preferences (B_3) is taken and build a classifier m_3 on B_{c_3}. m_2 can then be used to impute all missing second preferences, and m_3 to impute $B_3 \setminus B_{c_3}$ and generate B'_3. This process can then be iterated until the generation of $B'_{|C|} = B'$, an imputation of the entire ballot matrix. A winning outcome can then be selected by applying any standard voting rule S to the imputed matrix to select a member of the outcome set O (typically either a candidate, or a linear ordering over the candidates): $S(B') = o'$.

The described process works only for top orders, but should be easily understood as a basis for the technique. Next, a description of a more general approach to using machine learning algorithms to impute missing ballots and thus decide an election with incomplete information is described. The approach described is conceptually similar to the one for top orders discussed in this example, but works for any set of partially ordered ballots, and can use learning algorithms that are not explicitly intended for classification, or that impute preferences in a different order than the way described here.

4.2.3 Formalized System

More formally, the system is described in a very general way in Algorithm 1. The system can use *any* machine learning algorithm M capable of learning a mapping from partial to complete preferences. M is applied to the ballots (expressed preferences) of the electorate, producing a model m, which can then be used to impute the original ballots. The imputed ballots are then used to select an outcome[5] via voting rule S. The runtime of Algorithm 1 depends entirely on the runtimes of its parameters, and is in $O(M(B) + m(B) + S(m(B)))$. For instance, if model selection is done using logistic regression with k features, and there are $|C|$ candidates, then the runtime of $M(B)$ is in $O(lk|B||C|^2)$, where l is the number of iterations needed, and $m(B)$ is in $O(k|B||C|^2)$. If **Borda** is used for S, then S is in $O(|C||B|)$, so the overall runtime of a system that used logistic regression with k features would be dominated by the training cost of the models. In contrast, if a more computationally expensive voting rule like **Kemeny-Young** were used (which requires $O(|C|!|B||C|)$

[5]Alternatively, a multiple-imputation approach [Schafer, 1999] could be used to provide an estimate of the uncertainty in the decision, rather than a simple point estimate of the outcome.

work when computed naively), then the total runtime would be dominated by S instead.

Algorithm 1 An algorithmic formulation of the proposed system for applying conventional machine learning algorithms to social choice problems with incomplete information. Given machine learning algorithm M, a social choice function for complete preferences S, and a set of ballots (expressed preferences) B over candidates C, the algorithm returns a member of output set O.

```
procedure IMPUTATION-ELECTION($M$,$S$,$B$, $O$, $C$)
    let $m = M(B)$ be a model s.t. $m : \mathcal{P}(C)^{|B|} \to \mathcal{L}(C)^{|B|}$.
     return $S(m(B))$
end procedure
```

Although Algorithm 1 is very general, it does not provide a clear road-map for using existing machine learning algorithms. How exactly should a model that maps incomplete to complete preferences be produced? In later chapters, some new domain-specific algorithms[6] are considered, but for now, an approach that allows any supervised learning algorithm to be applied to the problem is specified instead. Algorithm 2 describes a machine learning algorithm that wraps any standard supervised learning algorithm capable of multi-class classification, and allows the production of an imputation model using partial ballots. Each candidate in C is treated as a possible label (class) for a partial ballot. The supervised learning algorithm is used to find a mapping between incomplete ballots and their next preference (which will be one of the possible labels in C). Like Algorithm 1, the runtime of Algorithm 2 depends almost entirely on the choice of M_c and F, and is in $O(|C|(|B| + F(X) + M_c(X,Y))$.

The algorithm starts with an empty list of sub-models, m. Then, for each possible position in a ranking of candidate set C (i.e. there are 1 to $|C|$ positions to rank a candidate from among a set of $|C|$ candidates), a classifier m_i is trained, and added to m. The resulting list of classifiers that is returned by Algorithm 2 contains one classifier for each ballot position.

The training of each classifier consists of first filtering out just those ballots that rank a single candidate in position i. This could occur if voters provided a top-t style ranking (listing their t most preferred candidates and leaving the remainder tied for last place), as seen in the motivational example of Chapter 3. It could also occur with more general partial preferences if there are clear constraints present in the ballot. For instance, if there

[6]i.e. designed to address the problem of completing ballots with missing information.

Algorithm 2 A general algorithm for converting any supervised learning algorithm to an imputation algorithm suitable for use in social choice with incomplete information. The algorithm iteratively learns sub-models that predict i^{th} preferences of voters on the basis of whatever preferences have been provided by the voter, or specified by earlier models.

```
procedure ImputationModel($M_c$, $B$, $C$)
    let $m \leftarrow \emptyset$
    for $1 \leq i \leq |C|$ do
        let $B_i$ be the subset of $B$ with a specified $i^{th}$ preference
        let $Y_i$ be the $i^{th}$ preferences of every ballot in $B_i$.
        let $X_i$ be $B_i \setminus Y_i$
        let $F : \mathcal{P} \rightarrow \mathbb{R}^z$ be a featurization function, mapping partial ballots to a $z$-dimensional real-valued feature space.
        let $m_i \leftarrow M_c(F(X_i), Y_i)$
        let $m \leftarrow \{m, m_i\}$
    end for
    return $m$
end procedure
```

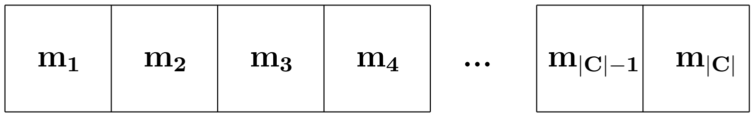


Figure 4.4: A depiction of a chained classifier model, as output by Algorithm 2. The output model m is a list of classifiers, $\{m_1, ..., m_{|C|}\}$. Model m_i predicts the i^{th} preference of a given ballot.

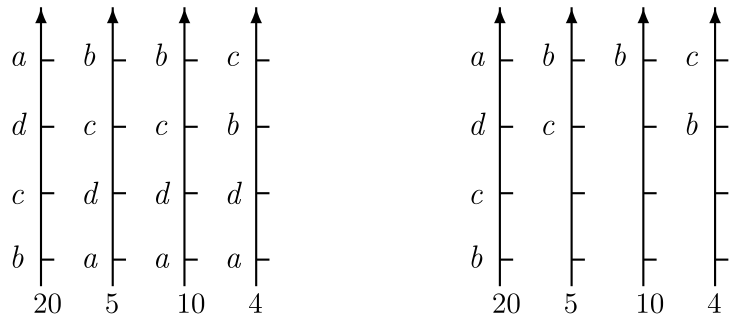


Figure 4.5: Example preferences used to elaborate on the process described in Algorithm 2.

is a unique candidate that is preceded by $i-1$ others, then that candidate has effectively been ranked in i^{th} place, even if the ballot does not specify a total ordering on the $i-1$ candidates that precede it. For example, the preferences $c_1 \sim c_2 \sim c_3 \succ c_4$, do not specify any relationship between c_1 through c_3, but does state that all are preferred to c_4. There is no definite candidate in places 1 through 3, but there is a definite candidate in position 4, so this ballot would be included in B_4, the subset of B in which all voters specify a definite fourth preference.

Once the ballots have been filtered to include only those with a definite i^{th} preference, they are further partitioned. The i^{th} preferences of each ballot are split off into the label set Y_i. For each element $b_j \in B_i$, a label $y_j \in Y_i$ denotes their i^{th} preference. Removing all information about y_j from b_j yields the corresponding input preference $x_j \in X_i$. This is now a supervised learning task: to find a mapping from X_i to Y_i. X_i can then be featurized, converted to a matrix representation, and the matrix can be augmented by Y_i to create a canonical representation of a classification task, suitable for use by any supervised learning algorithm. The algorithm M is then applied to the problem to produce a classifier m_i.

As a more detailed example, consider the preferences in Figure 4.5. To train classifier m_1, the subset of ballots with a definite first preference, B_1, is determined. This is the complete set of ballots, because every voter has specified a unique first preference. The labels Y_1 would comprise 20 elements with value a, 15 with value b, and 4 with value c. The input data X_1 would be 20 ballots specifying $d \succ c \succ b$, but not (explicitly) whether a was first; 5 stating only that c was second; 10 blank ballots, and 4 ballots ranking b second and stating no other information. X_1 is thus the original set B_1, but with all information on first preferences suppressed (i.e. X_1 is $B_1 \setminus Y_1$). The classification algorithm will learn a sub-model m_1 that maps from X_1 to Y_1. Then B_2 will be computed, containing all the ballots *except* the 10 that are missing a definite second preference. The set Y_2 will be 20 elements with value d, 5 elements with value c, and 4 elements with value b. The set X

will contain twenty elements with values $a \succ c \succ d$ (associated with label b); five with $b \succ c \sim d \sim a$ (associated with label c); and four elements with label $c \succ d \sim b \sim a$ (associated with label b). The machine learning algorithm M will output a classifier m_2 that maps from X_2 to Y_2.

Although a list of models in the form produced by Algorithm 2 is a valid output, it is not immediately clear how it should be used to complete an incomplete ballot, or even that it is a reasonable way to convert a classification algorithm to an imputation algorithm for this domain. Algorithm 3 describes the process of imputing a ballot using a model of this form. The ballot is iteratively imputed. Its highest ranked missing preference is selected on the basis of any ballot information that is provided, and then the next highest on the basis of both the original information *and* the imputation made by the first model. This process is repeated until the ballot is completed. There are a few oddities in the algorithm itself. In particular, a perfectly valid sub-model may predict that the correct candidate for a given location is one that has already been assigned to another location. To prevent this, the algorithm imputes the candidate to which the submodel m_i assigns the highest confidence, from among the set of *valid candidates* that could be imputed at this location. For example, suppose b_j expresses the relation $c_1 \sim c_2 \sim c_3 \succ c_4$. Algorithm 3 will first try to impute position 1, and uses model m_1 assign it to candidate c_1, yielding the preferences $c_1 \succ c_2 \sim c_3 \succ c_4$. Model m_2 is then applied, but predicts that c_1 is also the best choice for position 2. The algorithm will compute the set $\varsigma = \{c_2, c_3\}$ of valid alternatives that can be assigned to this position, and selects the element of ς to which m_2 assigns the highest confidence[7]. As with the two earlier algorithms, the runtime of this algorithm depends mostly on the runtime of parameter m, and is in $O(|C| + |C|^2 m(x))$.

It is worth noting that there exist other approaches to converting standard classification algorithms into algorithms suitable for the kind of structured imputation that must occur in social choice with incomplete information. A reasonable alternative might be to create a much larger set of models, each of which answers the question "$c_j \succ c_k$?", rather than the models from Algorithm 2 which answer the question "Which candidate should be imputed at position i?". Such an approach would have the advantage of phrasing a simple binary question. However, this arrangement becomes problematic when applying the resulting model to impute an incomplete ballot. For instance, suppose a ballot expresses the preferences $c_1 \sim c_2 \sim c_3 \succ c_4$ again. The model imputes that $c_1 \succ c_2$, and that $c_3 \succ c_1$, but also that $c_2 \succ c_3$. By the transitive property of a partial order, it cannot be the case that $c_3 \succ c_1 \succ c_2$ and $c_2 \succ c_3$. This is similar to the problem addressed in Algorithm 3 where models for two different positions both express the highest confidence

[7]This convention is adopted because not all supervised learning algorithms output confidence measures that are interpretable across models.

Algorithm 3 An algorithm describing how models created using the ImputationModel procedure of Algorithm 2 can be used to convert an incomplete ballot (i.e. expressed preference) to a complete ballot (i.e. imputed preference). The algorithm iteratively fills in the ballot from highest-ranked to lowest-ranked preference by using the appropriate sub-model for each position.

```
procedure IMPUTEBALLOT($m$, $b_j$, $C$)
    for $1 \leq i \leq |C|$ do
        if $b_j$ has a unique candidate ranked at position $i$ then
            //Do nothing.
        else
            Let $\mathbf{c} \leftarrow m_i(F(b_j))$
            //$\mathbf{c}(c_k)$ is $m_i$'s confidence $c_k$ belongs at $i$ in $b_j$.
            Let $\varsigma \leftarrow \{c \in C | (\mathrm{Pos}(c, b_j) < i + |\varsigma|) \wedge (\mathrm{Pos}(c, b_j) > i - 1)\}$
            Set position $i$ on $b_j$ to $\mathrm{argmax}_{c \in \varsigma}\, \mathbf{c}$
        end if
    end for
    return $b_j$
end procedure
```

in the same candidate. Since the candidate cannot be in two positions on the ballot at once (in a total ordering), it is assigned to the higher position, and the lower position is assigned to the candidate the second model is second most confident about. However, the problem appears greatly exacerbated in this alternative representation, as many more models must be compared, and more importantly, it is not clear what order they should be compared in. In Algorithm 3, picking a *position* to impute first (e.g. the first position on a ballot) is unbiased with respect to the set of candidates. Before seeing the data (ex ante), all candidates are equally likely to benefit from this arrangement. However, in the alternative pairwise representation, each model has been assigned to just two candidates, and producing something that closely resembles Condorcet's paradox [Black et al., 1958; de Caritat, 1785], since the order in which candidates are compared determine the order they appear on in the ballot.

4.2.4 Conventional Machine Learning as Social Choice

The previous subsection describes a general approach to resolving social choice problems with incomplete information using machine learning algorithms to impute missing preference information. The next chapter offers an empirical validation of this idea, comparing performance against state-of-the-art-competitors, and the first section of this chapter outlined the ability of this technique to solve problems that were difficult for existing techniques. Now this section offers a theoretical connection between the proposed technique and social choice techniques in general. The result is interesting because it provides a means of viewing supervised learning algorithms as social choice functions, allowing them to be studied with a rather different set of tools. For instance, if a supervised learning algorithm is equivalent to some kind of social choice function, it becomes reasonable to ask questions about how *fair* the algorithm is. It also allows for direct theoretical comparisons between different combinations of voting rules and machine learning algorithms, and between with other ways of deciding an election with partial information, a topic addressed in detail in Chapter 8[8].

The section begins by showing that any supervised learning algorithm that learns a deterministic mapping from partial ballots to complete ballots on the basis of other voters' ballots is equivalent to a social choice function, though one that operates over a somewhat strange domain. Conversely, it is straightforward to convert any social choice function into a supervised learning algorithm for this domain, by using the ordering the algorithm outputs as a basis for imputation.

Consider a learning algorithm M that learns to impute the i^{th} position on partial ballots over a set of k candidates C, on the basis of a set of n incomplete ballots B. The algorithm's output is a policy $m_i : \mathcal{P}_{\neg i} \rightarrow \mathcal{P}_i$, where $\mathcal{P}_{\neg i}$ is an partial order in which no candidate has *exactly* $i - 1$ candidates preceding it, and $|C| - i$ following it, and $\mathcal{P}_i$ is the set of ballots that *do* have such a candidate. Call such an algorithm a "classifier". That is, a classifier is a function mapping from ballots that *do not* have an explicitly stated i^{th} preference, to a version of that ballot that *does*.

Lemma 1. *Any classifier is equivalent to a social choice function.*[9]

Proof. A social choice function in this context is any function mapping a set of complete

[8]A simplified version of these results appeared in [Doucette et al., 2015]. However, while those results showed that every classifier or chained classifier was a social choice function over some output space, here the author also explicitly shows that chained classifiers are subject to Arrow's Theorem [Arrow, 2012] and the Gibbard-Satterwaithe Theorem [Gibbard, 1973; Satterthwaite, 1975]. The earlier publication stated that chained classifiers were subject to both theorems simply by virtue of being members of a general family of social choice functions. This was correct, since the social choice functions were not over the same set of alternatives as in in the original problem, but did not show clearly the applicability of important possibility results, which are proven here.

[9]This proof is reproduced from [Doucette et al., 2015] with alternations in notation (for consistency), and an elaboration on the importance of the mapping function f.

orderings over some finite set of outcomes O, to a single outcome. A classifier that predicts missing values in the i^{th} candidate on n ballots over $|O|$ candidates functions by selecting and then applying a policy. Note that, if there are N_{ic} ballots that do not specify a i^{th} preference in B, and the preceding $i-1$ preferences on each ballot have all been imputed by other means, then there fewer than $(|O|-i+1)^{N_{ic}}$ possible imputations that could be created (each row has $i-1$ elements already specified, out of $|O|$ outcomes, and in the worst case every ballot is unique). Call this the outcome space O_{impute} of the classifier, and note that this space is finite.

Note that, although orderings over O are not complete orderings over O_{impute}, it is possible to automatically define an onto mapping from partial orders over O_{impute} to partial orders containing at least $j-1$ elements of O. This automatic construction could be incorporated into a social choice mechanism. If $|\mathcal{P}(O_{impute})| > |\mathcal{P}(O)|$, arbitrary onto mappings can be constructed automatically. However, for some problems O_{impute} will be smaller than O, and consequently, constructing an onto mapping becomes more complex. In these cases, a chain of mappings $f = \{f_1, ..., f_n\}$ is relied on. Ballots over O_{impute} are sorted (via any comparison function q, as long as identical ballots are adjacent), and the first copy of each possible ballot is mapped to a ballot over O according to mapping f_1. The second copy is mapped according to f_2, and so on. The resulting chained mapping is still deterministic.

Represent the process of selecting an imputation policy m from the completed votes as a function M, and note that this function is deterministic and produces exactly one output (M is a deterministic model selection algorithm for classification). Define $m_{candidate} = M(B_c)$ as the policy selected by applying M to the ballots in B to select an imputation policy, and $B_{candidate} = m_{candidate}(B_{ic})$ as the $N_{ic} \times 1$ vector produced by the application of this policy to the ballots not specifying a i^{th} preference ($B_{ic} = B \setminus B_c$).

Then a scoring function S_j that maps from sets of preferences over O_{impute}, called B_{impute}, to elements of O_{impute} can be defined:

$$S_j(B_{impute}, o) = -\Delta(M(f(q(B_{impute})))(f(q(B_{impute}))), o)$$

where Δ is the number of vector components in which the two vectors have different values (i.e. the sum over component-wise delta functions)[10]. A social choice function over O_{impute} that selects the outcome with the highest score in S_j will produce identical output

[10]Note that if q has re-ordered the ballots, than o must be re-ordered correspondingly (i.e. if ballot b_j was imputed with the value at position o_j, then the element of $M(f(q(B_{impute})))(f(q(B_{impute})))$ that corresponds to the original j^{th} ballot must be compared to o_j)

to a classifier selected via M, and thus, is equivalent. Note that ties are not possible for this rule, since there exists a unique member of O_{impute} with score 0 under S_j. $\square$

Although the resulting social choice functions might have very strange properties, there can be no doubt that a classifier is effectively aggregating the opinions of voters on how ballots should be imputed. The mapping f is a slightly odd component, but ultimately there exist voting rules like **Plurality** that function similarly, effectively mapping richer preferences to a simplified space before making a decision. Arguably rules that make decisions on partial ballots directly are performing the opposite mapping (at least in some cases), implicitly converting voter preferences that were specified in a small space to preferences in some larger space. For instance, **Borda** applied to a set of ballots that each only specify preferences about two favourite candidates (i.e. a top-2 preference) from a set of a hundred is implicitly mapping preferences sampled from a small preference space ($\binom{100}{2}$) to a much larger one (100!).

Theorem 1. *Every chained classifier is equivalent to a social choice function.*

Proof. The proof is by induction on the number of candidates.

It is easy to show using Lemma 1 that if $|C| = 2$, any chained classifier has an equivalent scoring function and thus an equivalent social choice function. There is only one decision to make (since imputing one candidate fully determines the position of the other one), and Lemma 1 shows that a classifier that imputes just one position is equivalent to some social choice function.

From this, it is possible to define a scoring function for larger values of $|C|$. Here B_{impute} is defined as being preferences over the set of possible imputations of the *entire* set of missing preferences in the ballots, rather than just over the set of missing i^{th} preferences. This outcome space is necessarily larger than the one used in Lemma 1, so an analogous onto mapping f can be constructed. o is analogously defined to be a member of the set of possible imputations of the *entire* set of missing preferences in B. M_i is defined as a classification algorithm used to select a policy for the i^{th} position on the ballots, and o_i as the portion of o consisting of the definite assignment of candidates to position i. A social choice function can then be recursively defined using the recurrence:

$$S_i(B_{impute}, o) = -\Delta(M_i(f(q(B_{impute})))(f(q(B_{impute}))), o_i) + S_{i-1}(B_{impute}, o)$$

with the base case that $S_0(B_{impute}, o) = 0$. The rule $S_{|C|}(B_{impute}, o)$ is then equivalent to a chained classifier that uses algorithms $\{M_1, ..., M_{|C|}\}$ to learn to impute i^{th} preferences. $\square$

Impossibility Results for Classifiers

Theorem 1 shows that the process described in Algorithm 3 is equivalent to some form of social choice function. The status of the method as a social choice function implies that important results like Arrow's Theorem [Arrow, 2012; Pini et al., 2009] and the Gibbard-Satterwaithe Theorem [Gibbard, 1973; Satterthwaite, 1975; Reffgen, 2011] apply (modulo their more modern augmentations to cope with partial orders). However, the proof of Theorem 1 relies on mapping the learning process onto social choice functions with a different domain and range than the original problem. One might wonder then whether the algorithms are subject to the results of Arrow and Gibbard-Satterwaithe with respect to the *original problem domain*. That is, if voters cast a ballot over the original set of candidates $|C|$, then is a method that is used to impute ballots on the basis of this information subject to these impossibility results?

Recall that Arrow's Theorem demonstrates the mutual incompatibility of four criteria in elections with at least 3 alternatives:

1. Transitivity: If the system ranks $a \succ b$, and $b \succ c$, then it must also rank $a \succ c$.
2. Unanimity: If *every* voter ranks $a \succ b$, the system does as well.
3. Independence of Irrelevant Alternatives: If the system ranks $a \succ b$, then it does so whether or not a third candidate c is present in the election.
4. Non-dictatorship: There exists no voter v_i such that if $a \succ_i b$, then the system also ranks $a \succ b$, regardless of the preferences of other voters.

Unanimity is violated by chained classifiers, because they act as deterministic mappings between incomplete and complete ballots. If voters simply provide rankings over the set of candidates, there is no way for a voter to specify that they prefer an outcome where two identical ballots are imputed in very different ways, for instance. However, such an outcome is possible, and might even be desirable if the voter prefers to split the support for a popular candidate, or considers two options equally good. Thus, even if all voters prefer it, it can never be achieved. More intuitively, if the set of candidates is smaller than the set of possible imputations, than clearly there is not an onto mapping from voters' individual preferences to the space of preferences over outcomes, even if (as shown in Lemma 1) there is a mapping to any possible *profile*. Effectively, voters are restricted to *self-consistent* preferences: if they prefer a to b, then the also prefer that $a \succ b$ as often as possible on imputed ballots. The next two proofs show that this constraint (perhaps surprisingly) does not impact the applicability of the aforementioned impossibility results.

First, some logical extensions of Arrow's axioms to this slightly different domain are required. An imputation is said to be *elementwise transitive* if, when imputing a particular ballot, ranking $a \succ b$ and $b \succ c$ among the imputed preferences implies that it has also ranked $a \succ c$. It is said to be *elementwise unanimous* if, given that every voter ranks $a \succ b$, then when imputing a given ballot it also ranks[11] $a \succ b$. It is said to possess *elementwise I.I.A.* if when the system ranks $a \succ b$ during the imputation of a single ballot, it does so independent of whether any other single candidate is present in the election. Finally, the system is said to possess *elementwise non-dictatorship* if there exists no voter v_i such that if $a \succ_i b$, then the system also ranks $a \succ b$, regardless of the preferences of other voters. A voter is said to be an *elementwise dictator* if its existence violates the elementwise non-dictatorship property. A *global dictator* is an elementwise dictator for every decision about the relative ordering of a pair of candidates.

Lemma 2. *If an imputation method satisfies elementwise transitivity, unanimity, and I.I.A., then there exists a voter v_i who is an elementwise dictator for* any *single imputation decision involving at least three alternatives.*

Proof. Consider the case where only 1 ballot contains missing information, and it has left the relative order of at least three candidates indeterminate. By the elementwise I.I.A. property, only the preferences expressed by voters over these indeterminate candidates can be considered. The system is making a decision over a domain consisting of orderings over the candidates, using partial ballots expressed over those candidates. By assumption the method satisfies elementwise transitivity, unanimity and I.I.A. When making decisions over a domain consisting of orderings of candidates using partial ballots expressed over candidates, elementwise transitivity, unanimity and I.I.A. are identical to the non-elementwise versions of these axioms. Three of the four conditions of Arrow's Theorem (for partial ballots [Pini et al., 2009]) are thus satisfied, therefore non-dictatorship cannot also be satisfied. □

Theorem 2. *If an imputation method satisfies elementwise transitivity, unanimity, and I.I.A., then there exists a voter v_i who is an elementwise dictator for* every *imputation decision involving at least three alternatives (i.e. a global dictator).*

Proof. First, consider the case where there exists a ballot b_1 that is missing *every* candidate (i.e. a completely empty ballot). By Lemma 2, there is a dictator for this imputation

[11] Note that following such an imputation, this property would still hold for all voters that rank a relative to b.

decision. Call this voter v_i, and denote their preferences $\succ_i$. Assume without loss of generality[12]that $\succ_i$ is a total order, and therefore that b_1 was imputed with exactly $\succ_i$. The order imputed to ballot b_1 is called $\succ_1$.

Now suppose there exists some other ballot to be imputed b_2. If the ballot is missing every candidate, then the imputation decision is made over the same set of outcomes as for b_1, and using the same set of preferences. Since the imputation rule creates a deterministic policy, it must impute b_2 with exactly the same ordering, $\succ_i$, the preferences of the dictator for b_1. So v_i is a dictator for b_2 also.

Suppose b_2 is missing fewer candidates than $|C|$. By elementwise I.I.A., if the system ranks $a \succ b$ when the imputation decision is an election over the entire set of candidates, it must *also* rank $a \succ b$ when the decision is an election over some subset of them. However, it has just been shown that v_i is a dictator for *any* election involving the entire set of candidates. Therefore, $a \succ_1 b$ in such an election if and only if $a \succ_i b$. It follows that in this smaller election, $a \succ_2 b$ if and only if $a \succ_i b$ also. Therefore v_i is a dictator for this election also. □

This result shows that imputation methods must violate one of Arrow's criteria. Very simple imputation approaches like hot-deck imputation (i.e. simply selecting the preferences of a single voter with a total ordering and using them to impute all missing information) are obviously dictatorial. However, most methods appear to fail the I.I.A. criteria. For example, the logistic regression method described in the next section uses information about which candidates a voter has already ranked to determine how their preferences should be imputed. Intuitively it is not even clear that elementwise I.I.A. is a desirable property for imputation systems to possess. For example, if a voter ranks the Socialists first on their ballot, then a very different relative ordering for the Liberals and Conservatives might be computed than would be if the voter had not ranked the Socialists at all. Although unaminity in general is violated, if voters are limited to self-consistent preferences, unaminity within this restricted domain seems important to preserve, and transitivity and non-dictatorship both seem vital. It is not therefore so concerning that I.I.A. is the axiom that imputation methods seem predisposed to violating.

A very similar result can be derived for the Gibbard-Satterwaithe Theorem. An imputation method is defined as elementwise incentive compatible if a voter that prefers a to b and reports preferences $\succ_i$ in a single imputation decision where the system imputes $b \succ a$ cannot reverse the imputation decision by instead reporting preferences $\succ_i'$.

[12]If v_i does not rank some candidates, it is easy to show by elementwise I.I.A. that v_i is still a dictator for the relationships between all the candidates it does rank.

Lemma 3. *If an imputation method satisfies elementwise incentive compatibility and can return at least three alternatives for any elementwise decision with at least three alternatives (when provided with some set of preferences), then there exists some voter v_i that is a dictator for any elementwise decision.*

Proof. Consider the case where only 1 ballot contains missing information, and it has left the relative order of at least three candidates indeterminate.

If the imputation method satisfies elementwise incentive compatibility it must also satisfy elementwise I.I.A.. By the elementwise I.I.A. property, only the preferences expressed by voters over these candidates can be considered. The system is making a decision over a domain consisting of orderings over the candidates, using partial ballots expressed over only those candidates. Since there is only one element, the elementwise version of incentive compatibility is identical to the general version of that axiom. Since by assumption the imputation method also satisfies the ability to output at least three alternatives for this decision, then it must be subject to the Gibbard-Satterwaithe theorem (for partial ballots [Reffgen, 2011]), so non-dictatorship cannot also be satisfied. $\square$

Theorem 3. *If an imputation method satisfies elementwise incentive compatability, and can return at least three alternatives for any elementwise decision with at least three alternatives (when provided with some set of preferences), then there exists a* single *voter v_i that is a dictator for* every *elementwise decision (i.e. a global dictator).*

Proof. First, consider the case where there exists a ballot b_1 that is missing *every* candidate (i.e. a completely empty ballot). By Lemma 3, there is a dictator for this imputation decision. Call this voter v_i, and denote their preferences $\succ_i$. Assume without loss of generality[13]that $\succ_i$ is a total order, and therefore that b_1 was imputed with exactly $\succ_i$. Call the order imputed to ballot b_1 $\succ_1$.

Now suppose there exists some other ballot b_2 requiring imputation. If the ballot is missing every candidate, then the imputation decision is made over the same set of outcomes as for b_1, and using the same set of preferences. Since the imputation rule creates a deterministic policy, it must impute b_2 with exactly the same ordering, $\succ_i$, the preferences of the dictator for b_1. So v_i is a dictator for b_2 also.

Suppose b_2 is missing fewer candidates than $|C|$. By elementwise I.I.A., if the system ranks $a \succ b$ when the imputation decision is an election over the entire set of candidates, it must *also* rank $a \succ b$ when the decision is an election over some subset of them. However,

[13]If v_i does not rank some candidates, it is easy to show by elementwise I.I.A. that v_i is still a dictator for the relationships between all the candidates it does rank.

it has just been shown that v_i is a dictator for *any* election involving the entire set of candidates. Therefore, $a \succ_1 b$ in such an election if and only if $a \succ_i b$. It follows that in this smaller election, $a \succ_2 b$ if and only if $a \succ_i b$ also. Therefore v_i is a dictator for this election also. □

Given the applicability of Arrow's result, the Gibbard-Satterwaite result is not surprising. However, it does imply that voters will have incentives for strategic behaviour when they submit information to an imputation method. Chapter 9 discusses this problem in greater detail, including possible remedies.

4.3 An Initial Implementation

While Section 4.1 showed how machine learning could be used to solve the motivational example, and argued that the assumptions underlying such a system were reasonable, and Section 4.2 formalized a general system for applying machine learning to problems in social choice with incomplete information, this section will discuss the finer-grained implementation details of the first version of such a system. While the system will conform to the general model from Section 4.2, there are many important details that were not specified in that section, including the choice of learning algorithm, what features should be extracted from an incomplete ballot to best facilitate learning, and how to trade off model complexity and model performance in a system that imputes its own training data. The section describes a complete implementation, which is then evaluated empirically in the next chapter.

4.3.1 Algorithm Selection

The core learning algorithm used in the initial experiments was L_2 Regularized Logistic Regression. Logistic Regression is a simple supervised learning algorithm originally developed by statisticians in the 1950's [Cox, 1958]. McCullagh and Nelder [1989] provide a good modern overview of the model.

Given a $n \times k$ matrix of k features over n data points, $\mathbb{X}$, augmented with a $n \times 1$ column vector of all ones; and a column vector $\mathbb{Y}$ of n *binary* labels, logistic regression finds a $k \times 1$ column vector of k weights $\mathbb{W}$ according to the optimization problem:

$$\max_{\mathbb{W}} \sum_i \log\left(\frac{1}{1+\exp(-x_i\mathbb{W})} I(Y_i) + \left(1 - \frac{1}{1+\exp(-x_i\mathbb{W})}\right) I(\neg Y_i)\right) \tag{4.1}$$

where $||\cdot||$ is the Euclidian norm function (i.e. the square root of the dot-product of the vector with itself), and the other arithmetic operations are performed element-wise. This amounts to maximizing the sum of the likelihood of each binary label $y_i \in \mathbb{Y}$ given $x_i \in \mathbb{X}$, under the assumption that $\mathbb{W}$ has taken on a given value.

Figure 4.6 shows a pictorial representation of the optimization problem for 1-dimensional data, along with several different models fit to the data. In this example problem, the data are given by:

$$\mathbb{X}^{\mathbb{T}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -6 & -5.2 & -4.5 & -3.8 & -2.8 & -1.8 & -1.2 & 0.5 & 1 & 2.1 & 2.8 & 3.9 & 5.8 \end{bmatrix} \tag{4.2}$$

$$\mathbb{Y} = \{0,0,0,0,0,1,0,1,1,1,1,0,1,1\}^T \tag{4.3}$$

Note that the leftmost column of $\mathbb{X}$ is a set of dummy variables that are used to fit the intercept of the model, exactly as in standard linear regression. The model is comprised of two weights, $\mathbb{W} = \{w_0, w_1\}^T$. w_0 contributes the same value for every point, and has the effect of controlling the position of the curve, translating it along the x_1 axis. w_1's contribution depends on the value of the second column of $\mathbb{X}$, and has the effect of making the curve steeper or less steep along that axis. The same effects generally hold for logistic regression problems in higher dimensions. Each weight controls the steepness of the curve along one dimension, and w_0 controls the offset of the curve. The predictions made by the model correspond directly to points on a logistic curve. For example, suppose that $w_0 = 0$ and $w_1 = 1$, corresponding to the curve depicted in Figure 4.6 (Top-Left). The model predicts a value of $\frac{1}{1+\exp(-(0+1\times-6))} = 0.002$ for the first point in $\mathbb{X}$, very close to the labelled value of 0. On the other hand, the 6^{th} point does not fit very well, with a predicted value of $\frac{1}{1+\exp(-(0+1\times-1.8))} = 0.14$, which is very far from the labelled value of 1. The predictions generated by a logistic regression model correspond to an estimate of the conditional probability that a given point has label 1 and not label 0 given the value of x.

Standard logistic regression predicts only binary labels, but in the imputation-based approach to social choice, one must predict which of several candidates will come next on a ballot. If there are more than two possible alternatives, one logistic model can be created for each class in the One-Versus-All style of multiclass classification. Every model is trained on the same matrix $\mathbb{X}$ (the details of $\mathbb{X}$'s construction are provided in the next section), but a model trained to predict whether the next candidate should be c_i or not uses a label vector $\mathbb{Y}$ which has value 1 only for votes that did rank c_i as the next candidate. When predicting which candidate should be ranked next for an ballot that is missing information, each of the models (one for each candidate) predicts whether or not this ballot should be completed with c_i. Since the outputs of the models correspond to estimates of the probabilities that the ballot should be completed with each candidate, the label corresponding to the model that outputs the highest value is then selected. In the event of a tie, a decision is made uniformly at random from among the tied candidates.

Logistic regression is used in initial experiments because it is a relatively simple model, with few parameters that need manual configuration, and because it is easy to interpret.

4.3.2 Feature Construction

This section discusses the construction of a numeric matrix $\mathbb{X}$ from a partial order, in a way that will facilitate the application of logistic regression to the problem. A reasonable starting point is to represent a ballot *by a set of pairwise comparisons.* These columns of $\mathbb{X}$ encode three-valued indicator variables showing whether $c_j \succ c_k$ or not for every pair of candidates. These columns (features) are denoted with $I_{c_j \succ c_k}$. A value of 1 indicates that $c_j \succ c_k$, a value of -1 indicates that $c_k \succ c_j$, and a value of 0 indicates that $c_j \sim c_k$ (i.e. that the ordering of the two candidates is unknown). For example, ballot $c_1 \succ c_2 \sim c_4 \succ c_3$ could be encoded in the fashion shown in Table 4.2. This encoding is relatively compact ($O(|C|^2)$), and the coefficients of a logistic regression model trained on a dataset of this form have a straightforward meaning. For example, if a logistic regression model trained to predict whether c_5 should be imputed at position 6 has a large positive co-efficient on the feature corresponding to $I_{c_1 \succ c_2}$, then the model has learned that the probability of c_5 being present at position 6 is higher when c_1 precedes c_2, and lower when the opposite is true. A large negative co-efficient has the opposite interpretation. A small co-efficient indicates that the relation of c_1 to c_2 is not very predictive of whether c_5 belongs at position 6.

Using only pairwise indicator variables as features does not allow the model to clearly express the idea that *omitting* information about a candidate is predictive of one candidate or another being ranked next on the ballot, because a value of 0 will not contribute anything to the dot-product of a given row vector (representing the feature values of a new data point) and the weight vector, and so effectively only a value intermediate between those for the different settings of the indicator variables is permitted. It also does not allow explicit expression of the idea that ranking a candidate higher or lower in absolute position (regardless of where they sit relative to another candidate) is predictive. For this reason, two additional sets of features are included. The first is a set of binary indicator variables I_{c_k}, that take on value 1 if candidate c_k is present on a ballot, and 0 otherwise, for every

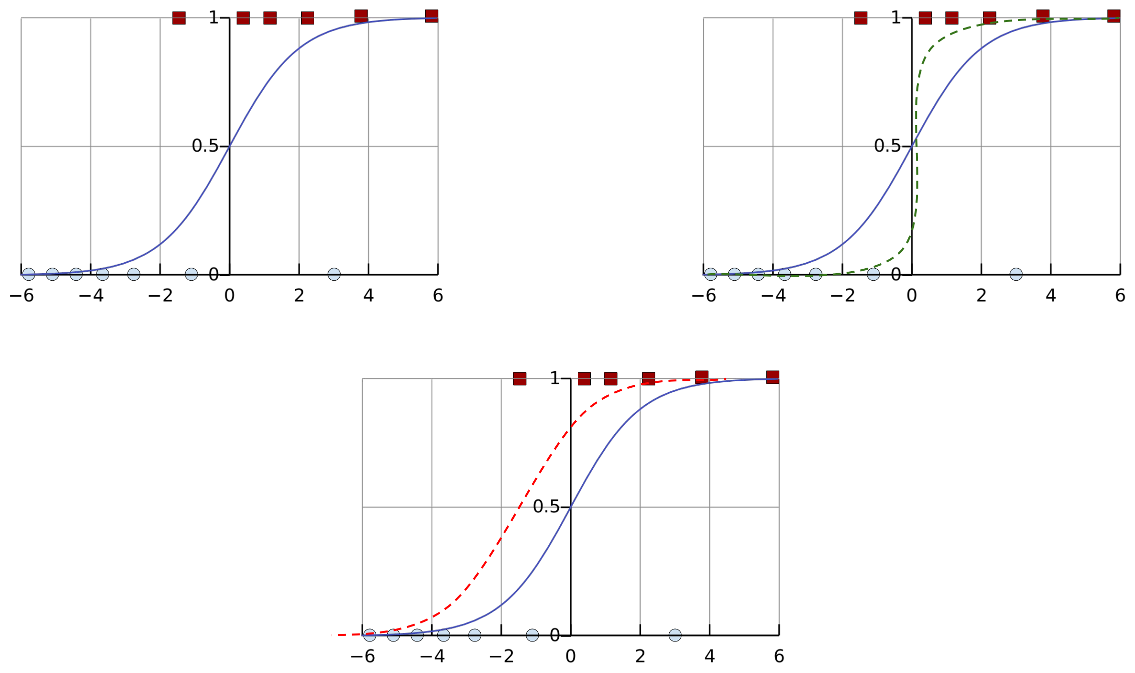


Figure 4.6: A visual depiction of logistic regression with one dependent variable. The data consist of points labelled 1 (red squares) and 0 (blue circles). The y-axis shows the label of each data point, while the x-axis shows the value of the single feature. The top-left figure depicts a reasonable logistic curve fit to this data. By changing the co-efficient of w_1, the weight applied to the single feature, the steepness of the curve can be changed (top-right figure). By changing the intercept weight w_0, the curve can be translated to the left or right (bottom figure). The optimal combination of weights minimizes the square of the distance between each point and the curve.

Feature	Value
$I_{c_1 \succ c_2}$	1
$I_{c_1 \succ c_3}$	1
$I_{c_1 \succ c_4}$	1
$I_{c_2 \succ c_3}$	1
$I_{c_2 \succ c_4}$	0
$I_{c_3 \succ c_4}$	-1

Table 4.2: A table showing a representation of the preferences $c_1 \succ c_2 \sim c_4 \succ c_3$ as a series of binary features.

candidate c_k. The second is a set of integer-valued variables Pos_{c_k}, again one for every candidate c_k. The integer-valued features take on a value equal to the position of the corresponding candidate c_k on the ballot, if c_k has been assigned a definite position.

A final set of useful features is the magnitude of the pairwise differences between the positions of candidates. If two candidates have a defined relationship on the ballot, and both have definite positions $\text{Pos}(c_j)$ and $\text{Pos}(c_k)$ respectively, then the value of feature Diff_{c_j,c_k} is $\text{Pos}(c_j) - \text{Pos}(c_k)$ (i.e. the number of positions by which c_j follows c_k on the ballot; negative if c_j precedes c_k).

To recap then, four sets of features are used in the example implementation: $I_{c_j \succ c_k}$, I_{c_j}, $\text{Pos}(c_j)$ and Diff_{c_j,c_k}. These four features each allow a logistic regression model to express different relationships between the information present on a ballot, and which candidate should be imputed next. To further facilitate the reader's understanding, the section closes with a complete example of converting several ballots to a matrix of features, extending the one from Table 4.2.

The example uses four ballots, $b_1, ..., b_4$ over four candidates, and are shown as preference relations in Table 4.3. The resulting data matrix $\mathbb{X}$ is shown in Table 4.4. The first three ballots show the same preference, but with the names of the candidates permuted. The corresponding change in the features provides some intuition for the reader about how the features change in response to changes in preferences. The final ballot encodes no information at all, and the resulting feature set can be contrasted with the other ballots. Note that missing information is always encoded with the value zero, and that the first column of the feature matrix contains a set of dummy variables used to find the intercept of the regression model.

Ballot Name	Relation
b_1	$c_1 \succ c_2 \sim c_3 \succ c_4$
b_2	$c_4 \succ c_3 \sim c_2 \succ c_1$
b_3	$c_2 \succ c_1 \sim c_4 \succ c_3$
b_4	$c_3 \sim c_2 \sim c_1 \sim c_4$

Table 4.3: A table showing four example ballots that are used in the full featurization example. The ballots each express a partial order over four candidates $c_1, ..., c_4$.

4.3.3 Model Selection

Supervised learning methods have an inherent tradeoff between *bias* and *variance*. Bias is the extent to which a learning method makes an assumption about the world that may be untrue, while variance is the extent to which a learning method can learn patterns that are not really present in the data, resulting from excessive sensitivity to the shape of the data. In logistic regression, model bias manifests from the linearity of the sigmoid. Although the sigmoid is a non-linear function, the choice of whether or not a given data point is classified as high or low comes down to whether it has a value above or below a fixed threshold[14] in the learned model. The set of points with values lying exactly at this threshold will form a line, plane, or hyperplane in the feature space, so if the true decision boundary is actually non-linear, the model's error due to bias will be large.

To reduce the model's bias, more features can be added, as illustrated in Figure 2.6 of Chapter 2. For instance, if the decision boundary is thought to be quadratic, then new features can be created from the products of pairs of existing features. In the previous subsection, this was done by adding information on pairs and distances, broadening the class of relationships that the regression can model. However, decreasing the model's bias must increase its variance, which manifests as extremely large weight values in the case of logistic regression. To more carefully control the bias-variance tradeoff, L_2 regularization is used [Tikhonov, 1963; Press et al., 2007] to enforce a bias toward simpler models. Instead of minimizing only the error of the model, weights are selected to minimize a joint function of error *and* the sum of the squared elements in the weight vector:

$$\max_{\mathbb{W}} \sum_i \log(\frac{1}{1+\exp(-x_i\mathbb{W})} I(Y_i) + (1 - \frac{1}{1+\exp(-x_i\mathbb{W})}) I(\neg Y_i)) - \lambda ||\mathbb{W}||^2 \tag{4.4}$$

[14]Often 0.5, but other values can be used if the costs of making an classification error are higher for one class than for the other.

Ballot	Dummy	I_{c_1}	I_{c_2}	I_{c_3}	I_{c_4}	Pos(c_1)	Pos(c_2)	Pos(c_3)	Pos(c_4)	$I_{c_1 \succ c_2}$	$I_{c_1 \succ c_3}$	$I_{c_1 \succ c_4}$	$I_{c_2 \succ c_3}$	$I_{c_2 \succ c_4}$	$I_{c_3 \succ c_4}$	Diff$_{c_1,c_2}$	Diff$_{c_1,c_3}$	Diff$_{c_1,c_4}$	Diff$_{c_2,c_3}$	Diff$_{c_2,c_4}$	Diff$_{c_3,c_4}$
b_1	1	1	0	0	1	1	0	0	4	1	1	1	0	1	1	0	0	-3	0	0	0
b_2	1	1	0	0	1	4	0	0	1	-1	-1	-1	0	-1	-1	0	0	3	0	0	0
b_3	1	0	1	1	0	0	1	4	0	-1	1	0	1	1	-1	0	0	0	-3	0	0
b_4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.4: Two tables showing a full featurization example, representing the ballots from Table 4.3 as a matrix of features. The lower table is a continuation of the upper, which is too wide for the page. Each column corresponds to one of the features discussed in the text (from left to right: indicators for presence on the ballot; position on the ballot; pairwise indicators; and pairwise differences). A dummy column with value 1 for every ballot is placed on the extreme left, to find the model intercept.

Compare this with Equation 4.1, which minimizes only the error. Equation 4.4 contains a parameter λ that provides direct control over the bias-variance trade-off of the regression, under a given feature set. At $\lambda = 0$, the selected model will minimize error on the training data, regardless of complexity. At $\lambda = \infty$ the selected model will set every element of the weight vector to 0, regardless of the error on the training data. In between these extremes, a value of $\lambda = 1$ requires the model to justify increased complexity with proportionate increases in performance (i.e. reducing the total error of the model's predicted values for the training data by exactly the amount that parameter values increase by). Higher values penalize complex models more severely, while lower values penalize incorrect models more severally.

Although the addition of regularization provides a means to control the bias-variance trade-off of logistic regression, it still does not provide a means of determining how to configure that tradeoff. As discussed earlier in the chapter, the No Free Lunch theorems mean no single choice will be suitable for every problem. To find reasonable values of λ, parameter tuning is conducted using cross-validation on every problem to which logistic regression is applied. In this process, the training data is partitioned into n "folds" (i.e. subsets) of equal size (throughout the thesis 10 folds are used during parameter selection, for every machine learning method that requires parameterization). The quality of a given parameter setting is estimated by learning a model under those settings on every combination of $n - 1$ of the folds, and then measuring error on the fold that the model was not trained on. If training and test data points are sampled at random from the same overall population, and each fold is selected uniformly at random from the training data, then the error measured on the holdout fold should be drawn from the same population as the error on the test data will be [Golub et al., 1979]. Repeating the experiment for each fold and taking the mean value gives a bootstrapped estimate of the generalization error under those parameter settings, though the estimate's validity is predicated on the training and test populations being similar, and the error in the estimate may be substantial. Although the estimate is by no means perfect, it does provide a reasonable heuristic for selecting model parameters, and should avoid selecting parameters that are exceptionally poor.

Note especially that the use of cross-validation for the selection of model parameters relies on the assumption that the training and test data are drawn from the sample population. In the application of social choice with incomplete information, this manifests as a very specific assumption about the relationship between true and expressed preferences. For example, imagine a classifier that predicts whether candidate c_1 is at position 3 in the true preferences of voters. The training data is the set of expressed preferences that rank some candidate at position 3, whether it is c_1 or not. The test data is the set of voters who *did not* rank a definite candidate at position 3. If there is a strong relationship between whether a voter definitely expresses a third place preference or not, and whether the voter expresses a preference for c_1 at position 3, then cross-validation will not give a reasonable estimate of the generalization capabilities of the model. This is not necessarily an idle concern. In politics, voters with more information might well have a strong preference for one party over another, which would induce a bias of exactly the sort just described. The motivating example of the Martian swarm imagined a world where richer companies are both better informed (more funds to spend gathering information), and have a greater risk tolerance (more funds to absorb losses with). The experiments in the next chapter show some evidence of this phenomenon, and a discussion there considers ways to avoid it, and its relation to a more common problem in supervised learning called *class imbalance*.

4.4 Summary

This chapter showed how the motivational example from Chapter 3 could be addressed with machine learning techniques, and discussed the inherent tradeoff between different approaches to social choice problems with incomplete information in the context of the No Free Lunch Theorems. A formal system for solving social choice problems with incomplete information by leveraging machine learning was presented, and it was shown that at least some variations of the system are theoretically equivalent to a form of social choice function. Further, such variations are subject to Arrow's Theorem and the Gibbard-Satterwaith Theorem, both important impossibility results in social choice. The chapter concluded with a more detailed example showing one possible instantiation of the system, and discussing the many design decisions that an implementation entails.

Chapter 5 continues by performing an empirical evaluation of the initial system, comparing it to existing techniques for social choice with incomplete information.

Chapter 5

Feasibility Study

Those among us who are unwilling to expose their ideas to the hazard of refutation do not take part in the scientific game.

Karl Popper [2005]

This chapter attempts to answer the question of whether real-world social choice problems are more consistent with the assumptions of the model from Chapter 4, or with the assumptions of competing models. The chapter opens with a discussion of the motivations behind the different validation studies contained within the chapter, and some empirical results characterizing the scope of the difficulty of making social choices with incomplete information, particularly the extent to which poor decisions are even possible. The latter item is evaluated by examining real-world electoral data. Section 5.2 then describes the experiment design in detail, and discusses what it can and cannot show regarding the relative performance of different models on real world data. Limitations and assumptions of the experiment design are also addressed by examining the experimental data. Following a discussion of the design, a long section covers the results of many different variations on the core experiment, and drills down to answer questions about why some models performed better or worse than others. The chapter concludes with a discussion of the results, putting them in context.

Chapter 6 continues with this theme, describing a general purpose testbed, and expanding on the results of this section, using similar experimental designs. Chapter 7 then discusses a new machine learning algorithm for this application, which is evaluated via the testbed framework, and compared with these earlier results. This chapter assumes familiarity with the content from Chapters 2-4, but is largely independent of the content in chapters following Chapter 7, and so the reader may skip ahead to that content if desired.

5.1 Motivation

As explained in Chapter 4, there is no *general* theoretical basis for preferring one method of solving social choice problems with incomplete information to another [Wolpert, 2012]. Instead, the question is whether real-world problems of this kind will tend to be more consistent with the assumptions underlying one model than another. While one can speculate about whether certain assumptions are more or less reasonable then others, we know already that even expert assessment of what is common within a given problem domain can often be incorrect. The results from voting correctly [Lau and Redlawsk, 1997; Lau et al., 2014] indicate that prior to an *empirical* assessment of the question "Are voters casting ballots that are inconsistent with their intentions?" much expert knowledge and attention was invested in the design of models that were resistant to inconsistent voter behaviours, or in criticism of models that relied on consistent behaviours. Lau and Redlawsk showed that the problem was not nearly as large in reality as had been supposed in theory. This illustrates that theoretical assumptions about the nature of voting problems can be wrong in unintuitive ways. The only way to know for sure whether an assumption (and by extension, a model) is valid or reasonable is to check it.

The results of this chapter are at base, a sort of "sanity check", though the experimental design developed here will be reused in later evaluations as well. The core idea is to demonstrate that the system outlined in the previous chapter is up to the challenge of deciding between competing preference outcomes at all, regardless of whether it does a particularly good job of it. As such, the validation will consider only a small number of datasets, a single voting rule, and a single competitor, though several different machine learning algorithms are considered. The next chapter greatly expands on these results, considering performance over many different voting rules, a larger number of datasets (including synthetic data), and a larger number of competitors.

Before preceding to the experimental design however, it is worth addressing the question of scale: how often do real world social choice problems with incomplete information offer the *potential* for making bad decisions, and how bad exactly can those decisions get? As an example, consider the four preference profiles depicted in Figure 5.1. The profiles on the left correspond to the true preferences of two different groups of voters, while the profiles on the right are the expressed preferences of the groups. In the top set of preference profiles, b

is a Condorcet winner [Black et al., 1958] in voters' true preference profile, *and* a Condorcet winner in their expressed preference profile. No fancy methods are needed to decide the election with incomplete information, as candidate b is strongly preferred in the expressed preferences, and there is no completion under which that fact would change (assuming voters have expressed top preferences, as depicted). In the bottom election, b is once again a Condorcet winner in voters' true preferences. However, in voters' expressed preferences, there is no Condorcet winner. a and b tie, with an equal number of first place votes. Voters true preferences are unknown, and there are many possible preference profiles that could have led to the expressed profile that has been observed. It is possible to construct a profile under which any of the four candidates is the winner (under **Borda Count**) that is consistent with voters' expressed preferences. Here it is possible to make arbitrarily bad decisions (i.e. to pick candidates that are arbitrarily distant from the true winner under a problem specific pseudometric, like Borda Count distance). The question then is whether real world electoral datasets look more like the top set of preference profiles (i.e. they have definite winners, or at least, voters' expressed preferences limit the scope of possible choices to candidates that must be reasonable choices), or more like the bottom set of profiles (i.e. there are few constraints on who can win, and so bad decisions are possible to make).

5.1.1 Measuring Problem Difficulty

There are a number of ways to measure the potential for poor decisions in a given set of ballots, and to assess the scope of the problem in real world data. Some natural starting points are the number of possible winners (as a proxy for the complexity of the decision), how bad the worst possible winner can be (as a baseline for poor performance), expressed as a function of the problem specific distance pseudometric. All of these approaches rely on determination of whether or not there exists a completion of the ballots under which a given candidate would win, a problem called Possible Winner Determination (**PW**). However, **PW** is known to be NP-Complete for many voting rules [Xia and Conitzer, 2008]. Whether or not a candidate is a possible winner of an election with partial ballots *cannot* be computed in polynomial time for voting systems including **Copeland**, **Borda**, and a variety of other scoring rules (including approval voting but not **Veto**), supposing $P \neq NP$. The intuition behind the difficulty of this problem is that making one candidate win may entail making many others lose simultaneously. While it is easy to construct a completion of a profile that makes a particular candidate's **Borda** score as high as possible, its score still needs to exceed those of other candidates, and ranking some of the other candidates lower may necessitate ranking others higher. Ruling out a set of preferences that balance these competing needs can be computationally challenging.

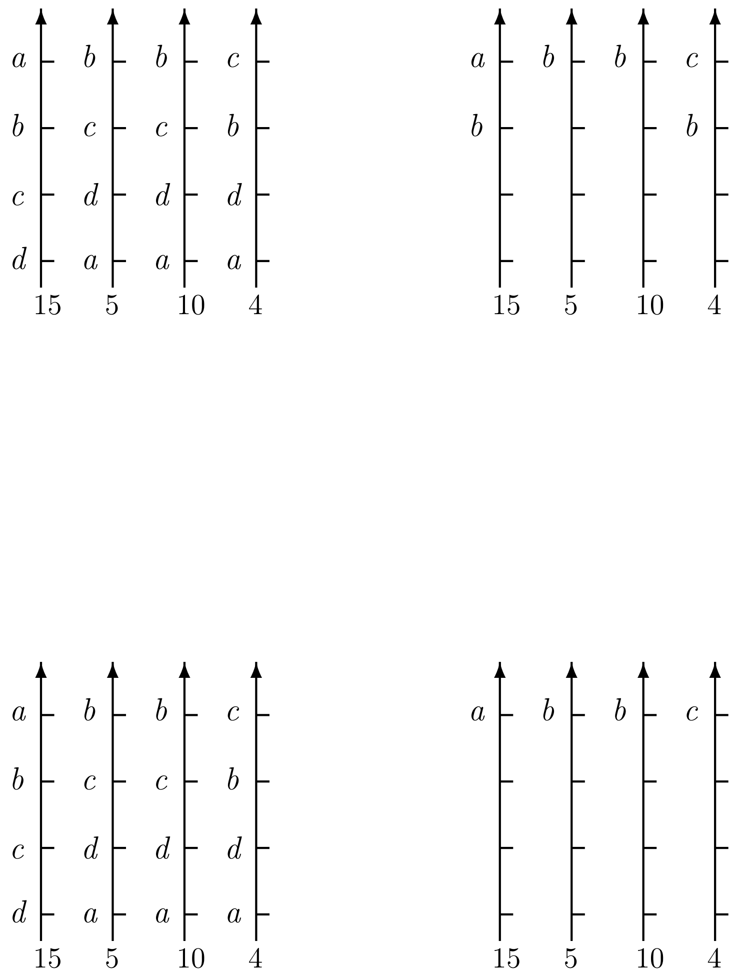


Figure 5.1: Two example voting problems with incomplete information. The top pair of preference profiles depicts a case where a Condorcet winner is present in the true preferences of voters (left), and is implied by their expressed preferences as well (right). The bottom pair of preference profiles also has a Condorcet winner in voters' true preferences (left), but the revealed preferences are consistent with many possible winners (right).

To address this, a heuristic approach to completing a worst-case profile is adopted. In the experimental setup adopted here, and in subsequent chapters, voters' true preferences are *known* as well as their expressed preferences. This allows for easy measurement of performance, but it also allows for the construction of imputations that are maximally inconsistent with voters' true preferences. Suppose that, in aggregate, voters rank c_i before c_j on average. If those voters' expressed preferences do not specify an ordering on c_i and c_j, a maximally inconsistent ordering is to impute that $c_j \succ c_i$, that is, to impute an ordering that is the opposite of the true preference of the group. If this is done for every pair of unordered candidates, on every ballot in a profile of expressed preferences, the resulting profile will tend to yield decisions that are maximally distant from the true profile. Although the distance between the aggregates of the two profiles may not be maximal (e.g. if the voting rules are non-monotonic), this allows for straightforward computation of lower bounds on several interesting measurements. Algorithm 4 describes the process of generating this approximation of the maximally inconsistent profile from a profile of partial preferences. The runtime of the algorithm as written is in $O(|B||C|^2)$, provided that the result of calling **Borda** is cached. This heuristic is used because for rules like **Copeland**, the computation of the maximally inconsistent profile is not computationally straightforward.

Given a maximally inconsistent profile $\bar{B}$, and true preferences $\succ_T$, three measurements can be computed with respect to a voting rule S_r that outputs *orderings* (i.e. rankings) over a set of candidates C, and the voting rule S_f that returns the set of top ranked candidates in the order output by S_r (i.e. the first place candidates). Different distance metrics δ correspond to the different methods of quantifying the difficulty of a voting problem with incomplete information. First, if S is used only to pick a winning candidate, what is the distance from the winner under the maximally inconsistent profile to the winner under the true preferences? As a starting point, one might consider what the rank of the selected winner is under voters' true preferences. However, a somewhat more complex expression is needed, because of the potential for ties. If two candidates are tied for first place under voters' true preferences, but not under the maximally inconsistent profile, then a penalty 0.5 candidates is used instead. Similarly, if there is a tie under the maximally inconsistent ordering, but not under the ground truth, then a distance penalty of 0.5 is applied. Formally, this results is the following distance measure:

$$\delta_w(o_1, o_2, c^*) = |\{c \in C | c\, o_1\, c^*\}| + 0.5(|\{c \in C | c \sim_{o_1} c^* \wedge c^* o_2 c\}| + |\{c \in C | c \sim_{o_2} c^* \wedge c^* o_1 c\}|) \tag{5.1}$$

where o_1 is an ordering, and c^* is a single element of that ordering. Effectively this returns

Algorithm 4 An algorithm for computing the maximally inconsistent profile $\bar{B}$ from an expressed profile B and true preference profile $\succ_T$, both over candidate set C. The output profile is consistent with B, but yields decisions that are far from $\succ_T$. The algorithm is used to empirically assess the potential for making poor group decisions on a given profile B.

```
procedure MaximallyDistantProfile($B$, $\succ_T$, $C$)
    let $\bar{B} \leftarrow B$.
    for all $b_i \in B$ do
        Let $\hat{o} = \textbf{Borda}(\succ_t)$ be the ordering of $C$ when Borda is applied to voters' true preferences.
        Let $\bar{b}_i \in \bar{B}$ be the ballot in $\bar{B}$ corresponding to $b_i$.
        for all $(c_j, c_k) \in C \times C$ do
            if $\neg(c_j\ b_i\ c_k) \wedge \neg(c_k\ b_i\ c_j)$ then /* If neither is ranked on $b_i$ */
                if $c_j\ \hat{o}\ c_k$ then
                    Set $c_k\ \bar{b}_i\ c_j$
                else
                    Set $c_j\ \bar{b}_i\ c_k$
                end if
            end if
        end for
    end for
    return $\bar{B}$
end procedure
```

the rank of c^* in o_1, along with a penalty for ranking c^* ahead of candidates in o_2 that it is tied with in o_1. A distance of zero is obtained if and only if c^* is in the winning set of o_1, and the winning sets of o_1 and o_2 are equal. The *Single Winner Distance* or **SWW** can then be defined as:

$$\text{SWW} = \frac{\sum_{c \in S_r(\bar{B})} \delta_w(S_r(\succ_T), S_r(\bar{B}), c)}{|S_r(\bar{B})|} \tag{5.2}$$

which is the average position of candidates who won the election under the maximally inconsistent profile, in the overall ranking produced by the election with voters' true preferences. If SWW is large, then this means it is possible for very low ranked candidates to win the election outright, suggesting the decision problem is difficult. If SWW is small, the problem can still be difficult however. For instance, there may be other profiles that are not as inconsistent with $\succ_T$, but that exhibit extreme bias towards a particular candidate, for instance. Effectively SWW puts an *upper* bound on the quality of the worst possible decision that could be made. Note that the average over the set of winners that is output is used because certain voting rules like **Copeland** are very prone to ties. The penalty for inconsistent ties between members of the winning set is used to ensure that this average has an appropriate meaning.

A natural extension of the SWW measurement is the multi-winner worst distance **MWW**, which is simply the Kendall τ rank correlation coefficient between the two rankings. A pseudometric wrapper is not used, so that the presented values serve as simple correlations between the sequence output by S_r under voters' true preferences, and under $\bar{B}$. Formally, this is given by

$$\text{MWW} = \tau(S_r(\succ_T), S_r(\bar{B})) \tag{5.3}$$

MWW effectively puts an upper bound on the Kendall-tau distance between the ground truth ranking and the worst possible ranking. Low (i.e. more negative) values of MWW indicate that there is great potential to make poor decisions in the problem, but yet poorer decisions may still be possible.

The final measurement considered is First Error distance (**FE**), which measures the location of the first (i.e. highest) rank at which the decisions made on the ground truth preference profile $\succ_T$ and the maximally inconsistent profile $\bar{B}$ first differ. On some problems, SWW may have value 0 (indicating that the correct winner is in some sense "easy" to pick), but MWW may have a value much less than 1.0 (indicating that the order of the other candidates may be difficult to determine). The FE distance gives some indication of *how much* of a ranking is "easy" to determine. For instance, a FE score of 5 would indicate that the order of the top 5 candidates is invariant even if voters' unexpressed preferences are imputed with values opposite those of their true preferences. Formally,

$$\delta_{\mathrm{FE}}(o_1, o_2) = \underset{i}{\operatorname{argmax}} \, \forall j < i, o_{1,j} = o_{2,j} \tag{5.4}$$

and the reported values of FE correspond to $\delta_{\mathrm{FE}}(S_r(\succ_T), S_r(\bar{B}))$.

Collectively, these three measurements will provide a means to compare the performance of various algorithms, to provide insights into which algorithms are best used in which problem domains. When examining worst case performance, these measures further provide information about how difficult a problem domain might be, or at least, how much potential for making a poor decision a given problem may provide.

5.1.2 Empirical Results

To empirically assess the difficulty of making good decisions in social choice with incomplete information, the three measurements described above were collected on a large number of problems generated using real-world election data, from ten different elections. All of the measures in question are expressed in terms of voters' true preferences. However, in data from a real world election where voters can elect to express partial ballots, true preferences are unobserved, and therefore are unavailable. Consequently, to measure the difficulty of a typical problem requires that only the ballots of voters that *did* specify a complete ordering are used. These ballots are collected, and then ablated in a way that is representative of the distribution of missing data in the original set[1]. Ballots from ten different elections were considered, drawn from the contests in three counties of the 2002 Irish national elections, and the elections of the Debian foundation during the years 2002–2012. This included 6 leadership elections, and one election to determine the logo of the organization. All elections required voters to specify a partial ranked ballot, in a top-t format (i.e. voters submit a total order over any subset of the candidates, and any un-ordered candidates are assumed to be tied at the bottom of the ranking). The raw data for each election was taken from the preflib.org repository [Mattei and Walsh, 2013]. For each dataset, 10,000 problems were generated using the method from Algorithm 7. A maximally inconsistent completion was then computed for each problem, from which the FE, SWW and MWW scores were determined under four different voting rules (**K-Approval**, **Borda**, **Copeland**, , and

[1]The precise process used is documented in the next section, as part of a general description of the experimental design that has been adopted.

	FE	SWW	MWW	$\lvert C\rvert$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	5.00 ± 0.02	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.13 ± 1.10	0.00 ± 0.01	0.96 ± 0.05	7	15.5
Debian 2006	$\mathbf{5.28} \pm 2.94$	0.00 ± 0.00	0.96 ± 0.04	8	14.8
Debian 2007	$\mathbf{7.17} \pm 1.57$	0.01 ± 0.08	$\mathbf{0.92} \pm 0.03$	9	19.1
Debian 2010	5.00 ± 0.01	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	$\mathbf{3.27} \pm 1.18$	0.00 ± 0.07	$\mathbf{0.66} \pm 0.17$	8	40.0
North 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{2.00} \pm 0.00$	$\mathbf{0.15} \pm 0.01$	12	58.5
West 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{2.03} \pm 0.17$	$\mathbf{-0.16} \pm 0.04$	9	50.8
Meath 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{5.00} \pm 0.05$	$\mathbf{-0.23} \pm 0.01$	14	66.8

Table 5.1: Table showing the mean SWW, MWW and FE ratings of 10,000 social choice with incomplete information problems generated on the basis of 10 real world datasets, computed with respect to the **K-Approval** social choice function, with $\lfloor \frac{|C|}{2} \rfloor$. The reported measurement errors are the sample standard deviations. Measurement errors are omitted when all measurements were identical.

Veto). Tables 5.1–5.4 show the mean value of each measurement over all 10,000 problems, for each of the 10 datasets. For reference, the tables also show the number of candidates in each election, and the average percentage of candidates that were left unranked on each ballot. The datasets vary from 5 to 14 candidates, and from 11-66% missing data. Recall that, for an election with $|C|$ candidates, scores for FE will range from 0 (bad; the first candidate was always incorrectly ranked) to $|C|$ (good; all candidates were always correctly ranked), scores for SWW will range from 0 (good; the correct winner was always picked) to $|C|-1$ (bad; the true last place candidate was always picked as the winner), and scores for MMW will range from -1 (bad; the relative ordering of all candidates was always reversed) to $+1$ (good; all candidates were output in the correct order).

The results in Table 5.1 summarize the mean difficulty of social choice problems with incomplete information on under the **K-Approval** voting rule, with $k = \frac{|C|}{2}$. The winner of the election under this rule is the candidate that appears most often in the top half of voters' preferences. Since voters' preferences are top-orders, and the imputation-based approach to social choice assumes voters' true preferences are consistent with their expressed preferences, on many of the ballots the maximally inconsistent imputation still amounts only to rearranging candidates on the second half of the ballot, who do not receive any

	FE	SWW	MWW	$\lvert C\rvert$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	5.00 ± 0.07	0.00 ± 0.00	1.00 ± 0.01	5	13.6
Debian 2005	6.56 ± 1.42	0.00 ± 0.04	0.99 ± 0.03	7	15.5
Debian 2006	$\mathbf{6.24} \pm 0.77$	0.00 ± 0.00	$\mathbf{0.94} \pm 0.03$	8	14.8
Debian 2007	$\mathbf{5.41} \pm 2.31$	0.01 ± 0.09	$\mathbf{0.85} \pm 0.05$	9	19.1
Debian 2010	4.96 ± 0.34	0.00 ± 0.00	1.00 ± 0.02	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	$\mathbf{2.80} \pm 0.74$	0.00 ± 0.02	$\mathbf{0.57} \pm 0.12$	8	40.0
North 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{1.55} \pm 0.50$	$\mathbf{-0.27} \pm 0.03$	12	58.5
West 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{5.01} \pm 0.08$	$\mathbf{-0.04} \pm 0.06$	9	50.8
Meath 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{11.60} \pm 0.51$	$\mathbf{-0.63} \pm 0.03$	14	66.8

Table 5.2: Table showing the mean SWW, MWW and FE ratings of 10,000 social choice with incomplete information problems generated on the basis of 10 real world datasets, computed with respect to the **Borda** social choice function. The reported measurement errors are the sample standard deviations. Measurement errors are omitted when all measurements were identical.

points, regardless of their order. Even when a voter specifies the order of fewer than $\frac{|C|}{2}$ top candidates, only a few of the candidates will be imputed into the wrong half of the ballot. Consequently, we observe that there is not much potential for making poor decisions on the smaller Debian datasets. On Debian 2006 and 2007, there are occasional errors in the ordering of low ranked (unimportant) candidates, and Debian Logo often contains errors in the middle of the overall ordering, but the correct winner is picked in essentially every one of the 10,000 sample problems. The Irish sets have much greater potential for the generation of difficult problems. The FE measure indicates that in *every* generated problem from the 10,000 sampled, it was possible to pick an incorrect winner, a finding borne out by the SWW measurements, where the true third place (or sixth place in the case of Meath) candidate could be selected as the winner in every run. On North the worst case overall ordering was virtually uncorrelated with the true ordering, as shown by the MWW scores. On West and Meath, MWW actually shows a slightly negative correlation. Overall, the table shows a trend toward harder problems on sets with greater amounts of missing data. In fact, all sets with less than 14% missing data yielded pre-determined outcomes in every problem, while all those with greater than 14% missing data had at least some problems where sub-optimal decisions were possible.

	FE	SWW	MWW	$\|C\|$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	$\mathbf{4.56} \pm 0.61$	0.01 ± 0.10	$\mathbf{0.91} \pm 0.12$	5	13.6
Debian 2005	6.50 ± 1.63	0.08 ± 0.27	0.99 ± 0.03	7	15.5
Debian 2006	$\mathbf{6.00} \pm 0.00$	0.00 ± 0.00	$\mathbf{0.93} \pm 0.00$	8	14.8
Debian 2007	$\mathbf{2.09} \pm 0.51$	0.00 ± 0.00	$\mathbf{0.70} \pm 0.03$	9	19.1
Debian 2010	$\mathbf{3.05} \pm 1.37$	0.00 ± 0.00	$\mathbf{0.82} \pm 0.14$	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	$\mathbf{3.81} \pm 1.48$	0.00 ± 0.00	$\mathbf{0.53} \pm 0.11$	8	40.0
Dublin North	$\mathbf{1.14} \pm 0.35$	$\mathbf{0.43} \pm 0.17$	$\mathbf{-0.48} \pm 0.01$	12	58.5
Dublin West	$\mathbf{1.14} \pm 0.35$	$\mathbf{6.45} \pm 0.35$	$\mathbf{-0.28} \pm 0.07$	9	50.8
Meath	$\mathbf{1.00} \pm 0.00$	$\mathbf{10.02} \pm 0.62$	$\mathbf{-0.90} \pm 0.01$	14	66.8

Table 5.3: Table showing the mean SWW, MWW and FE ratings of 10,000 social choice with incomplete information problems generated on the basis of 10 real world datasets, computed with respect to the **Copeland** social choice function. The reported measurement errors are the sample standard deviations. Measurement errors are omitted when all measurements were identical.

The results under the **Borda** social choice function, summarized in Table 5.2, are similar to those under **K-Approval**, but there is slightly more potential for making poor decisions overall. The proportion of missing information in the data still exhibits a strong relationship with the potential to make poor decisions, and the Irish West and Meath sets not permit maximally inconsistent orders that are significantly anti-correlated with the true ordering of the candidates. On Meath it is now possible to select the 12^{th} or 13^{th} placed candidates as the winner on nearly every problem set, and on West the 6th place candidate can be made to win. Although the Debian sets still typically allow only one possible winner, there is a slightly greater potential for poor orderings of the lower-ranked candidates to be generated.

Table 5.3 summarizes the same set of results for the **Copeland** voting rule. Copeland appears more open to poor decision making, with both the Debian 2003 and 2010 sets having poor performance here in the multiwinner case, unlike under the **Borda** and **K-Approval** rules. This is likely because if two unpopular candidates are unranked on a majority of ballots, it is easy to construct a completion where the more unpopular one wins the pairwise contest between them. When elections have many such unpopular candidates, then quite low ranked candidates can be made to win a large number of pairwise

	FE	SWW	MWW	$\lvert C \rvert$	% Missing
Debian 2002	4.00 ± 0.04	0.00 ± 0.01	1.00 ± 0.00	4	11.9
Debian 2003	$\mathbf{2.71} \pm 1.98$	$\mathbf{0.57} \pm 0.49$	$\mathbf{0.88} \pm 0.10$	5	13.6
Debian 2005	$\mathbf{1.01} \pm 0.09$	$\mathbf{3.57} \pm 0.67$	$\mathbf{0.35} \pm 0.12$	7	15.5
Debian 2006	$\mathbf{1.91} \pm 0.84$	$\mathbf{0.51} \pm 0.72$	$\mathbf{0.70} \pm 0.09$	8	14.8
Debian 2007	$\mathbf{1.10} \pm 0.30$	$\mathbf{1.56} \pm 0.69$	$\mathbf{0.68} \pm 0.08$	9	19.1
Debian 2010	$\mathbf{1.29} \pm 1.03$	$\mathbf{1.11} \pm 0.50$	$\mathbf{0.78} \pm 0.10$	5	11.0
Debian 2012	3.98 ± 0.18	0.00 ± 0.00	1.00 ± 0.03	4	13.2
Debian Logo	$\mathbf{1.00} \pm 0.06$	$\mathbf{5.76} \pm 1.36$	$\mathbf{-0.44} \pm 0.21$	8	40.0
North 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{11.00} \pm 0.00$	$\mathbf{-0.61} \pm 0.04$	12	58.5
West 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{8.00} \pm 0.00$	$\mathbf{-0.13} \pm 0.03$	9	50.8
Meath 2002	$\mathbf{1.00} \pm 0.00$	$\mathbf{11.00} \pm 0.00$	$\mathbf{0.03} \pm 0.03$	14	66.8

Table 5.4: Table showing the mean SWW, MWW and FE ratings of 10,000 social choice with incomplete information problems generated on the basis of 10 real world datasets, computed with respect to the **Veto** social choice function. The reported measurement errors are the sample standard deviations. Measurement errors are omitted when all measurements were identical.

contents. Many of the Debian sets now permit significant mistakes, including very low MWW correlations on Debian 2007 and 2010, as well as errors in the selected winner on Debian Logo. The three Irish sets now permit overall orderings that are almost entirely opposite the true ordering of the candidates (i.e. anti-correlated), and it is possible to select very low ranked candidates as the winner on Dublin West and Meath.

Finally, Table 5.4 summarizes the results for the **Veto** voting rule, which is in some sense a pathological case for top-ordered ballots. Veto picks the candidate that is ranked *last* on the fewest ballots. However, only about 6 or 7% of voters specified a total order on large sets like North and Meath, meaning that as much as 90% of the ballots can be assigned to candidates as far as possible from the ones voters would truly prefer. Interestingly, on the Debian 2002 and 2012 sets, this hardly seems to matter, and there is still virtually no potential to pick the incorrect candidate. Perhaps this is because most voters have ranked the true winning candidate somewhere higher in their preferences, meaning the maximally inconsistent ordering cannot assign it to the last place on any ballots. However, on most sets, there exists the possibility for very poor decisions indeed. On North and West, the last place candidate can be made to win, and the orderings are anti-correlated with the ground truth overall. Surprisingly, even the Debian 2005 set, which otherwise was nearly impossible to generated difficult decision problems from, now has a relatively low MWW correlation, and virtually every sample problem permits someone other than the true first place candidate to be chosen as the winner.

Overall these results suggest that the Debian 2002 and 2012 sets can be omitted from subsequent experiments, since there exists no voting rule under which they provide interesting problem instances. However, the other eight sets do offer various degrees of potential for making poor decisions. In particular, the three Irish sets and Debian Logo are reliably difficult across all of the four voting rules considered, and Debian 2007 is the hardest among the other sets. This finding provides some calibration for the interpretation of further experimental results: a method that fails on Debian 2003, 2006, or 2010 is interesting insofar as failure is quite difficult (though not impossible) to achieve on these sets. In contrast, perfect (or nearly perfect) performance on the Irish sets is quite a strong result, since many of the problems there offer the potential for a very large set of possible outcomes.

5.2 Experimental Design

The previous section showed that there exist real-world problems where it is possible to make decisions that are inconsistent with voters' true preferences. This is especially true when voters specify relatively few preferences, or when there are many candidates in the election, as happens when using voting rules that place great emphasis on the pairwise ordering of all candidates (e.g. **Copeland**), or on ballot positions that most voters do not assign a definite candidate to (e.g. **Veto** with top-ordered preferences). This section explains how the problems used in that evaluation were generated, and then describes a comparison between the example logistic-regression based imputation system from Chapter 4, the Minimax Regret approach [Lu and Boutilier, 2011b], and a method that imputes the order of candidates uniformly at random, approximating the maximum likelihood approach [Xia and Conitzer, 2008]. The comparison results appear in the next section.

5.2.1 Data

As explained in the previous chapter, the question of whether a particular social choice function is effective or not in practice is not one that can be answered without recourse to empirical performance on test problems that we have strong reason to believe will be representative of future applications. In the case of voting systems, there is no especially compelling reason to suppose that preferences drawn from an artificial data distribution like a Mallows model [Mallows, 1957] or Random Utility Model [Azari et al., 2012; Luce, 1959] are representative of future preference distributions. These models are theoretically compelling, and are reasonably descriptive of some domains, but the symmetry assumptions they make are quite strong. In contrast, more complex models like Mixtures of Mallows [Melia and Chen, 2010], Generalized Random Utility Models [Azari Soufiani et al., 2013] or Generalized Riffle Insertion Models [Lu and Boutilier, 2011a] can represent extremely broad classes of preferences, at least some of which we might reasonably expect to be representative of human (or agent) behaviours in future elections. However, if we were to generate data from such a model, it would need to be parametrized in some fashion, which requires either a symmetry assumption (i.e. some default set of parameters), or an empirically motivated one, based on real data (in which case the data itself ought to be used).

If claims are to be made about future performance of a method with respect to the class of problems comprising social choice with incomplete information, then the most representative problems would be sampled directly from that class. However, generally voters cast ballots only once on a given issue, and their "correct" preferences are thus neither revealed, nor recorded (nor perhaps, even known to themselves). Consequently, the author is not aware of ranked ballot data for any large scale real world election that includes both a revealed and "correct" component[2]. Without "correct" preference information the quality of any decision is in a sense subjective: it *could* be the case that voters truly prefer radically different candidates from what their expressed preferences suggest.

Consequently, a validation approach is adopted that uses problem instances generated from real world data that are similar to (and based upon) said data. Although these problems may not be identical to real world data, grounding the problems in real world data at least constrains the extent to which they can differ from the real set of problems to which they are to be applied in the future.

A large repository of real world electoral data is provided by preflib.org [Mattei and Walsh, 2013], and the evaluation conducted in this chapter uses eight of them, comprising the portion of the 10 mentioned briefly in the previous section under which there was any potential for poor decision making. Table 5.5 summarizes the properties of the different datasets. Seven of the datasets are taken from the Debian Project's elections. The Debian Project is the group of software developers responsible for the creation, maintenance, and improvement of the Debian free and open source operating system, and its elections are notable for being of a medium size (hundreds, but not thousands, of votes), and for being cast by voters operating under the **Schulze Method** voting system, a ranked ballot system

[2]Though Chapter 10 discusses the process of creating some in the future.

allowing incomplete preferences (recorded as top orders). The elections stored within preflib.org include six votes for the leader of the organization, and one to change the organization's logo, with between 4 and 9 candidates, and between 400 and 500 ballots (with the exception of the Logo election). The goal in each electoral contest was to select a single winner.

The other three datasets considered in this chapter are much larger, and are drawn from preflib's collection of Irish electoral data. Each dataset contains ballots from one constituency in Ireland's 2002 national election. Ballots were recorded as top orders, and the original elections took place under the **Single Transferable Vote** voting system[3]. The top three candidates were returned as winners in the original election, and the elections involve a very large number of votes, and between 9 and 14 candidates. In contrast to the Debian elections, they have a much higher proportion of missing data, which (as discussed in the previous section) tends to increase the potential for making poor decisions.

The experiments in this chapter used Debian 2003–2010 leadership elections, Debian Logo, and the three Irish sets. Debian 2002 and Debian 2012 were not used because there is essentially no potential for mistakes on those sets: voters preferences are very well defined (i.e. there are few missing preferences), the number of candidates is very small, and the margin of victory between candidates is large enough that there is already an effectively unique winner, as shown by the analysis in the previous section.

5.2.2 Problem Generation

As mentioned earlier, the model adopted by this thesis and outlined in Chapter 3 is difficult to evaluate directly on real-world data. Social choice with incomplete information is modelled as the process of mapping from a set of incomplete preferences to some approximation of voters' true preferences (i.e. the preferences they would express if they had better or more complete information about the candidates). However, in typically electoral datasets, the *expressed* preferences of voters are recorded, and the true preferences are neither elicited nor, perhaps, even known to the users. How then can the correctness of different approaches to resolving the election be compared? Since the true preferences of voters are generally unknown, it is not possible to compute the true outcome, and so no distance measure can be effectively applied.

However, in many electoral datasets, even if most voters do not specify a total ordering of the candidates, some subset will. For example, in the Dublin North County ballots

[3] A ranked ballot system which simulates a number of rounds of **Plurality**-like contests, dropping the weakest candidate in each round, and reassigning their supporters to other candidates.

Name	# Ballots	$|C|$	% Missing	Type
Debian 2002	475	4	11.9	Leadership
Debian 2003	488	5	13.6	Leadership
Debian 2005	504	7	15.5	Leadership
Debian 2006	421	8	14.8	Leadership
Debian 2007	482	9	19.1	Leadership
Debian 2010	436	5	11.0	Leadership
Debian 2012	403	4	13.2	Leadership
Debian Logo	134	8	40.0	Logo (Single Winner)
Dublin North	43,942	12	58.5	Multiwinner
Dublin West	29,988	9	50.8	Multiwinner
Meath	64,081	14	66.8	Multiwinner

Table 5.5: Summary of the datasets used in the initial evaluation of the system, including the number of ballots, number of candidates, percentage of missing data, and type of election.

from the 2002 Irish national election, about 7% of voters ranked all 12 candidates. Since there are tens of thousands of ballots, this actually yields a reasonably sized set of ballots that attempted to rank all candidates. Lau and Redlawsk [Lau and Redlawsk, 1997; Lau et al., 2014] find that information voters provide is correct about 70% of the time, and it is not unreasonable to speculate that in elections where voters are permitted to omit candidates from their rankings, the voters who *do* rank every candidate are (on average) even better informed than a typical participant. On this basis, it is reasonable to think that the voters who did rank every candidate have expressed some approximation of their true preferences. Considering just this subset of the ballots from a large collection of electoral data then yields a problem with a well defined ground truth (i.e. the true preferences of voters are known), but no missing information (since the expressed preferences of every voter in this subset are a complete ordering of the candidates). This set is called $\hat{\succ}$.

The original set of ballots from an election with partial information may not contain clear information about the true preferences of most voters, but they do contain information about the *distribution of missing preferences*. For example, it is easy to answer questions about the average number of candidates ranked by each voter, or whether certain candidates are more or less likely to be ranked than others. By answering these questions, an empirically-parametrized model of the missing information, called N, can be constructed from a profile of ballots. Provided that the model can be used to *predict* missing preferences, it can then be applied to the subset of ballots for which voters' true preferences are known, to produce an ablated set of "expressed" preferences. These preferences are not the ballots voters actually cast, but, provided the model of missing information is accurate, they will be representative of how *typical* voters expressed information. The ablated version of $\hat{\succ}$ is called $\hat{B}$. Together $\hat{\succ}$ and $\hat{B}$ form a social choice problem with incomplete information, exactly as mandated by the model adopted in this thesis.

The ablation model N can be constructed in different ways, and the extent to which it generated realistic ablations of the data have direct implications on the generalizability of the results observed on these problems to true real-world problems. Since the datasets of interest are typically comprised of top-ordered ballots, the problem can be simplified to finding a model that selects the point at which a ballot should be truncated. Perhaps the simplest such model is to assume that the probability of specifying one more preference is independent of the number of preferences specified already and of which candidates were specified. This model needs just one parameter, p, the probability of specifying one additional preference, and the probability of specifying y preferences would be proportionate to p^y. However, this model does not agree very well with the empirical distribution of the truncation points on top-ordered ballots, at least in the datasets considered. In particular, while the model prescribes a smooth exponential distribution, the observed distributions on sets with many alternatives is actually shaped like a sigmoid, wherein voters are unlikely to cast a ballot with more than a few alternatives on it at all, but once they've ranked most alternatives, they were much more likely to rank the remainder than this model would predict. Figure 5.2 illustrates this difference.

Instead, a model could be learned with independent parameters for the probability of the truncation point being at each location. This distribution would exactly match the observed distribution with respect to the total amount of missingness per ballot, but would not capture relationships between the amount of missing information and preferences for certain candidates. For example, if voters that supported more radical candidates were more likely to rank the entire set of candidates in the original set of ballots B, such a model would not ensure this relationship was preserved in that ablated ballots $\hat{B}$. The initial experiments conducted in the thesis use this model, but Chapter 7 considers more complex models, which can depend on the choice of candidate. Empirically, there was good agreement between the distribution of expressed preferences for each candidate on the original (B) and ablated ($\hat{B}$) ballots when this method was used however. Algorithm 5 explains the process for learning ablation models of this kind, and runs in $O(|C||B|)$ time, assuming that the i^{th} position on a ballot can be queried in constant time, which can be achieved by storing ballots as hashmaps. Algorithm 6 shows how an existing model can be used to stochastically ablate a given (complete) ballot. Each sub-model n_i of the learned

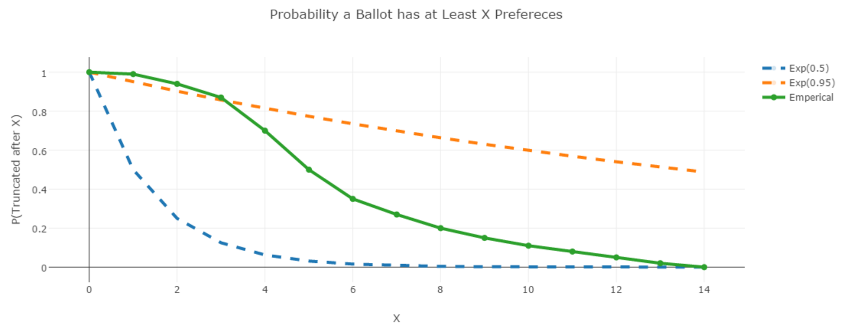


Figure 5.2: Two probability density functions for exponentially distributed preference truncation points in top-orders are shown with dashed lines. The observed density function for the preference truncation point in top-orders is shown as a solid line. Neither of the exponential distributions are able to fit the observed rates at both the top and bottom effectively.

ablation model encodes $n_i = P(|b_j| \geq i + 1 \mid |b_j| > i)$. To sample a truncation point from this model, one starts with the truncation point at location 0 (i.e. all preferences are unspecified), and simply generates a uniformly distributed random number u over the unit interval. If $u < n_1$, then the truncation point is incremented, and a new uniformly distributed random number is generated, which is compared to n_2, and so on. Eventually, either $u > m_i$ for some i, or there are no more sub-models. The process stops there, and outputs a ballot that is identical to b_j prior to the truncation point, but for every pair of candidates c_k, c_l to the right of the truncation point, the output ballot ranks $c_k \sim c_l$. This algorithm runs in $O(|C|^2)$ time.

Algorithm 7 details the process of constructing a sample problem using the approach outlined above. Given a set of incomplete expressed preferences B over a set of candidates C, it first learns an ablation model N from B. B is then reduced to $\hat{\succ}$ by filtering out all ballots that are incomplete. $\hat{B}$ is then generated from $\hat{\succ}$ by applying the ablation model N to each ballot in turn, and generating a partial preference that is consistent with said ballot. The total runtime is in $O(|B||C|^2)$. Note that although the applications of the algorithms considered in this chapter and Chapter 6 are concerned entirely with top-ordered ballots, the algorithm is general provided that an ablation model suitable for general partially ordered ballots is provided.

Algorithm 5 An algorithm for learning an ablation model from a set of top-ordered ballots B over a set of candidates C. The model is composed of sub-models n_i. The parameter stored in model n_i is $P(|b_j| \geq i+1 \mid |b_j| > i)$. If the truncation point is exponentially distributed, than every submodel should learn (approximately) the same parameter.

```
procedure LearnTopOrderAblationModel($B$, $C$)
    let $N \leftarrow \emptyset$
    let $B_0 \leftarrow B$
    for $1 \leq i \leq |C|$ do
        let $B_i$ be the subset of $B$ with a specified $i^{th}$ preference
        let $n_i \leftarrow \frac{|B_i|}{|B_0|}$
        let $N \leftarrow \{N, n_i\}$
    end for
    return $N$
end procedure
```

5.2.3 The Imputation Based Approach

The imputation-based approach selected for initial validation is very similar to the one described as an initial implementation near the end of Chapter 4. However, since the data take the form of top orders and not arbitrary partial orders, some alterations are necessary in the selection of features.

The features used were the four sets described in the initial implementation: indicator variables showing whether or not a candidate was present on the ballot; the numeric location of each candidate; indicator variables showing whether one candidate comes before or after another; and the distance between the positions of any two candidates. However, when a feature set was generated for a classifier that was intended to impute the missing preferences at position i, only information about each voter's top $i-1$ ranked candidates was used to generate these features. Since in this problem any ballot that needs its i^{th} preference imputed has no preferences beyond the $i-1^{th}$ (because ballots are top orders), any method that relies on information from preferences beyond the i^{th} is prone to failure. As an example, a ballot $a \succ b \succ c \succ d \succ e$ when used as input to a learner trying to infer voters' second preferences would be replaced with the ballot $a \succ b \succ c \sim d \sim e$.

Additionally, Information Gain-based feature selection was used to reduce the features to smaller (and more predictive) subset. In this approach, the information gain of each feature with respect to the label set Y is computed. The top 30 features were used in each run. To reduce the overfitting of the model L1 Regularization was used with a constant

Algorithm 6 An algorithm for learning applying an ablation model N to a totally ordered ballot b over candidates C, producing ablated top-ordered ballot $\hat{b}$. The algorithm first samples a truncation point t, and then returns a copy of b with all candidates past position t assigned indeterminate orders relative to each other.

```
procedure TopOrderAblation($N$, $b$, $C$)
    let $\hat{b} \leftarrow \emptyset$
    let $t \leftarrow 0$ be the truncation point.
    for $1 \leq i \leq |C|$ do
        if unif(0,1) $< n_i$ then $t \leftarrow t + 1$
        else
            break
        end if
    end for
    for all $(c_i, c_j\ inC \times C$ do
        if $|c \in C|c\ b\ c_i| \geq t\ \wedge |c \in C|c\ b\ c_i| \geq t$ then
            Set $c_i \sim c_j$ in $\hat{b}$.
        else
            if $c_i\ b\ c_j$ then
                $c_i \hat{b} c_j$
            else
                $c_j\ \hat{b}\ c_i$
            end if
        end if
    end for
     return $m$
end procedure
```

Algorithm 7 An algorithm for generating problem instances from real world datasets. Given ballots B over candidates C, and ablation model learning algorithm N, this algorithm first trains an ablation model n over B, then reduces B to the subset of complete ballots, which becomes the true preference of voters in this problem, $\hat{\succ}$. Each ballot in B is then ablated using n, to produce a corresponding ballot in $\hat{B}$, the expressed preferences of voters in this problem.

```
procedure ImputationModel($N$, $B$, $C$)
    Let $n \leftarrow N(B)$
    Let $B \leftarrow B \setminus B_{ic}$
    Let $\hat{B} \leftarrow \emptyset$
    Let $\hat{\succ} \leftarrow \emptyset$
    for all $b_j \in B$ do
        $\hat{\succ}_i \leftarrow b_j$
        $\hat{B} \leftarrow \hat{B} \cup TopOrderAblation(n, b_j, C)$
    end for
     return $(\hat{\succ}, \hat{b})$
end procedure
```

value of $\lambda = 10$. This value was selected on the basis of small scale experiments, rather than full cross validation on each problem, to keep experimental runtimes small. Using the same constant value across all datasets also ensures that model selection will favour less detailed (smoother) models on sets with fewer examples, and more detailed models when there is more data to justify their refinement.

5.2.4 Competitors

The initial validation was conducted by comparing against the **Minimax Regret** algorithm [Lu and Boutilier, 2011b] or MMR. As described in Chapter 3, MMR operates by computing the profile of completed ballots under which each candidate receives the lowest possible ranking under whatever voting rule is being used. This is the *maximum regret* that could be experienced on the part of the electorate if a given candidate was elected. If the election is merely to select a winner, than the candidate with the minimum maximum regret is declared the winner. If the election is to order the set of candidates, then they are output in order of ascending maximum regret. More formally, MMR for the single-winner case was defined as:

$$\text{MMR}(B, C) = \operatorname*{argmin}_{c \in C} \operatorname*{argmax}_{\pi \in \Pi} \delta_\pi(S(\pi), c)$$

where δ_π was the distance from c to the winner of the election under π, in the ordering produced by deciding the election using the extension π.

The set of all possible completions of ballots is very large, and so enumerating it to find the completion that maximizes regret for a given candidate is infeasible. Fortunately, under many voting systems, it is possible to find this profile much faster. Lu and Boutilier [2011b] provide a simple approach to computing the maximum amount that a candidate c_1's score can exceed that of a candidate c_2 under any positional scoring rule. This includes rules like **Borda**, **Veto** and **K-approval**. The algorithm is based on the idea that for each ballot, one of three possible cases holds, and in each, there is a straightforward way to maximize the margin by which candidate c_1 defeats candidate c_2. Each partial ballot contains one of the situations depicted in Figure 5.3. In the figure, the rounded boxes show sets of candidates, and their relation to the two candidates of interests a and w. For example, in the leftmost case, A is the set of all candidates that appear before candidate a, D is the set of all candidates that appear after candidate a, but have an undefined order relative to candidate w. B is the set of all candidates appearing between a and w.

Under the **Borda** and **Veto** rules, the completion of a ballot is readily determined by categorizing it according In the leftmost case of Figure 5.3, the ballot used will be $A \succ a \succ U \succ D \succ B \succ C \succ w \succ X$. In the rightmost case, a and w are incomparable, and the order harming w most is $A \succ E \succ a \succ U \succ C \succ D \succ w \succ X \succ F$. In the middle case, where w precedes a, the order is $A \succ w \succ B \succ E \succ a \succ F \succ U \succ W$. The **K-Approval** rule is slightly more involved (because the sizes of the sets matter), but can still be computed very quickly by considering at most $|C|$ possible sizes for the set of candidates to place between a and w.

The Copeland rule is more complex, and for general partial orders it is NP-Complete [Lu and Boutilier, 2011b; Xia and Conitzer, 2008] to find the maximum regret for a given candidate. However, for top orders, it is possible to compute maximum regret efficiently, via a new approach, presented here as Algorithm 8. The algorithm operates on the idea that, if some candidate c_1 has lost to some other candidate c_2 by the widest possible margin under Copeland, than c_1 has lost the largest number of contests possible, and c_2 has won the largest number possible. Fortunately, in a top-order, these two criteria are always achieved by the same completions of each ballot. To show this, consider several cases:

Case 1: A ballot b_j ranks both c_1 and c_2, and is a top-order ranking k candidates in all. Regardless of how b_j is completed, c_1's position is fixed, and c_2's position is also fixed, relative to both every ranked and every unranked candidate (since the unranked candidates come after every ranked candidate). Therefore all completions of b_j have the same effect on the pairwise contests between c_1 and any other candidate c_i, and between c_2 and c_i.

Case 2: A ballot b_j ranks c_1, but not c_2 and is a top-order ranking k candidates in all. By definition of a top order, c_1 already wins pairwise contests on this ballot with every unranked candidate. Therefore, all rankings of the unranked candidates will have identical effects on the Copeland score of c_1. If c_2 is ranked first among the unranked candidates, it wins the largest number of pairwise contests possible on this ballot. Therefore, any ranking that puts c_2 as the $k+1^{th}$ candidate will maximize the Copeland score of c_2, and minimize the Copeland score of c_1.

Case 3: A ballot b_j ranks c_2, but not c_1 and is a top-order ranking k candidates in all. By definition of a top order, c_2 already wins pairwise contests on this ballot with every unranked candidate. Therefore, all rankings of the unranked candidates will have identical effects on the Copeland score of c_2. If c_1 is ranked last among the unranked candidates, it loses the largest number of pairwise contests possible on this ballot. Therefore, any ranking that puts c_1 as the $|C|^{th}$ candidate will maximize the Copeland score of c_2, and minimize the Copeland score of c_1.

Case 4: A ballot b_j ranks neither c_2 nor c_1, and is a top-order ranking k candidates in all. By definition of a top order, both candidates have already lost pairwise contests on this ballot with every ranked candidate. Ranking c_2 first, and c_1 last among the unranked candidates will then maximize the number of pairwise contests won by c_2 on b_j, and lost by c_1 on b_j.

Since Algorithm 8 always ranks c_1 last if possible, and always ranks c_2 as high as possible, it is therefore maximizing the number of pairwise contests won by c_2 on each ballot, and minimizing the number won by c_1. Over the entire profile, this will produce the largest number of pairwise victories possible for c_2, and the smallest number for c_1. This is by definition the maximum regret of picking candidate c_1 instead of c_2 under the Copeland rule. One (or more) of the candidates in $C \setminus c_1$ will have largest regret, and this is the maximum regret possible for picking c_1 at all. The algorithm performs a constant time operation on each ballot for every pair of candidates, so it runs in $O(|C|^2 N)$ time.

In addition to MMR, a comparison is made against a competitor that was constructed specifically for the purpose of these experiments, a "random" imputation method, that imputes each ballot with a suffix selected uniformly at random from the set of possible suffixes (i.e. the suffixes that do not contain any candidates that were already ranked on the ballot). This technique is similar in spirit to the maximum likelihood approach to voting with partial ballots [Xia and Conitzer, 2011] introduced in Chapter 3 because,

Algorithm 8 An algorithm for computing the maximum regret of a candidate c^* under the **Copeland** voting rule, with a set of top-ordered ballots B over a set of candidates C. The algorithm is facilitated by two helper algorithms, which respectively answer the question of whether or not a candidate c_1 must win a pairwise contest with candidate c_2 under any completion of the ballots B, and whether there exists a completion under which c_1 beats c_2.

```
procedure CouldWin($c_1$,$c_2$, $B$)
    Let counter=0
    for all $b_j \in B$ do
        if $c_1$ $b_j$ $c_2$ or $\neg$($c_1$ $b_j$ $c_2$ or $c_2$ $b_j$ $c_1$) then
            counter = counter + 1
        end if
    end for
    return $I(counter > \frac{|B|}{2})$
end procedure

procedure MustWin($c_1$,$c_2$, $B$)
    Let counter=0
    for all $b_j \in B$ do
        if $c_1$ $b_j$ $c_2$ then
            counter = counter + 1
        end if
    end for
    return $I(counter > \frac{|B|}{2})$
end procedure

procedure MaxRegretCopeland($c^*$, $B$, $C$)
    Let MaxRegret = 0 and WorstOpponent = null
    for all $c_1 \in C \setminus c^*$ do
        Regret = CouldWin($c_1$,$c^*$, $B$)
        for all $c_2 \in C \setminus \{c^*, c_1\}$ do
            Regret = Regret + CouldWin($c_1$, $c_2$, $B$) - MustWin($c^*$, $c_2$, $B$)
        end for
        if WorstOpponent == null or Regret > MaxRegret then
            MaxRegret = Regret and WorstOpponent = $c_1$
        end if
    end for
    return MaxRegret
end procedure
```

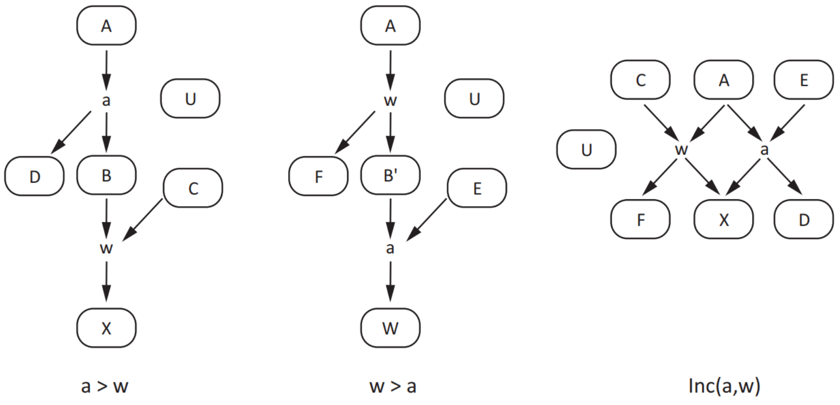


Figure 5.3: The three cases to consider when maximizing the regret for picking one candidate (a) over another (w), reproduced from [Lu and Boutilier, 2011b]. In the first case (left), a comes before w. In the second (middle), a comes after w, and in the third case, a and w are incomparable. The various bubbles represent sets of candidates, and under a given voting rule, these sets must be ordered differently in the regret maximizing completion. See text in this chapter for an example.

since all completions are equally likely under the maximum likelihood model, the decisions reached via random imputations will be similar in aggregate (but much easier to compute).

5.3 Results

Results were produced by generating a number problem instances using the approach described in the previous section. For the smaller sets (All Debian sets and Dublin West), 1,600 problems were generated. 550 problems were used for Dublin North, and 100 for Meath. 1,600 runs were selected because this was the number required to obtain 95% confidence intervals with a width of ±0.1 of a standard deviation, given the large number of individual hypotheses that were to be assessed. The two larger sets used a smaller number of problems because some of the methods under evaluation have runtimes that are quadratic in the number of candidates. For instance, 100 runs on the Meath dataset takes longer than 1,600 runs on each of the Debian sets. The exact numbers used for the two larger sets were selected on the basis of being the largest round number that could be obtained in approximately 24 hours of computation.

For each problem instance, the imputation based approach (using L1-regularized logistic regression, and the feature set described in the previous section) was applied to the expressed preferences, followed by the application of each of the four voting rules ($\frac{|C|}{2}$-**approval**, **Borda**, **Copeland**, and **Veto**), resulting in a total ordering of the alternatives. MMR was separately applied to the expressed preferences to obtain an ordering of the candidates in terms of their minimax regret under each of the voter rules. The three performance measures described in the first section of this chapter were then computed for MMR and the imputation based approach across each of the problem instances. The mean performance measures for each of the two methods, on each of the four voting rules, are then summarized in Tables 5.6 to 5.13, along with information about each dataset for reference. In each case, the mean value along with the standard deviation of each method is reported. Figures 5.4- 5.15 provide a visual comparison between MMR and the imputation-based approach across the datasets under each of the different measures and each of the different voting rules.

5.3.1 Results Overview

The results in this section are presented in terms of three measurements, all similar to the distances used in the worst-case method's evaluation earlier in this chapter. The Single Winner Error measurement (abbreviated Single Winner in the tables) corresponds to the number of candidates that precede the winner selected by a given method in the ground truth ordering. Like the SWW measurement, if there are multiple candidates tied for the winning position, then it takes the average score over all of them. For example, if MMR picked the candidate that finished 2nd under voters' true preferences, it would have a Single Winner Error of 1 (one candidate precedes its choice in the ground truth ordering). If it picked the 4th place candidate, it would have a Single Winner error of 3 (since three candidates precede its choice in the ground truth ordering). If there was a tie under voters' true preferences, and the winning set contained two candidates, then if MMR picked one of the two as a unique winner, and ranked the other in second place, it would have a distance of 0.5 (see Equation 5.2 above).

In addition to the Single Winner Error measurement, it is important to know how close a given method was to recovering the entire ordering of a set of candidates. Under voters' true preferences there will exist candidates that finished second, third, fourth, and so on. In some contests the entire ordering is relevant (for example, if the vote is to prioritize the order in which mining sites are to be prospected, even though all will eventually be explored). In other votes there may be interest in only the top k candidates overall. For instance, political elections that use multi-member districts, like the Irish system, would require an accurate ordering of the top candidates, but possibly not the remainder. To capture these two distinct needs, two additional measurements are taken. The τ measurement is the Kendall correlation [Kendall, 1938] between the order a voting rule S returns when given voters' true preferences, and the order returned by a system of interest when provide with voters' expressed preferences. This measure ranges from -1 to 1, and is proportionate to the number of pairwise orderings that agree between the two rankings. For instance, if one order ranks $c_1 \succ c_2$, but the other ranks $c_2 \succ c_1$, then the rankings have lower Kendall correlation than if they agree on the relative ordering of those two candidates. To capture the idea that higher ranked candidates may be more important, the First Error measurement corresponds to the location of the first incorrectly ranked candidate in the ordering output by a given method. For example, if a method picks not only the correct winner, but also the correct second and third place finishers (i.e. the second and third place finishers in the ordering obtained from voters' true preferences match those output by the method when supplied with voters' expressed preferences), it has a First Error score of at least 4 (the first error is not at position 1, 2, or 3).

The presentation of the results begins with a discussion of the overall performance of each method, and a summary of the results in tabular form to facilitate reference or discussion of the exact numeric values. Although the performance of the imputation-based approach is contrasted with the two comparison methods and the worst-case results, the results are compared only qualitatively. Following the presentation of all results, a quantitative comparison is performed, including statistical analysis and facilitated by graphical depictions of the results from earlier tables. Note that the redundancy in the presentation of the results (i.e. both graphical and tabular presentations) is to allow the reader to compare the approaches visually (i.e. with graphs), while retaining the precise details of the results (i.e. in the tables).

Tables 5.6 to 5.9 summarize the results for the imputation based approach on the four voting rules considered. Under the **K-Approval** rule, the system uniformly selects the correct winner, an encouraging result. Results under the Kendall Correlation (τ) are also promising, with high correlations across the board. The First Error measure reveals that when the method makes a mistake, it seems to do so consistently. For instance, on Debian 2007 and Dublin West, the mistakes are consistently located at the 2^{nd} position. However, mistakes are rare on the whole.

Results under the **Borda** voting rule (Table 5.7) are broadly similar. Although the correct single winner is recovered nearly all the time, on Debian 2007, 17% of runs picked the true second place candidate as the winner. Despite this, Kendall Correlations were general higher than under **K-Approval**, and the location of the first error in the ranking was also improved, except for the Meath dataset where the incorrect candidate was routinely selected for second place (though the high value of τ indicates that most of the correct ordering was still recovered there). **Borda** on the whole appears to be slightly easier for the imputation based approach than **K-Approval**, despite being slightly harder according to the worst-case method. Possibly this is because errors around the $\frac{|C|}{2}^{th}$ candidate have a sharp cost under **K-Approval**, while costs are uniform throughout for slight errors in a candidate's position under **Borda**.

The **Copeland** rule was found via the worst-case imputation method to be harder still than **Borda**. Results for the imputation-based approach are summarized in Table 5.8. Overall Single Winner selection is still extremely strong. A consistent error is made on the Meath dataset, where either the true third or fourth place (of 14) candidates is selected routinely. Performance on Dublin West is quite good however. The performance in terms of the entire ordering is also strong. The four harder sets (Logo, North, West, and Meath) all have lower values of τ than under the earlier rules, but the orderings are still strongly correlated with the ground truth ranking. First Error measurements support this idea, as the first four candidates on Dublin North and West are usually recovered in the correct order. The bulk of the correct ordering was also recovered on Debian Logo . Although performance under the **Copeland** rule is clearly lower than under **K-Approval** or **Borda**, it is still far superior to the worst-case performance measured earlier in the chapter.

Last to consider in the results for the imputation based approach is the **Veto** rule. This is a pathological voting rule when used in conjunction with logistic regression or another classifier, because to correctly impute the final position on a ballot, a chained classifier must correctly impute *all* preceding positions. In contrast, under Borda, a minor error in the position of a single candidate is less concerning, since it will have only a small impact on that candidate's vote total. The results for the imputation based approach using a chained classifier based on logistic regression are summarized in Table 5.9. As expected, performance is markedly worst, but on the whole still not unreasonable. On the Debian sets, the correct single winner is obtained much of the time, and even when it is not, the true second or third place candidate is usually selected (indicated by the low standard deviation in the measured Single Winner Error). Kendall Correlations are also very high across all of these sets except Debian Logo, and although the correct winner is sometimes not picked, the First Error results (with high standard deviation) show that when the correct winner is picked, most of the remainder of the order is recovered as well. Performance on the Irish sets was less good. While the correct winner is usually recovered on Meath, as well as the bulk of the overall ranking ($\tau = 0.62$), performance on the North and West sets is poor, and the true 5^{th} place candidate is usually picked as the winner. Despite results being less than ideal, they are still quite good considering the pathological nature of this combination of voting rule and learning algorithm.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	7.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	7	15.5
Debian 2006	6.55 ± 2.51	0.00 ± 0.00	0.98 ± 0.04	8	14.8
Debian 2007	2.09 ± 0.73	0.00 ± 0.00	0.89 ± 0.04	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian Logo	2.16 ± 0.64	0.00 ± 0.00	0.77 ± 0.10	8	40.0
Dublin North	5.77 ± 2.03	0.00 ± 0.00	0.94 ± 0.03	12	58.5
Dblin West	2.07 ± 0.25	0.00 ± 0.00	0.84 ± 0.01	9	50.8
Meath	3.00 ± 0.00	0.00 ± 0.00	0.74 ± 0.00	14	66.8

Table 5.6: Table showing the mean SWW, MWW, and FE measures for the instantiation of the imputation-based approach using logistic regression on the **K-Approval** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.93 ± 0.59	0.00 ± 0.00	1.00 ± 0.01	7	15.5
Debian 2006	7.94 ± 0.36	0.00 ± 0.00	1.00 ± 0.01	8	14.8
Debian 2007	7.55 ± 3.07	0.17 ± 0.37	0.99 ± 0.02	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian Logo	5.05 ± 2.75	0.00 ± 0.00	0.93 ± 0.06	8	40.0
Dublin North	4.25 ± 0.43	0.00 ± 0.00	0.83 ± 0.01	12	58.5
Dublin West 2	2.00 ± 0.00	0.00 ± 0.00	0.89 ± 0.00	9	50.8
Meath	2.00 ± 0.00	0.00 ± 0.00	0.75 ± 0.01	14	66.8

Table 5.7: Table showing the mean SWW, MWW, and FE measures for the instantiation of the imputation-based approach using logistic regression on the **Borda** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	4.94 ± 0.49	0.01 ± 0.12	0.99 ± 0.07	5	13.6
Debian 2005	7.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	7	15.5
Debian 2006	6.35 ± 0.76	0.00 ± 0.00	0.94 ± 0.03	8	14.8
Debian 2007	9.00 ± 0.00	0.00 ± 0.00	0.90 ± 0.08	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian Logo	5.18 ± 1.36	0.00 ± 0.00	0.75 ± 0.11	8	40.0
Dublin North	4.01 ± 0.07	0.00 ± 0.00	0.85 ± 0.01	12	58.5
Dublin West	3.85 ± 1.88	0.82 ± 0.38	0.81 ± 0.06	9	50.8
Meath	1.00 ± 0.00	3.54 ± 0.32	0.74 ± 0.02	14	66.8

Table 5.8: Table showing the mean SWW, MWW, and FE measures for the instantiation of the imputation-based approach using logistic regression on the **Copeland** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	2.24 ± 1.85	0.58 ± 0.43	0.86 ± 0.09	5	13.6
Debian 2005	2.98 ± 2.16	0.25 ± 0.49	0.88 ± 0.09	7	15.5
Debian 2006	2.18 ± 1.55	0.43 ± 0.60	0.83 ± 0.09	8	14.8
Debian 2007	3.45 ± 2.81	0.31 ± 0.50	0.91 ± 0.06	9	19.1
Debian 2010	4.59 ± 1.21	0.06 ± 0.19	0.98 ± 0.06	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	2.00 ± 0.93	0.15 ± 0.25	0.53 ± 0.12	8	40.0
Dublin North	1.00 ± 0.00	3.92 ± 0.51	0.28 ± 0.03	12	58.5
Dublin West	1.00 ± 0.00	3.78 ± 0.53	0.54 ± 0.06	9	50.8
Meath	1.87 ± 0.34	0.11 ± 0.30	0.62 ± 0.02	14	66.8

Table 5.9: Table showing the mean SWW, MWW, and FE measures for the instantiation of the imputation-based approach using logistic regression on the **Veto** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

Tables 5.10-5.13 similarly summarize the performance of the Minimax Regret approach across the same three performance measures: Single Winner Error (denoted Single Winner), Kendall Correlation (denoted τ), and First Error Location (denoted First Error). Table 5.10 shows the results on under the **K-Approval** voting rule. Immediately noticeable is that while the imputation-based approach selected the correct winner uniformly under this rule, MMR makes occasional mistakes on the Debian 2007 and Dublin West sets, and consistent mistakes on the Meath set. On Meath, MMR consistently picks the true second place candidate as the winner. However, Kendall Correlation is actually slightly higher on many of the sets, and the First Error measure reveals that when MMR manages to pick the correct winner, it usually also recovers the bulk of the sequence[4].

A similar pattern is found under the **Borda** rule, with results summarized in Table 5.11. MMR makes consistent errors in selecting the winner on both the Dublin West and Meath sets under this rule, and actually selects the true third place candidate as the winner routinely on Dublin West. The Kendall correlations for MMR are quite strong under **Borda**, and as with **K-Approval**, when MMR picks the correct winner, it usually recovers the bulk of the correct sequence too. However, since the imputation-based method's performance under **Borda** was actually slightly better than under **K-Approval**, the two methods score similarly here in terms of their Kendall correlations.

In contrast to the first two voting rules, where performance for MMR and the imputation-based method were similar, the results under the **Copeland** rule show a marked divergence. Under **Copeland** MMR's single winner performance is comparable to that of the imputation-based approach, apart from doing somewhat better on Meath. However, the Kendall correlation results show the limitations of MMR's worst-case approach, with extremely low values on Meath and Dublin North, the elections with the largest amount of missing data, and the largest number of candidates (i.e. the hardest sets). This is probably because of the large number of "fringe" candidates in these elections, that were unranked on many ballots. Since the order of these candidates is indeterminate with respect to each other, MMR declares a large multi-way tie. However, such a tie is actually quite unlikely, leading to the very low correlations with the ground truth ranking. Although returning an outcome of "Undecidable" is not unreasonable, it does indicate the importance of further elicitation to MMR's strategy. In some application domains, eliciting further preferences from the electorate is not possible. For example, when a robotic swarm submits its ballots, the robots may have no easy way of obtaining additional information from their owners (the owners may not know themselves, or the robots might be out of direct communication). In a non-secret ballot vote, some voters may be unwilling to reveal complete information,

[4]Some of the performance differences here are not statistically significant, and are intended to provide only a qualitative view. A more statistically rigorous comparison is made in the next section.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.94 ± 0.56	0.00 ± 0.02	1.00 ± 0.01	7	15.5
Debian 2006	5.17 ± 2.95	0.00 ± 0.00	0.96 ± 0.04	8	14.8
Debian 2007	7.95 ± 1.71	0.01 ± 0.08	0.98 ± 0.03	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian Logo	3.75 ± 1.74	0.00 ± 0.02	0.87 ± 0.08	8	40.0
Dublin North	10.66 ± 2.99	0.00 ± 0.00	0.99 ± 0.01	12	58.5
Dublin West	1.97 ± 0.17	0.03 ± 0.16	0.89 ± 0.01	9	50.8
Meath	1.00 ± 0.00	1.00 ± 0.00	0.91 ± 0.01	14	66.8

Table 5.10: Table showing the mean SWW, MWW, and FE measures for the Minimax Regret approach on the **K-Approval** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

even if it would significantly alter the outcome. In application domains where elicitation is not possible, or is undesirable, the method clearly suffers from serious deficiencies. Empirically, the initial results suggest these become pronounced when somewhere around 50% of preference information is missing. This is consistent with the idea of a multi-way tie: if c_1 and c_2 are unranked relative to one another on at least half of the ballots, then MMR will consider either victory possible, effectively declaring a pairwise tie.

Finally, Table 5.13 summarizes the results for MMR under the **Veto** voting rule. Under **Veto** MMR practically amounts to counting the number of ballots on which each candidate is unranked, and picking the candidate about whom there is greatest uncertainty (i.e. the one ranked on the fewest ballots), with a slight bias toward candidates that are never ranked last on the few complete ballots that exist in most of the sets. Perhaps surprisingly, this is not an especially good heuristic for **Veto**. MMR picks the correct winner more often than the imputation-based approach on just four of the nine datasets, and just two of the four difficult datasets, even though **Veto** is a pathological voting rule for the imputation-based approach. Kendall Correlation is higher for the imputation-based approach in the same fraction of the sets, and the imputation-based approach has a higher (i.e. better) average First Error Location on six of the sets, though on one of these the difference is very small.

The final set of results to be presented are those based on imputations selected uniformly at random. As discussed above, this provides an easily computable approximation of the

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.90 ± 0.70	0.00 ± 0.03	1.00 ± 0.01	7	15.5
Debian 2006	7.97 ± 0.33	0.00 ± 0.00	1.00 ± 0.01	8	14.8
Debian 2007	7.97 ± 2.16	0.00 ± 0.04	0.98 ± 0.03	9	19.1
Debian 2010	5.00 ± 0.11	0.00 ± 0.00	1.00 ± 0.01	5	11.0
Debian Logo	4.40 ± 2.00	0.00 ± 0.00	0.89 ± 0.08	8	40.0
Dublin North	7.87 ± 2.87	0.00 ± 0.00	0.96 ± 0.01	12	58.5
Dublin West	1.00 ± 0.00	1.99 ± 0.11	0.86 ± 0.03	9	50.8
Meath	1.00 ± 0.00	1.00 ± 0.00	0.91 ± 0.01	14	66.8

Table 5.11: Table showing the mean SWW, MWW, and FE measures for the Minimax Regret approach on the **Borda** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	4.40 ± 0.84	0.02 ± 0.09	0.90 ± 0.11	5	13.6
Debian 2005	6.57 ± 1.54	0.04 ± 0.13	0.99 ± 0.02	7	15.5
Debian 2006	6.00 ± 0.10	0.00 ± 0.00	0.93 ± 0.00	8	14.8
Debian 2007	2.05 ± 0.37	0.00 ± 0.00	0.87 ± 0.03	9	19.1
Debian 2010	3.04 ± 1.39	0.00 ± 0.00	0.82 ± 0.14	5	11.0
Debian Logo	5.44 ± 1.00	0.00 ± 0.00	0.74 ± 0.12	8	40.0
Dublin North	3.98 ± 0.14	0.00 ± 0.00	-0.09 ± 0.03	12	58.5
Dublin West	3.00 ± 0.00	0.72 ± 0.13	0.52 ± 0.05	9	50.8
Meath	2.97 ± 0.17	0.00 ± 0.00	-0.46 ± 0.04	14	66.8

Table 5.12: Table showing the mean SWW, MWW, and FE measures for the Minimax Regret approach on the **Copeland** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	2.91 ± 1.00	0.00 ± 0.04	0.85 ± 0.05	7	15.5
Debian 2006	1.00 ± 0.00	2.80 ± 0.39	0.72 ± 0.03	8	14.8
Debian 2007	1.01 ± 0.12	2.39 ± 0.66	0.64 ± 0.05	9	19.1
Debian 2010	1.00 ± 0.00	1.00 ± 0.00	0.80 ± 0.00	5	11.0
Debian Logo	1.03 ± 0.18	1.96 ± 0.89	0.58 ± 0.10	8	40.0
Dublin North	7.68 ± 1.09	0.00 ± 0.00	0.88 ± 0.02	12	58.5
Dublin West	2.08 ± 0.44	0.06 ± 0.24	0.58 ± 0.04	9	50.8
Meath	1.00 ± 0.00	3.00 ± 0.00	0.12 ± 0.01	14	66.8

Table 5.13: Table showing the mean SWW, MWW, and FE measures for the Minimax Regret approach on the **Veto** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

performance obtained by maximum-likelihood approach to voting [Xia and Conitzer, 2011], which also assumes unranked ballots are distributed uniformly at random (i.e. contribute no net value to the aggregates computed). The results are summarized in Tables 5.14-5.17. Table 5.14 shows the results under the **K-Approval** voting rule. The randomly generated imputations led to the selection of the correct winner on most of the datasets, illustrating (much like the worst-case results above) that there is relatively little room for poor decision making under the **K-Approval** rule on these sets. Interestingly, the method does exactly as well as MMR on the Meath dataset, consistently selecting the true second place candidate as the winner. This illustrates again the presence of real patterns in the expressed preferences of voters: a randomized imputation method that ignores these patterns fares worse than one that attempts to exploit them. Apart from this, the results are broadly similar to those for MMR. It appears that a random imputation has slightly better Kendall Correlation than the logistic regression-based approach on some of the sets, though the difference is not very large[5].

The results for the random approach under the **Borda** rule are summarized in Table 5.15. As under the **K-Approval** rule, the random approach makes mistakes that

[5]A possible explanation for this is that low-ranked preferences in these sets were in fact distributed nearly at random, making the rankings consistent with the assumptions of the random approach. If preferences really were expressed randomly, then random imputations would actually form an upper bound on performance for the low-ranked candidates on some sets. The result may thus reflect a limitation of the data that was used.

are effectively identical to those of MMR when selecting the winner of the election. On the Dublin West and Meath sets, the true third and second place candidates are respectively selected as the winner. Again, this mistake illustrates the presence of patterns for the (non-random) imputation-based approach to exploit. Clearly the assumption that unstated preferences are distributed uniformly at random is false, and better results are obtained via the observation of patterns in the stated preferences. The results for the random approach in terms of Kendall Correlation are very similar to those of MMR, and thus also, those of the imputation-based approach. The notable exception is on the Dublin North set, where imputations drawn uniformly at random recover the entire ordering most of the time. This is slightly surprising, as it suggests that voters' preferences were actually quite heterogeneous on this set. Further evidence of this heterogeneity is explored briefly near the end of the next chapter.

In contrast to MMR, the random approach also performs reasonably well under the **Copeland** rule (Table 5.16). Like MMR it routinely selects the correct winner, except on Dublin West where the third place candidate is picked instead. Kendall correlations are reasonably high, and comparable to those under the Imputation-Based approach, while the first error location is sometimes a little worse than MMR. This suggests the approach is recovering the correct ordering of the less popular candidates better than MMR by picking *some* ordering of the unpopular candidates, rather than assuming all orderings are equally likely. Again, this highlights the importance of elicitation to MMR's strategy. If no further information can be extracted from the voters, than picking an ordering conservatively can be detrimental.

To complete the tabular summary of the results, Table 5.17 displays the results for the random approach under the **Veto** voting rule. Interestingly again, the method performs worse than the imputation-based approach when picking the winner of the election on most sets, though not Dublin North or Dublin West. The Kendall Correlation results and First Error Location tell a similar story, with numbers that are broadly similar to those for MMR, and slightly worse than the imputation-based approach.

This concludes the presentation of the results for each of the models on each of the datasets. The next section provides a direct statistical comparison between the results summarized in tabular form here, and presents the same results graphically so that the viewer can more easily process them.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.94 ± 0.55	0.00 ± 0.00	1.00 ± 0.01	7	15.5
Debian 2006	5.21 ± 2.95	0.00 ± 0.00	0.96 ± 0.04	8	14.8
Debian 2007	7.99 ± 1.68	0.01 ± 0.07	0.98 ± 0.03	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian Logo	3.73 ± 1.66	0.00 ± 0.02	0.86 ± 0.08	8	40.0
Dublin North	10.36 ± 3.23	0.00 ± 0.00	0.99 ± 0.01	12	58.5
Dublin West	1.98 ± 0.16	0.02 ± 0.13	0.89 ± 0.01	9	50.8
Meath	1.00 ± 0.00	1.00 ± 0.00	0.91 ± 0.01	14	66.8

Table 5.14: Table showing the mean SWW, MWW, and FE measures for the Random Imputation approach on the **K-Approval** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	4.98 ± 0.23	0.00 ± 0.01	1.00 ± 0.01	5	13.6
Debian 2005	6.71 ± 1.18	0.00 ± 0.04	0.99 ± 0.02	7	15.5
Debian 2006	7.93 ± 0.52	0.00 ± 0.00	1.00 ± 0.01	8	14.8
Debian 2007	7.64 ± 2.40	0.00 ± 0.04	0.98 ± 0.03	9	19.1
Debian 2010	4.95 ± 0.37	0.00 ± 0.00	1.00 ± 0.02	5	11.0
Debian Logo	4.29 ± 1.94	0.00 ± 0.00	0.89 ± 0.08	8	40.0
Dublin North	7.81 ± 2.89	0.00 ± 0.00	0.96 ± 0.01	12	58.5
Dublin West	1.00 ± 0.00	1.95 ± 0.22	0.87 ± 0.03	9	50.8
Meath	1.00 ± 0.00	1.00 ± 0.00	0.91 ± 0.00	14	66.8

Table 5.15: Table showing the mean SWW, MWW, and FE measures for the Random Imputation approach on the **Borda** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	7.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	7	15.5
Debian 2006	7.17 ± 0.99	0.00 ± 0.00	0.97 ± 0.04	8	14.8
Debian 2007	7.18 ± 2.18	0.00 ± 0.00	0.97 ± 0.03	9	19.1
Debian 2010	4.98 ± 0.27	0.00 ± 0.00	1.00 ± 0.02	5	11.0
Debian Logo	5.08 ± 0.48	0.00 ± 0.00	0.76 ± 0.08	8	40.0
Dublin North	4.01 ± 0.25	0.00 ± 0.00	0.91 ± 0.00	12	58.5
Dublin West	3.00 ± 0.00	1.00 ± 0.02	0.72 ± 0.02	9	50.8
Meath	2.00 ± 0.00	0.00 ± 0.00	0.93 ± 0.01	14	66.8

Table 5.16: Table showing the mean SWW, MWW, and FE measures for the Random Imputation approach on the **Copeland** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

	First Error	Single Winner	τ	# Candidates	% Missing
Debian 2003	3.89 ± 1.79	0.25 ± 0.42	0.94 ± 0.09	5	13.6
Debian 2005	2.46 ± 1.17	0.23 ± 0.48	0.82 ± 0.07	7	15.5
Debian 2006	1.00 ± 0.04	2.73 ± 0.54	0.70 ± 0.06	8	14.8
Debian 2007	1.60 ± 0.88	0.91 ± 0.87	0.76 ± 0.06	9	19.1
Debian 2010	1.02 ± 0.26	1.00 ± 0.10	0.80 ± 0.02	5	11.0
Debian Logo	1.03 ± 0.26	2.22 ± 0.93	0.61 ± 0.10	8	40.0
Dublin North	7.44 ± 1.39	0.00 ± 0.00	0.88 ± 0.01	12	58.5
Dublin West	1.89 ± 0.54	0.21 ± 0.42	0.57 ± 0.04	9	50.8
Meath	1.00 ± 0.00	3.00 ± 0.00	0.12 ± 0.01	14	66.8

Table 5.17: Table showing the mean SWW, MWW, and FE measures for the Random Imputation approach on the **Veto** social choice function. Reported values are the mean over many problem instances, and reported measurement errors are the sample standard deviations.

5.3.2 Single Winner Performance

Figures 5.4– 5.7 provide graphical comparisons of the performance of MMR and the worst-case, random, and imputation-based models' performance across the four voting rules under the Single Winner Error measure. The results presented in these figures are based on those from the tables in the previous subsection, but should be easier for the reader to compare across models. Each figure summarizes data for one voting rule. Figures show a cluster of bars for each dataset, one bar for each model. The height of the bar shows the Single Winner Error of that method on the candidate — the mean value reported in the corresponding table. The error bars (sometimes called "whiskers" to distinguish them from the main bars of the plot) show a 95% confidence interval for the mean, assuming a t-distribution based on the appropriate number of samples and adjusted using the Bonferroni correction [Bonferroni, 1936; Dunn, 1961] for the fact that $3 \times 4 \times 9 \times 6 = 648$ hypotheses are being tested (3 performance measures, 4 voting rules, 9 datasets, and 6 tests between the 4 different methods for each treatment). This means there is at least a 95% chance that *every* true mean value falls between its corresponding set of bars. If the confidence intervals for two methods do not overlap, then a statistically significant difference is present between the methods' performance. When the intervals for two methods do overlap, a statistically significant difference may not be present. However, since the difference in such cases is vanishingly small for the data present here, no additional testing was performed, and for the purpose of assessing these results, such instances will be considered effectively equal.

Unsurprisingly, given the comparative ease of the decision problems generated, the **K-Approval** and **Borda** data summarized in Figures 5.4 and 5.5 is not especially interesting. All methods manage to recover the true winner on every one of the Debian sets, modulo some tiny errors on the Debian 2007 set under Borda. MMR (orange) and the random approach (green) make minor errors on the Meath and Dublin West sets. The worst-case approach is show in red, and demonstrates that the three Irish sets at least have some potential for poor decision making here, so it is encouraging that all perform well. The imputation-based approach appears to perform best overall. Figure 5.6 shows the performance of the four methods under **Copeland**'s rule. Here performance is again essentially perfect under the Debian sets, with a tiny error on the 2005 set. All three approaches sometimes make small mistakes on Dublin west, with MMR making statistically significantly fewer mistakes than either of the other two methods, and the imputation-based approach making fewer than the random imputation approach. On Meath the imputation-based approach makes consistent mistakes, selecting either the true fourth or fifth place candidate as the winner, MMR appears to perform best here, but not by a wide margin.

Results under the Veto rule, summarized in Figure 5.7, has much more varied results, since errors are possible on every set. Notable here is that the heuristic used to find the worst-case performance (described in Section 5.1) is actually inaccurate under veto. The truly worst candidate may not be a member of the possible winner set, but the heuristic will generate an imputation assigning the last position to this candidate wherever possible. This leads to odd looking results for the Debian 2005 and 2007 sets in particular, so the reader is reminded that the worst-case results are in fact the product of a heuristic, and so constitute a lower bound on the true worst case. On Debian 2005, 2007, 2010, and Logo, and on Meath, the imputation-based approach is a clear winner, despite the pathological nature of the voting rule. On Debian 2003 and 2005 it does slightly worse than competing approaches, but is still picking the correct winner much of the time. Only on the Dublin North and West sets does performance appear to suffer relative to the competitors.

In support of the visual results, a statistical analysis was performed. The data were analyzed with a three-factor analysis of variance (ANOVA) test, where the factors were the dataset, voting rule, and method. Only MMR, the imputation-based approach, and the random imputation approach were used as methods in the analysis. The null hypothesis was that all combinations of methods, datasets, and voting algorithms have equal performance. The null hypothesis was rejected and the test found that the method choice factor was a statistically significant predictor of performance even controlling for the presence of the other factors and their combinations.

Followup pairwise testing with Welch's t-tests was used to holistically assess the significance of differences between each method (i.e. across all datasets simultaneously). The difference in single winner error between the imputation-based approach with logistic regression and MMR had a 99.99% confidence interval between 0.067 and 0.10, favouring the imputation-based approach. A similar advantage was found for the imputation-based approach over random imputations, with a 99.99% confidence interval of 0.055 to 0.088 for the mean difference, again favouring the imputation-based approach. Interestingly, the difference between MMR and the random imputation approach was only significant at a 0.99% level (despite being based on more than 11,000 data points). A 0.99% confidence interval for the difference between means is 0.001 to 0.0027 in favour of the random approach.

Results under individual voting rules have somewhat larger (and therefore perhaps somewhat more informative) effect sizes. While the advantages in **K-Approval** are very small (about 0.01 in favour of the imputation-based approach), the imputation-based approach offers a mean advantage of 0.18 to 0.21 over MMR under **Borda**, and of 0.17 to 0.21 over random imputations. Under **Copeland** there was no significant difference between the imputation-based and random methods, and MMR offered an advantage of about 0.01 points of single winner error. Under **Veto** the imputation-based approach was best on

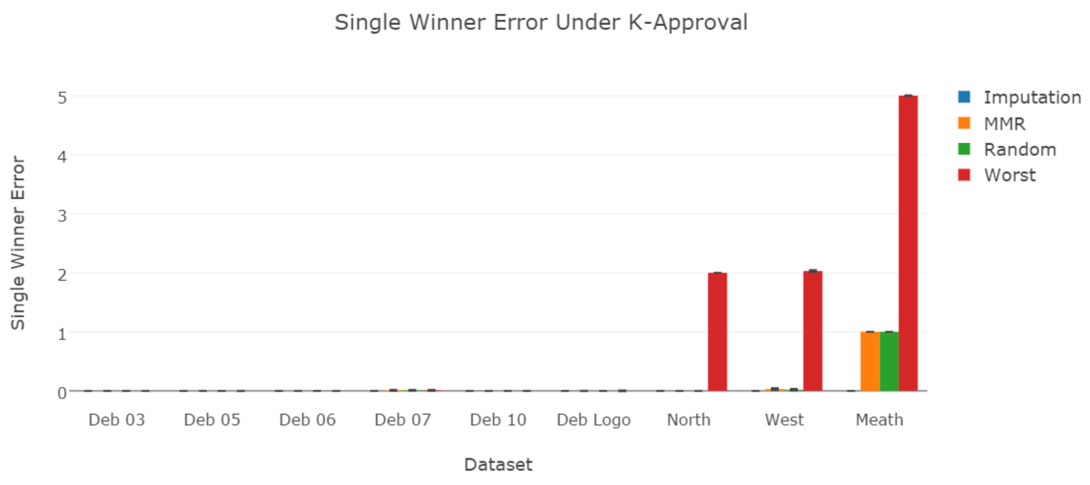


Figure 5.4: Figure comparing Winner Determination Performance of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **K-Approval** social choice function. Bars show mean SW Error, and whiskers show one standard deviation.

average again, with a 99.99% confidence interval for the mean advantage over MMR of 0.21 to 0.097, and over the random approach of 0.14 to 0.023.

On this basis, it is reasonable to conclude that the imputation-based approach is most likely to pick the true winner of the election, though only by a modest amount. On the **Borda** and **Veto** rules its performance amounted to picking a candidate one position closer to the true winner about 20% more often than the other approaches. On **K-Approval** the imputation-based approach picks a candidate one position closer to the true winner only about 1% more often, while under **Copeland** MMR picks a candidate one position closer about 1% more often.

5.3.3 Ranking Performance

The performance of the three methods of interest alongside the worst-case method's performance is now analyzed in terms of Kendall Correlation, a measure of their overall abilities to recover the correct ordering of all the candidates in an election. The results are summarized in Figures 5.8- 5.11, which respectively show the Kendall correlation under each of the four different voting rules considered in the experiment. As with the results for

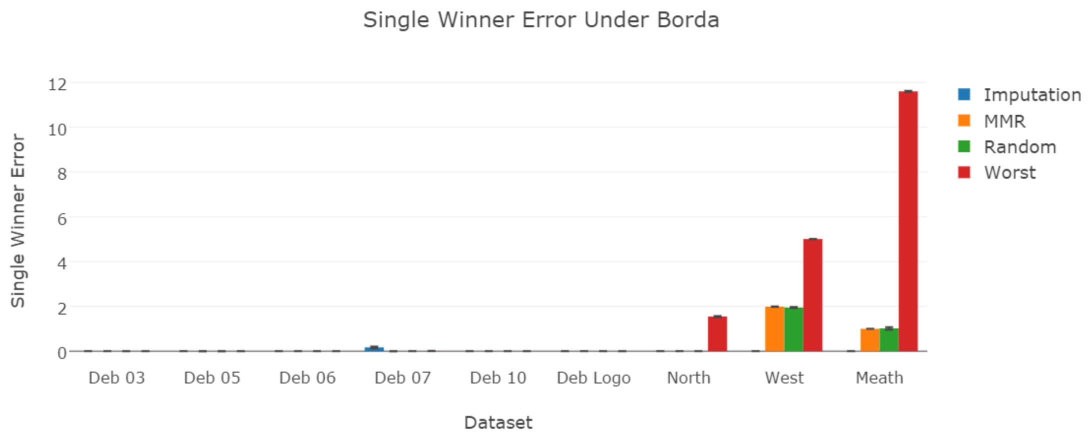


Figure 5.5: Figure comparing Winner Determination Performance of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Borda** social choice function. Bars show mean SW Error, and whiskers show one standard deviation.

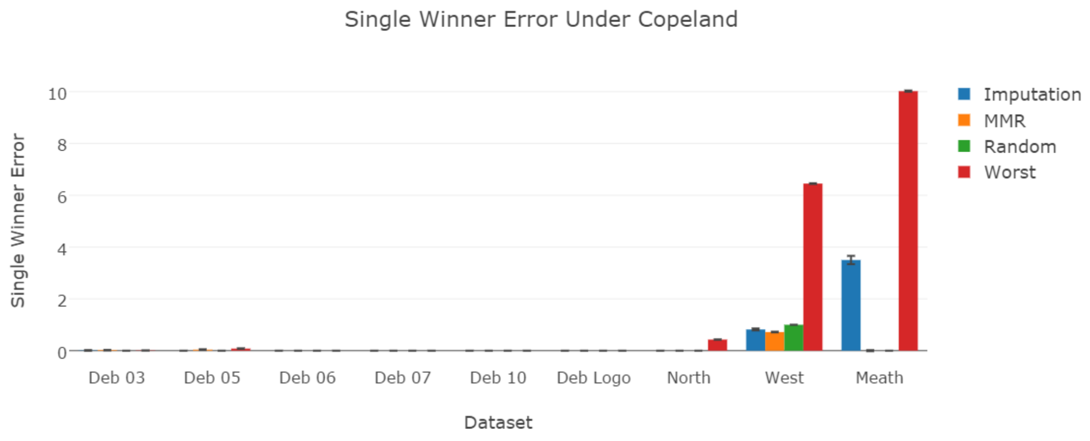


Figure 5.6: Figure comparing Winner Determination Performance of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Copeland** social choice function. Bars show mean SW Error, and whiskers show one standard deviation.

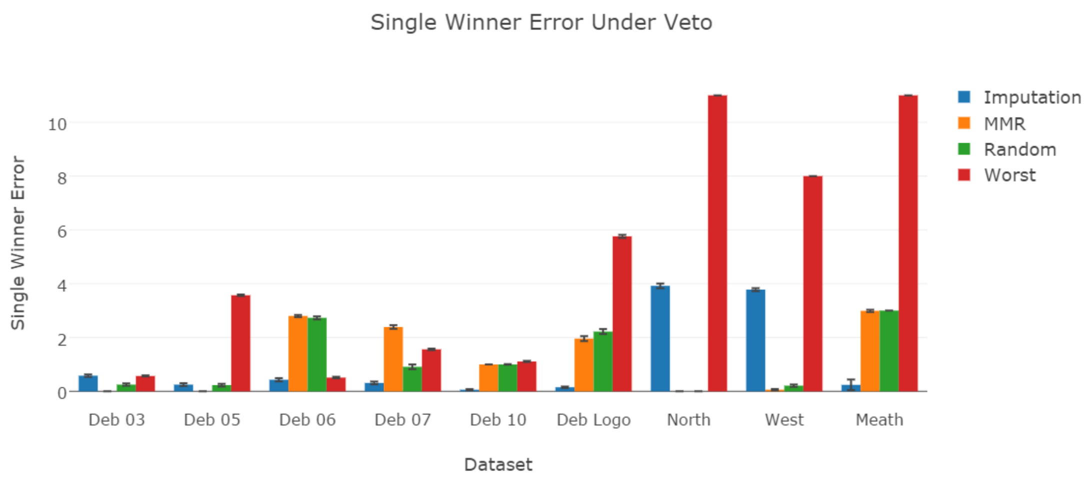


Figure 5.7: Figure comparing Winner Determination Performance of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Veto** social choice function. Bars show mean SW Error, and whiskers show one standard deviation.

Single Winner Error above, each figure shows a set of coloured bars clustered together for each dataset. The height of the bar corresponds to the Kendall-Tau correlation between the ranking output by a method and the ground-truth ranking. The error bars show a 95% confidence interval, corrected for the extremely large number of multiple comparisons being made in exactly the same way as in the Single Winner Error comparisons. Note that while high Single Winner Errors indicted a method was performing badly, high correlation is actually indicative of the opposite (i.e. methods with higher Kendall correlation are performing better than those with lower Kendall correlation).

Figure 5.8 shows the results under **K-Approval**. There appears to be a consistent trend for the imputation-based approach to perform very slightly less well than the competing approaches, with the exception being a small advantage for the imputation-based approach on the Debian 2006 set. Interestingly, the competitors exhibit very similar performance to each other on all the sets, appearing to be essentially indistinguishable from one another. This effect is also visible under **Borda** in Figure 5.9, but here the imputation-based approach out-performs its competitors as often as it is beaten by them. In all cases on both sets, performance for all the methods is very high.

Figure 5.10 shows the much more varied results present under the **Copeland** rule. Here performance for the imputation-based approach and the random approach is similar, but MMR is noticeably worse, and is even anti-correlated with the ground-truth ordering on the North and Meath sets. As discussed earlier, this is due in part to MMR's tendency to declare ties under **Copeland**, although a cursory probe of the results indicated that it also does make some genuine mistakes in the ordering. The imputation-based approach appears to be equal to the random approach with near perfect or perfect correlation in the Debian 2003, 2005, 2006 and 2010 sets It is slightly worse in the Debian 2007, North, and Meath sets, and slightly better in the West set. The two methods are tied in the Logo set as well, but with less than ideal correlations of around 0.6. It certainly appears safe to conclude visually that both methods are superior to MMR under this rule.

Results under **Veto** are rather similar to those under the first two rules, with all methods performing reasonably well. The imputation-based approach performs notably better on the Meath set, and on Debian 2005, 2006, 2007, and 2010, but slightly worst on Debian 2003, and Logo, and on the West set. A fairly large deficiency is present on the North set.

As before, a series of ANOVA tests, followed by pairwise hypothesis testing with Welch's t-tests, was used to verify whether there was an overall advantage for any of the three methods, and how large the advantages were in general and on each dataset. The first ANOVA had as the null hypothesis that all blocks in a three-factor design had the same mean performance, with the factors being respectively the datasets, the methods, and the voting rules used. The test indicated that model choice was a highly significant factor, even in the presence of the dataset and voting rule factors, and the combinations of said factors ($p < 2e - 16$). Followup testing found an overall advantage for the imputation-based approach over MMR, with a 99.99% confidence interval for the mean difference in the mean Kendall correlation lying between 3.0 and 3.7 percentage points. Surprisingly, the random imputation method was found to offer a small advantage over the imputation-based approach under the Kendall correlation measurement. However, the mean Kendall correlations of the two methods were 0.09173 and 0.09138 respectively, with a 99.99% confidence interval for the gain of using the random approach of just 0.00053 to 0.0066. The difference is significant, but extremely small. It corresponds to approximately one pair of adjacent candidates being flipped for every 2.5 runs on the most complex dataset (Meath), which has 14 candidates. In contrast, the advantage of using either the random approach or the imputation-based approach over MMR corresponds to approximately 3 such flips *per run.*

As before, the advantages on individual rules are also examined. Under the **K-Approval** rule, both competing approaches offer a performance advantage of about 2.5% over using the imputation-based approach. Under **Borda** all three methods are tied, with less than a 1% difference in mean performance. Under **Copeland** a large advantage is present for

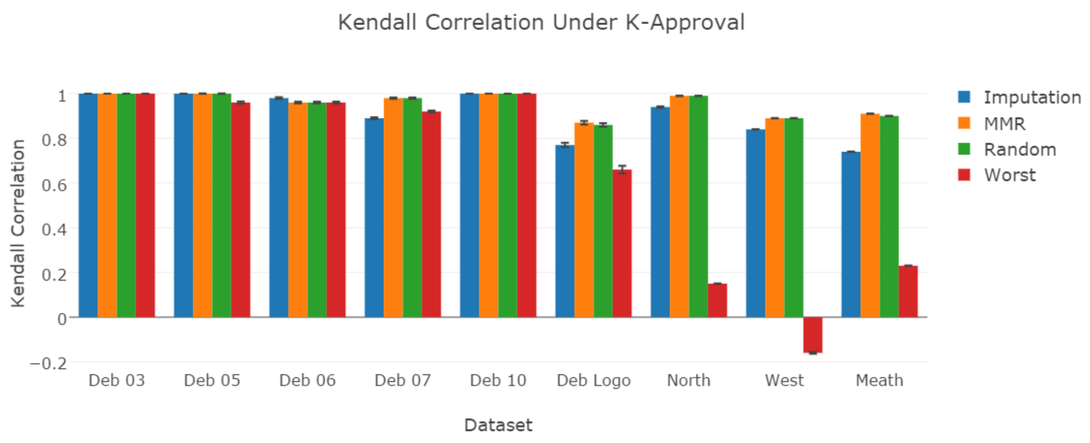


Figure 5.8: Figure comparing correlation of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **K-Approval** social choice function. Bars show mean Kendall Correlation, and whiskers show one standard deviation.

random over MMR (10.5 to 12.2 percentage points), and the imputation-based approach (9.7-11.5 percentage points). Using the random approach then offers a performance gain of between 1.1 and 0.1 percentage points). Under the **Veto** the imputation-based method offers a small advantage of 0.7-2.4 percentage points over the random method, and 4.5-6.2 percentage points over MMR. All reported values are 99.99% confidence intervals for the mean difference.

On the basis of these results, it must be concluded that minimax regret has substantially less ability to recover the entire sequence than either of the other two methods. Poor performance is especially notable on the **Veto** and **Copeland** rules, perhaps because the less popular candidates place very few constraints on the potential regret associated with each other. Interestingly, this analysis must also conclude that the randomized approach is at least a reasonable alternative to the imputation-based approach when ordering all candidates, even if it works less well when selecting only the winner. Possible reasons for this are discussed nearer the end of the section, in conjunction with the results under the First Error Location measurement.

Analysis under the First Error Location measurement was performed in much the same fashion as for the other two. The results are presented visually in Figures 5.12- 5.15. Again, each plots shows a cluster of bars for each dataset, with different colours corresponding

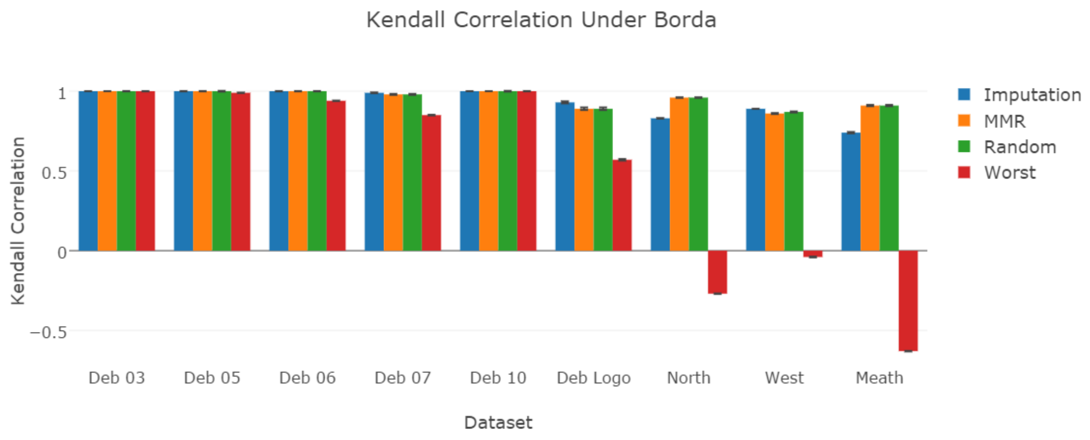


Figure 5.9: Figure comparing correlation of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Borda** social choice function. Bars show mean Kendall Correlation, and whiskers show one standard deviation.

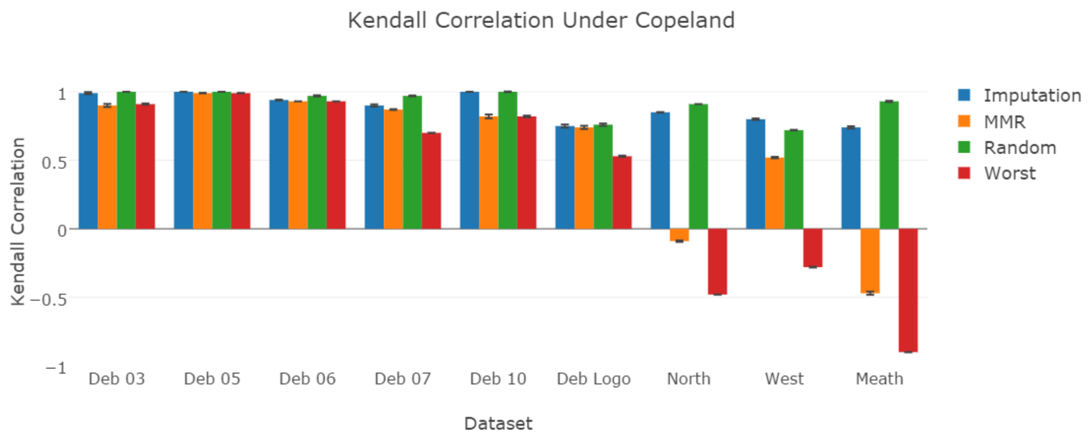


Figure 5.10: Figure comparing correlation of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Copeland** social choice function. Bars show mean Kendall Correlation, and whiskers show one standard deviation.

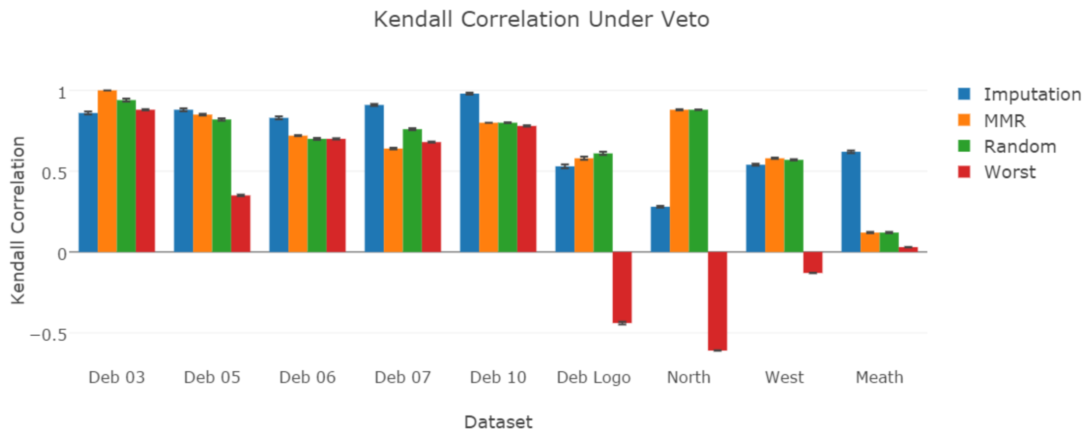


Figure 5.11: Figure comparing correlation of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Veto** social choice function. Bars show mean Kendall Correlation, and whiskers show one standard deviation.

to the different methods under consideration. In addition to the familiar four methods, a fifth bar has been added to each cluster corresponding to the number of candidates in the corresponding dataset, which is the upper bound for the worst possible First Error Location. As with Kendall correlation, higher bars indicate better method performance, since more of the highly ranked candidates from the ground-truth ordering have been ordered correctly. As before, whiskers show a 95% confidence interval under the conservative Bonferroni correction.

Figure 5.12 shows the results under **K-Approval**. Again, note that the worst-case method has used a heuristic approach, and may not always be a reliable bound. Overall it appears that the three methods of interest have similar performance under the Debian 2003, 2005, and 2010 sets, as well as under Dublin West. The imputation-based approach offers an advantage on Meath and Debian 2006, but has significant performance issues on Debian 2007, Logo, and Dublin North. The performance of MMR and the random approach appears identical throughout.

Figure 5.13 shows the results for the Borda set. A small performance advantage is present for the imputation-based approach on Debian 2005, and Logo, Dublin North, and Meath, while it performs somewhat less well on Dublin North. A few of the sets exhibit rather large whiskers. Readers may choose to interpret these as statistically significant differences even if they overlap, because of the highly conservative nature of the Bonferroni correction (in effect, the bars are all twice as wide as if only a single hypothesis were being tested). It is likely that they would be significant if tested with Holm's correction, for instance.

Figure 5.14 shows performance under **Copeland**'s rule. As with the Kendall correlation, here MMR appears to perform noticeably worse than the other methods on several of the sets. It offers a small advantage on Meath, but is far behind on Debian 2007 and 2010, and still worse than the other two methods on Debian 2003, 2005, 2006 and Dublin West. Also notable here is a modest advantage for the imputation-based approach, which does markedly better than both its competitors on Debian 2007 and Dublin West.

Finally, Figure 5.15 shows the results under the **Veto** rule. Immediately obvious is the difficulty of the rule, with all methods performing fairly poorly compared with the best-case performance shown in purple. The imputation-based approach does very well here, with advantages on Debian 2005, 2006, 2007, 2010, and Logo, as well as the Meath set. Interestingly, there is even one set (Debian 2006) where the competitors do worse than the 'worst case' approach, which as mentioned above, is a heuristic estimate of worst-case performance.

As before, the results in the graphs are supported by a more detailed statistical analysis. A three factor ANOVA (over the method, voting rule and dataset factors) again found that the choice of method was a highly significant factor in determining First Error Location ($p < 2e{-}16$). Pairwise hypothesis tests performed using Welch's t-tests found an advantage to using the imputation-based approach instead of MMR of between 0.25 and .15 points of First Error Location, (99.99% confidence interval for the difference in means), but also found that the random approach was preferred to the imputation-based one by between 0.0036 and 0.072 points of First Error Location, though this difference was only significant at the 99% level. The random approach also significantly outperformed MMR overall, with an advantage of 0.18 to 0.29 points of First Error Location.

Drilling down to look at performance on each voting rule individually, MMR offered a performance gain of 0.70-0.90 points over the imputation-based approach under **K-Approval**, and the random approach offered an identical gain. MMR and the random approach were not significantly different at the $p = 0.95$ level on under this rule. Under **Borda** however, the imputation-based approach performed better, with a 99.9% confidence interval for the difference between the means of 0.17 to 0.0035 over the random model, and statistically identical performance to MMR. Under **Copeland** however, the imputation-based approach was the best by a wide margin, with a 1.06 - 1.21 point advantage over MMR, and a 0.12 to 0.27 point advantage over the random approach, both at the 99.99%

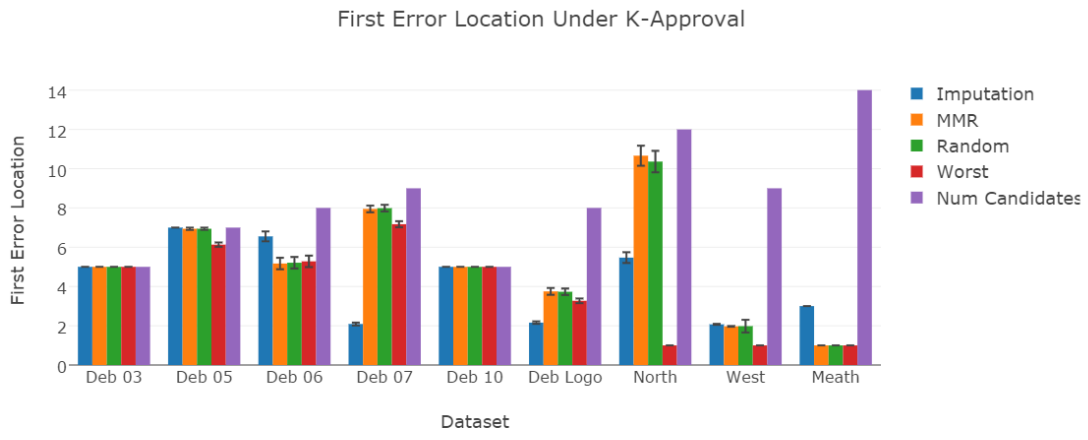


Figure 5.12: Figure comparing first error location of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **K-Approval** social choice function. Bars show mean location of the first ranking error, and whiskers show one standard deviation.

confidence level. A similarly large margin is present under **Veto**, where the imputation-based approach wins by a margin of 0.28-0.44 over the random approach, and 0.36-0.52 over MMR. Thus, somewhat counter intuitively, the imputation-based approach did best under **Copeland** and **Veto**, tied for best under **Borda** and did worst under **K-approval**, but overall was still slightly worse under the First Error Location measure than the random approach. As with the Kendall Correlation results, it is reasonable to conclude that MMR has performed less well on the whole than the other two rules. The difference between the random model's performance and the imputation-based approach's performance is small here, like with the Kendall correlation results. Again, for comparison, taking the mean differences, using the imputation-based method instead of the random method would result in getting the top $k+1$ candidates correct instead of the top k approximately once in every twenty elections. In contrast, using MMR instead of the imputation-based approach would result in an improvement of this kind in one in every five elections.

Overall the results suggest that the imputation-based approach should be preferred to its two competitors. It demonstrates significantly better performance than MMR overall under all three measurements (Single Winner, τ, and First Error), although on certain voting rules this advantage is small, or even slightly negative. Oddly however, although single winner performance is better for the imputation-based approach than for the ran-

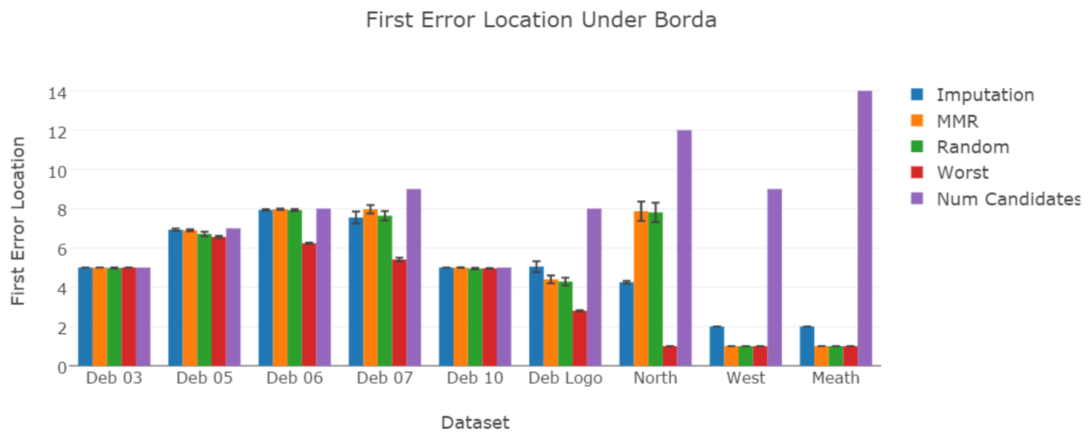


Figure 5.13: Figure comparing first error location of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Borda** social choice function. Bars show mean location of the first ranking error, and whiskers show one standard deviation.

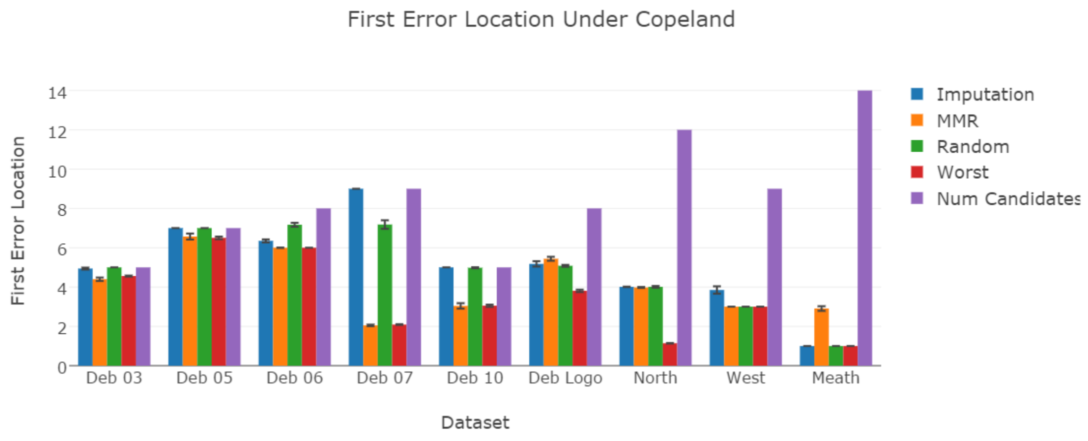


Figure 5.14: Figure comparing first error location of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Copeland** social choice function. Bars show mean location of the first ranking error, and whiskers show one standard deviation.

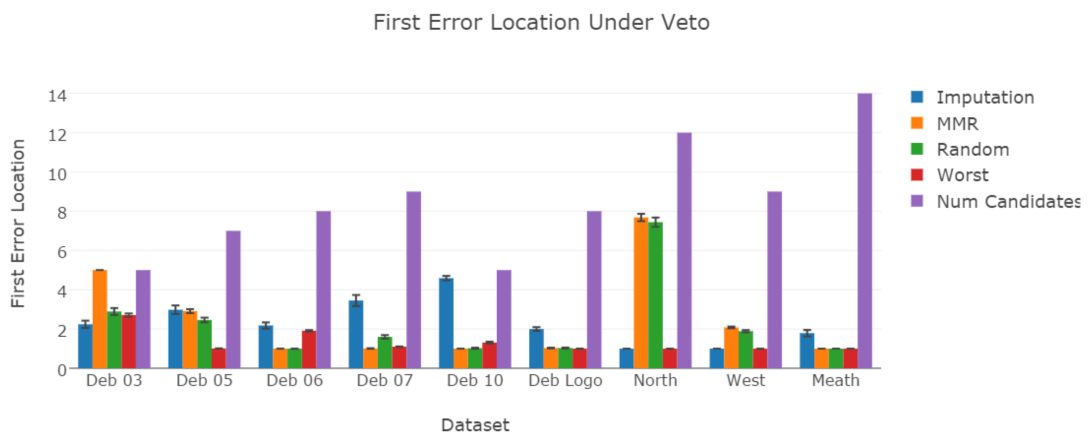


Figure 5.15: Figure comparing first error location of the imputation-based approach with logistic regression to MMR, random imputations, and worst-case imputations, under the **Veto** social choice function. Bars show mean location of the first ranking error, and whiskers show one standard deviation.

dom approach, the multiwinner results show a (very small) advantage to using random imputations. The next section provides an analysis of this phenomenon, and suggests both possible explanations and remedies.

5.4 Classifier Error Rates

The results in the previous section showed that although the imputation-based approach sometimes does perform better than its competitors, it appears to be highly comparable to using random imputations when measuring the recovery of the entire aggregate ranking. This is a rather unintuitive result, and so requires a compelling explanation.

As a first step toward explaining the phenomenon, it will be useful to develop an analytical model for the failure or success of the imputation-based model when ordering candidates. This model is based on the concept of the expected ϵ-case damage to the "correct" outcome of the election, which is denoted $ED(c, \epsilon)$, where $c \in C$ is the winning candidate, and ϵ is a real valued parameter in $(0, 1)$. $ED(c, \epsilon)$ is defined as the expected total decrease in the score assigned to a candidate under a scoring-based voting rule S to candidate c's under an imputation that places c at a fraction ϵ positions from the worst possible position on each ballot where c's position was not defined. Intuitively, this is somewhat similar to the idea behind minimax regret, but with greater continuity. Using $\epsilon = 1$ will compute the maximum distance that a candidate can be moved from its true position to one lower in the ranking. Using $\epsilon = 0$ computes the maximum distance that a candidate can be moved from its true position to one higher in the ranking. Using $\epsilon = 0.5$ will compute the distance a candidate will be moved by a random imputation (in expectation) for rules like Borda, although this is not exactly the same for other voting rules. More formally, suppose that the set of ballots B is generated by some process $\mathcal{N}$ generates top-ordered ballots from total orders $\succ$ over C by sampling a truncation point, and deleting all information about the relative ordering of candidates below that point. Suppose that an imputation method is then applied to B to produce $\succ_{c,\epsilon}$, such that c is placed, on average a fraction ϵ of possible positions below the maximum possible position on each ballot where it has been ablated. Then the expected ϵ-case damage to c under process $\mathcal{N}$ is given by:

$$ED(c, \epsilon) = E[S(c, \succ) - S(c, \succ_{c,\epsilon})]$$

For example, suppose that a candidate has Borda score 100 under the ground-truth data, and an average Borda score of 75 under random imputations after being ablated with a process $\mathcal{N}$ that ablates ballots according to a distribution of missingness learned from a more complete set of preferences (as in the experiments above). Then $ED(c, \epsilon)$ will have value -25.

Let $\mathcal{N}(j)$ be the probability that a total order is truncated at or before location j on a ballot by process $\mathcal{N}$. If a positional scoring rule S assigns scores using scoring vector $\{s_1, ..., s_{|C|}\}$, and total orders are generated by some process that assigns candidate c to position P with probability $P(\text{Pos}(c, \succ_i) = k)$, then it is easy to show that

$$ED(c, \epsilon) = \sum_{k=1}^{|C|} P(\text{Pos}(c, \succ_i) = k) \sum_{j=1}^{k} (\mathcal{N}(j) - \mathcal{N}(j+1))(s_k - \frac{1}{2}(s_{\lceil \epsilon|C| + (1-\epsilon)j \rceil} + s_{\lfloor \epsilon|C| + (1-\epsilon)j \rfloor})$$

The expected ϵ-case *gain* to a candidate c' $(EG(c', \epsilon))$ is the expected total increase in the score of c' if B is imputed to increase the score of c' in a manner inverse to that of the expected ϵ-case damage above. That is,

$$ED(c, \epsilon) = E[S(c', \succ^{g}_{c',\epsilon}) - S(c', \succ)]$$

where $\succ^g_{c',\epsilon}$ is the imputation after $\succ$ is ablated by $\mathcal{N}$ that places c' a fraction epsilon of possible positions higher than the worst case on every ballot where it was ablated. For position score rules, this can be written

$$EG(c',\epsilon) = \sum_{k=1}^{|C|} P(\mathrm{Pos}(c',\succ_i) = k) \sum_{j=1}^{k} (\mathcal{N}(j) - \mathcal{N}(j+1))(s_k - \frac{1}{2}(s_{\lceil (1-\epsilon)|C|+\epsilon j\rceil} + s_{\lfloor (1-\epsilon)|C|+\epsilon j\rfloor})$$

The ϵ-case margin of an election between c and c' after ablating $\succ$ with $\mathcal{N}$ is defined as

$$S(c,\succ) + ED(c,\epsilon) - (S(c',\succ) + EG(c',\epsilon))$$

When the margin is negative, the ablation is expected to cause enough damage that a systematic error favouring c' and harming c of ϵ per ballot in a candidate imputation will cause selection of the incorrect winner. For example, when $\epsilon = 0.5$, a negative margin indicates that an imputation generated uniformly at random is expected to result in selection of the wrong winner under rules like **Borda**. If $\epsilon = 1.0$, then a negative margin indicates that a worst-case imputation of the data would cause selection of the wrong winner in expectation. The bias of a classifier m with respect to a candidate c is defined as

$$bias(o,c) = \sum_{k=1}^{|C|} P(\mathrm{Pos}(c',\succ_i) = k)\mathcal{N}(k) \sum_{j=1}^{|C|} P(\mathrm{Pos}(c', m(B_i)) = j)(s_k - s_j)$$

where $P(\mathrm{Pos}(c', m(B_i)) = j)$ is the probability that m assigns a candidate c' to position j when imputing ballot B_i (under the assumption that imputations are done independently). That is, the bias of m with respect to c' is the expected change in the score of c' in the ballots ablated by $\mathcal{N}$ and then imputed by m, relative to the ballots originally generated. The bias of a classifier with respect to two candidates $bias(c,c',m)$ is just $bias(c,m) - bias(c',m)$.

With all of these definitions recorded, it is now possible to describe the circumstances under which a classifier m will cause the incorrect candidate to be selected as the winner in expectation. Under **Borda** and other monotonic positional scoring rules this occurs when $-bias(c,c',m) \leq$ the 0.5-case margin between c and c' for any $c' \in C$. Interestingly, this explains the small error made by the imputation-based approach when picking the winner under **Borda** on the Debian 2007 dataset. On this set the **Borda** scores of the winning and second place candidates differ by just 0.002%. A classifier that imputed the second place candidate 1 position higher than its correct location on just 10 ballots would thus cause the selection of the incorrect winner.

It is also apparent that classifier accuracy is thus most important in general when unpopular candidates are being imputed. Unpopular candidates will tend to have relatively small differences in their scores, since they will both tend to be ranked near the end of ballots, and thus near each other. They will also tend to have greater potential for damage and gain because they are more likely to be ablated on ballots than popular candidates are. However, the imputation-based approach will tend to have larger errors in the accuracy for imputing candidates that are unpopular because there is less training data to work with (again, because the candidates are more likely to have been ablated from ballots by $\mathcal{N}$).

Evidence supporting this hypothesis was found via a separate experiment that used only data from the Dublin North set. The imputation-based approach was run with logistic regression as the based classifier, and using only the Borda voting rule. The bias of the resulting imputations was measured for each of the twelve candidates over 100 problem instances generated using the approach described in Algorithm 7 above. Figures 5.16 and 5.17 summarize the results of this experiment. There is a clear trend to produce proportionately higher bias for candidates that have lower overall scores (i.e. for candidates that are less popular).

As further evidence of the bias toward more popular candidates, data from the original experiment was collected using a fourth performance measure, a weighted version of the Kendall correlation:

$$\tau_w(\sigma_1, \sigma_2) \propto \sum_{c_i \in C} (|C| - \text{Pos}(c_i, \sigma_1) \sum_{c_j \in C \setminus c_j} I(c_i \ \sigma_1 \ c_j \wedge c_i \ \sigma_2 \ c_j) + I(c_j \ \sigma_1 \ c_i \wedge c_j \ \sigma_2 \ c_i))$$

Under this measure, the imputation-based approach had statistically identical performance to the random imputation-based approach (ANOVA+t-tests, $p > 0.13$), despite the existence of a modest advantage in favour of the random approach using the unweighted Kendall correlation. This supports the idea that the imputation-based approach is favouring popular candidates, and that it is making mistakes about the order of less popular ones, rather than more popular ones.

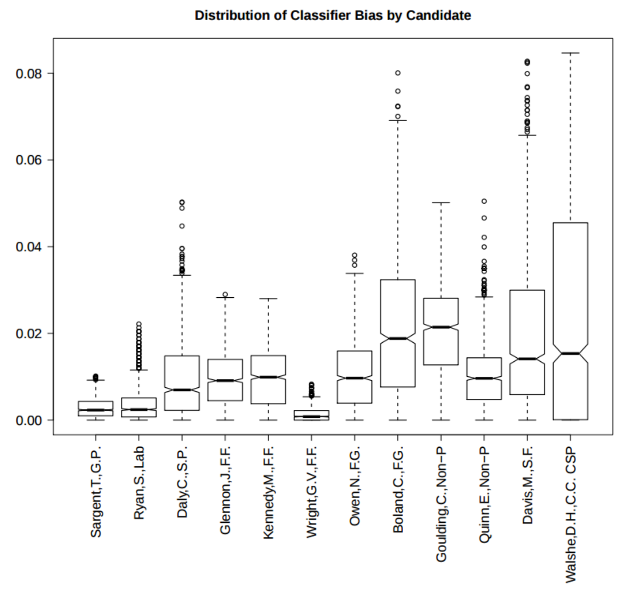


Figure 5.16: Boxplots showing the observed distribution of bias for each candidate in the Dublin North election when the imputation-based approach was used to decide the election. Each boxplot shows the median, and first and third quartiles of the magnitude of the bias in that candidate's score as a proportion of their total score, with whiskers showing the locations of the most extreme values within 1.5 times the height of the main box. The popularity of candidates increases from left to right.

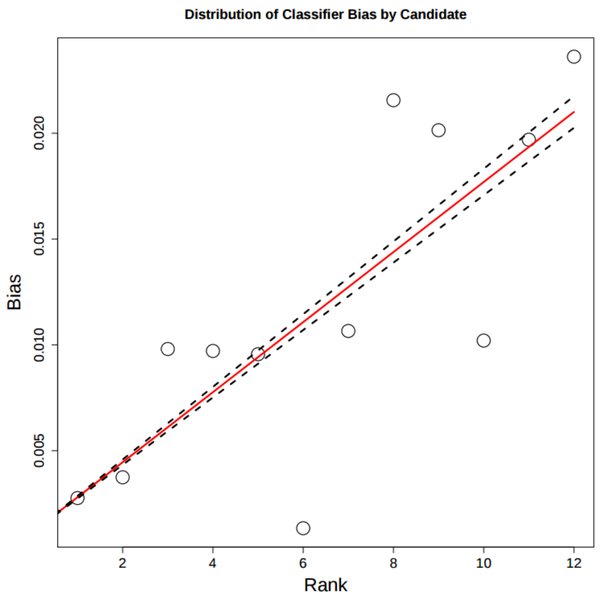


Figure 5.17: Plot of the mean bias for each candidate with candidates again sorted in order of their **Borda** scores on the ground-truth data. The solid red line shows the linear least squares regression on the points, using their ranks as the dependent variable. The dashed black lines show a 95% confidence interval for the slope of the line.

5.5 Discussion

The high performance of the imputation based approach clearly demonstrates its potential for use on real-world problems. The method was more effective than either competitor in picking the correct winner of an election, and was more effective than minimax regret when recovering the entire sequence. Oddly, the imputation-based approach performed very slightly worse than the random imputation approach in recovering the entire sequence however. This appears to be the result of a bias in the imputation-based approach toward more popular candidates. This is both because more popular candidates will tend to have larger margins between their scores and those of their neighbours (in absolute terms, for monotonic scoring rules), and because there are fewer observations of ballots that rank unpopular candidates (by definition).

There is a close parallel between this behaviour and the problem of class imbalance in classification. This problem arises when one of the classes in a dataset appears far more often than others do. For example, a dataset of liver function measurements collected at an ordinary hospital will tend to have many examples of patients with normal liver function, and very few with abnormal liver function. If the patients with abnormal liver function are subdivided further into classes based on the precise nature of their malfunction, then the ratio of the number of normal examples to the number of examples belonging to any of the abnormal classes individually may become enormous. If a binary classifier is trained to differentiate between two of these classes, extremely high accuracy can be obtained from obviously incorrect models. For example, a model that labels every patient as having normal liver function might accurately predict the labels of 99% of patients, even though such a model is essentially useless. Many classification algorithms perform poorly when the data exhibit class imbalance [Akbani et al., 2004] and the problem appears most often when the amount of data is small, the degree of imbalance is high, and the patterns in the data are complex [Japkowicz and Stephen, 2002]. Interestingly, these are precisely the conditions that are likely to be presented when learning to impute the orderings of unpopular candidates. Such candidates may not appear on many ballots, leading to a shortage of data. In fact, on many of the datasets only around 10% of voters rank a substantial portion of the candidates at all, and typically only a few "serious" candidates are commonly ranked (perhaps because of strategic voting), meaning the degree of imbalance may be exceptionally high. Finally, quite complex patterns may be present regarding the ordering of unpopular candidates. For example, in a political contest, such candidates might represent special interest parties that are located outside the normal political spectrum. In something like a robotic mining swarm, such alternatives might be sites that no companies have prospected in detail, leading to high variation in the assessments of the companies involved. There are a number of ways to address the class imbalance problem that could be used to improve the performance of the imputation-based approach. These include oversampling the minority class (i.e. adding extra copies of ballots that rank unpopular candidates until they appear roughly equal in popularity to more popular ones, but only when training the classifier); undersampling the majority class (erasing some fraction of the ballots ranking popular candidates until the classes are approximately balanced); and changing the cost function so that errors in minority class are viewed as more expensive, in proportion to the degree of class imbalance (e.g. in logistic regression, penalizing an error in the classification of unpopular candidates by more than an error in the classification of popular ones during training). These techniques are not explored in detail in the thesis, as the later model proposed in Chapter 7 provides a more elegant way to control the bias.

5.6 Summary

This chapter set out to answer the question of whether real-world social choice problems were more consistent with the assumptions of the imputation-based approach, or the assumptions of competing models. Answering this question involved the construction of an experiment design that used real-world data as the basis for the generation of many different problem instances for which the ground-truth (i.e. the true preferences of voters) was known. The imputation-based approach was compared to an implementation of the minimax regret algorithm [Lu and Boutilier, 2011b], and to a random imputation approach intended to mimic the maximum-likelihood approach to voting [Xia and Conitzer, 2011]. The imputation-based approach outperformed both models overall in terms of determining the winner of the election across a wide range of problems. It also outperformed MMR significantly in recovering the overall ordering of the candidates. However, the random model had a very small, though significant, advantage over both MMR and the imputation-based approach when inferring the entire ordering. This issue was explained in terms of the bias of the imputation-based approach, which causes it to make larger mistakes in inferring the positions of unpopular candidates. The bias problem was related to the issue of class imbalance in more general machine learning, but was not explicitly addressed within this chapter.

Chapter 6

Prefmine Experimental Testbed

Man is a tool-using animal ... Without tools he is nothing, with tools he is all

Thomas Carlyle [1893]

Chapter 5 provided some detailed results showing the feasibility of using the imputation-based approach to decide the outcome of elections with incomplete information. Although the experimental framework used was described at a high level, the framework is in fact a highly robust and flexible testbed system. The system could be used by other interested researchers and practitioners to evaluate new methods, to find patterns in new electoral datasets, or even to decide elections using any of the methods described in this thesis. The chapter contains several sections. Section 6.1 describes the testbed system in detail, serving both to situate it within the growing set of tools for working with electoral data, and as a simple user manual for the system. Potential improvements to the system, and a discussion of the extensible framework that was used are found near the end of the section. Section 6.2 contains a discussion of the problems that may occur when processing electoral data with a system that was not constructed as carefully as the testbed system described in Section 6.1, as well as a further analysis of different classifiers that could be used to instantiate the imputation-based approach, and an explanation of why the testbed system contains the specific algorithms that it does, and not others. The chapter is followed by Chapter 7, in which a new learning algorithm is proposed and evaluated thoroughly using the features of the testbed system.

6.1 The Prefmine System

The results in Chapter 5 involved 100 repetitions of experiments with 4 voting rules, 11 electoral datasets containing tens of thousands of ballots in total, and 4 different systems for deciding the outcome of an election. In total 4,400 decision problems were considered, and 17,600 decisions were made. When performing experiments on this scale, it pays to have a reliable system, in which one can be certain of the integrity of the data, experimental methodology, and results. Such a system was constructed largely from scratch, but in a carefully tested and designed fashion. The resulting system has several features that may be of great interest to practitioners and to other researchers. First, since the system contains implementations of the imputation-based approach to social choice under several classification systems, and also of the Minimax Regret approach [Lu and Boutilier, 2011b], a worst-case approach, and several other algorithms, it can serve as a ready means of comparison for practitioners interested in evaluating their own approaches, or in making electoral decisions. Researchers may also be interested in the system as an independent implementation of existing algorithms for social choice, given that typical scientific software has disagreement in the output results on the order of 10% between different implementations of the same algorithm [Hatton and Roberts, 1994; Hatton, 1997], perhaps because of poor testing practices adopted in the development of most scientific software [Kelly and Sanders, 2008]. By comparing the results of several independent implementations, researchers can be surer of their results, and uncover issues in their own implementations. Second, the system contains implementations of four voting rules, and is readily extensible to accommodate many more. Practitioners interested in using the system to decide elections can compare the results under several different voting systems, with or without additional systems like MMR or the imputation-based approach augmenting them. Further, the implementations of these systems adopt a simple parallelization strategy to provide quick evaluation. Finally, other researchers may be interested in the experimental design adopted by this thesis, which leverages real world datasets to create plausible problems where the ground truth is known, but concealed from the algorithms under evaluation. This could be a very potent evaluation tool for other new algorithms intended for the same problem domain.

The testbed system is called "Prefmine", since it facilitates data mining over the Preflib repository of datasets [Mattei and Walsh, 2013] (although the system could be readily extended to work with other online repositories). Prefmine is not the first system intended to mine preference data, though it fulfills a different niche than other methods. Web-based social choice systems like Pnyx [Brandt et al., 2015a], Spliddit [Goldman and Procaccia, 2015], Whale^3 [Bouveret], Democratix [Charwat and Pfandler, 2015] and RoboVote[1] provide user friendly implementations of social choice functions that may be difficult to implement or operate correctly, to assist with popularizing these techniques. Ordinary users can submit preferences to the systems and obtain results from sophisticated Condorcet extensions or other rules that are automatically selected according to expert knowledge, in order to fit the users' problem domain. The systems each offer some specialized benefits. For example, Democratix implements answer set programming [Gebser et al., 2011; Gelfond and Lifschitz, 1991] to select winners using voting rules that are NP-Complete, while Whale^3 boasts an extremely simple user interface. Preflib itself provides a set of associated tools for generating synthetic data for social choice [Matti], as well as data for matching domains [Dickerson et al., 2014].

Additionally there are several frameworks designed to facilitate machine learning over preference data. LPCforSOS [Hüllermeier and Fürnkranz, 2011] implements a binary classification approach similar to the one adopted in the initial implementation of the imputation-based approach to social choice (i.e. in Chapter 5). The more mature SVM-rank toolkit [Joachims, 2006, 2002] provides a means to apply the popular Support Vector Machine algorithm [Cortes and Vapnik, 1995] to learning rankings. There are also more general frameworks that allow the application and integration of several approaches, including tge Preference Learning Toolbox [Farrugia et al., 2015], a recent Java-based toolkit, and WEKA-LR [Balz and Senge], an extension for the popular Weka machine learning toolkit [Hall et al., 2009] that allows models to output rankings instead of classes during classification.

Prefmine should be understood as an integrated framework offering the features of a preference mining toolkit (i.e. learning algorithms capable of dealing with rankings) alongside features more typical of a toolkit for computational social choice (i.e. efficient implementations of voting rules, synthetic data generation routines). In this respect, it is novel compared with the approaches described above. Prefmine implements a combination of machine learning approaches similar to the label-ranking algorithms of LPCforSOS, alongside more general learning algorithms, and problem specific approaches like MMR that do not use machine learning, and instead optimize the social choice decision process directly. Additionally, Prefmine provides efficient implementations of several voting systems, and an extensible framework for implementing many more, alongside algorithms for the generation of synthetic data according to several common preference distributions. Prefmine should be viewed as complementary to existing systems, bringing together two different sets of functionality in a single package, while simultaneously providing seamless access to the data stored in Preflib itself.

[1]A forthcoming system with anonymous authors. See `robovote.org`.

6.1.1 System Design

Prefmine is designed both to facilitate experiments, and to ensure that they are conducted correctly. This broad design goal can be expanded into six objectives:

1. Data should be seamlessly obtained from Preflib, or other online repositories with similar formats.
2. The system should store the original data in a read-only format, so that experimenters cannot accidentally change it.
3. Problem instances should be easy to generate from both real-world and synthetic data.
4. It should be easy to run arbitrary combinations of social choice algorithms on a given problem instance, and easy to collect arbitrary performance measures.
5. The system should make use of all available computational resources (to avoid having experimenters implement their own error-prone parallelism).
6. The system should be easy for a new user to apply.

Seamlessly obtaining data from Preflib entails both fetching the desired datasets in a straightforward manner, and processing them from their current format in a representation more suitable for the imputation-based approach to social choice. It is important that this process be automated to ensure consistency, because manual processing can very easily introduce errors into the source data. For this reason also, Prefmine dynamically fetches data from the Preflib repository each time a new experiment is run, rather than storing it locally where it might be subject to (inadvertent) corruption by the experimenter. This is marginally slower than storing the data on a local disk, but the datasets on Preflib, and indeed, most social choice sets in general, are relatively small. They contain on the order of tens or at most hundreds of thousands of votes, and at most dozens of candidates. Preflib also uses a simple compression format to store the data more efficiently. The result is that fetching the data is a rapid step when modern network connections are utilized, lasting no more than a second or two.

Once the data has been obtained from Preflib, Prefmine stores it as an immutable data structure, using a language with true immutability (the `D` programming language). It is therefore not possible for an experimenter's own (novel) social choice algorithm to inadvertently modify the data, as attempting to do so will trigger a fault. This ensures that, for example, performance measures based on aggregates over a ground-truth ordering are never computed over data that has been modified by the methods under assessment. It also facilitates parallel processing of the original dataset, because there is no chance of race conditions arising from data being mutated.

Prefmine adopts an extensible framework to allow easy incorporation of new problem generation algorithms, social choice functions, imputation algorithms, and performance measures. The system uses a plugin-style architecture, where new features can be added without changing the core experiment, analysis, and data loading code at all, preserving the integrity of these components. The system also boasts a simple user interface featuring both a graphical mode and "headless" operation via the commandline. The interface also makes use of the plugin architecture, allowing users to seamlessly access newly developed functionality through the graphical interface.

Prefmine also seeks to use all computational resources available on the system upon which it is run. Modern commercial desktop computers, as well as servers, typically feature a multi-core architecture in which properly parallelized code can run an order of magnitude faster than code which runs sequentially. Importantly, parallelization is a highly error-prone process, and the subtle bugs it produces can yield slightly (or wildly) incorrect computations without giving off overt signs to the user like a program crash. Indeed, many development versions of Preflib exhibited these properties. With this in mind, Prefmine seeks to minimize the need for experimenters to consider parallelism when adding extensions to the system (e.g. new imputation methods, voting rules, etc.), and instead to parallelize the process inside the core experimental setup, making maximal use of computational resources. Although many of the results in this thesis were generated using a parallel version of Prefmine, recent updates to the core design of the D programming language (in which Prefmine is implemented) have necessitated a rewrite of the parallel processing code[2]. As a result, the current version of Prefmine uses only one CPU core at a time, and is significantly slower.

6.1.2 An Algorithmic Description of Prefmine

The previous few pages situated the Prefmine system within the context of other similar systems, and outlined the broad design goals of the system and the reasoning behind them. This subsection provides an algorithmic description of the core Prefmine system, the main experiment loop. The loop operates over a large set of parameters, which will be described

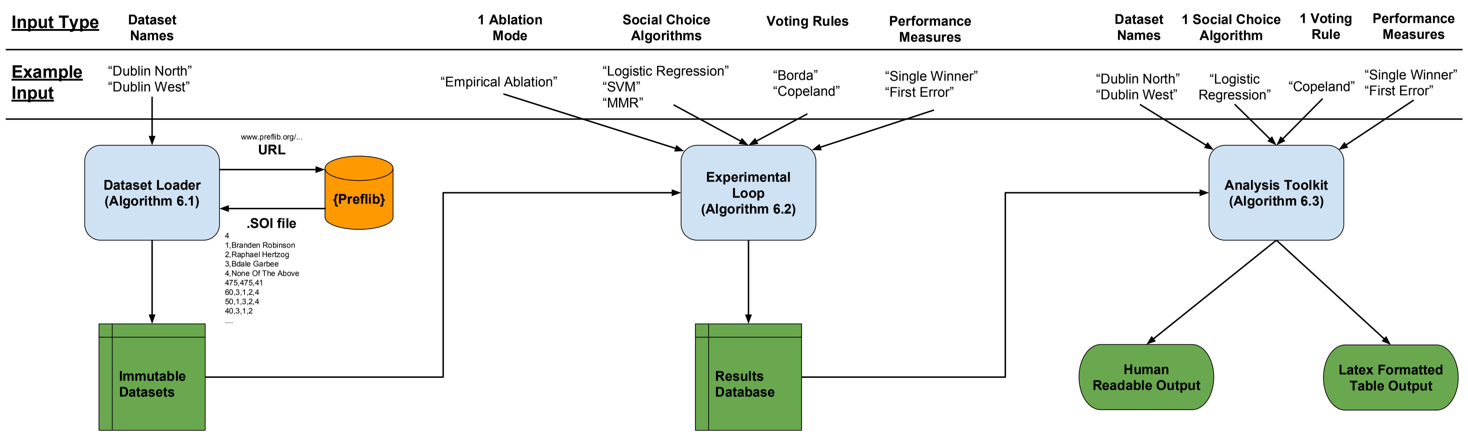
[2]In particular, the removal of semi-immutable maps has rendered much of the core Prefmine code non-threadsafe.

first. The remainder of this section within the chapter is dedicated to a "user manual"-style description of Prefmine, including both examples showing how the system is to be used, and explaining all of the possible settings present in the current version of the system. In many cases these settings are simply instantiations of algorithms presented earlier in the thesis, or in the existing research literature. A reader interested only in the general design of the system can safely skip Subsection 6.1.4 through to the start of Section 6.2. Later chapters may refer the reader back to these subsections when discussing the parameterization of subsequent experiments.

Prefmine's core consists of three components: a system for *loading* electoral data in .soi format [Mattei and Walsh, 2013] (i.e. one of several format's available for data from the Preflib repository, and a common native format for ballots) and storing them immutably; An *experimental loop* which can dynamically generate new problem instances from stored datasets, decide them using arbitrary combinations of imputation algorithms and voting rules, or using comparison algorithms like Minimax Regret, and assess the quality of the resulting decisions relative to the ground truth data; and an *analysis toolkit* that can produce both human readable and LaTeX-formatted summaries of the results from running the experimental loop. The three components are depicted together in Figure 6.1.

The Dataset Loader component of Preflib is summarized in Algorithm 9. The loader accepts a string corresponding to the name of a dataset, and looks up the appropriate URL on the Preflib website, obtaining a .soi file. The .soi format is a **S**trict **O**rder, **I**ncomplete list representation of the ballots in an election. The start of the file contains metadata: the number of candidates, a mapping from candidate names to numbers, and information about the total number of ballots in the set. After this, the file contains one line per *unique* ballot in the election. The first number on a line indicates the number of voters who cast this ballot. The remainder of the line is a comma-separated list of numbers, which are mapped to candidates according to the mapping in the metadata. Candidates that appear earlier (i.e. closer to the start of the line) in this list strictly precede those who come later. The list need not contain every candidate. The interpretation of unranked candidates depends on the election, but in many cases the original ballots were effectively interpreted as top orders. In its current implementation, Prefmine interprets all datasets obtained in .soi format as top orders. The runtime of the Dataset Loader algorithm is in $O(|B||C|)$

After obtaining a .soi file, the loader processes it into an immutable datastore. Candidate mapping is stored as an immutable hashmap. Each line is converted into a "Datum" data structure. A Datum is a map from positions to sets of candidates. If and only if the (partial) ordering implied by a given line could be completed such that a candidate c_i were placed in a position p_j, will the corresponding Datum structure return a set of candidates



```
4
1,Branden Robinson
2,Raphael Hertzog
3,Bdale Garbee
4,None Of The Above
475,475,41
60,3,1,2,4
50,1,3,2,4
40,3,1,2
....
```

Figure 6.1: A graphical depiction of the data flow in Prefmine, with example input arguments. Light blue boxes denote algorithms, presented in full in the text. Orange cylinders show external data stores. Green squares denote immutable internal data stores, output by one of the system's algorithms. Green ovals show the system's final output. Processing begins with the leftmost algorithm (the Dataset Loader), which downloads data from the Preflib repository, and then formats it as an immutable database. The immutable datasets are then passed to the middle algorithm (The Experimental Loop), which runs experiments on them according to its other input parameters, producing an immutable results database. The rightmost algorithm (The Analysis Toolkit) can be used to generate human readable and Latex-formatted tables by querying the results database. Queries are constructed from the input parameters. Existing valid parameter settings are discussed later in the text, but most parameter sets are easily extensible.

containing c_i when queried with key p_j. The loader's immutable datastore is called a "Data" instance, and contains metadata (the total number of ballots; the Candidate mapping), and a list of Datum structures corresponding to the list of orderings from the original .soi file. The entire Data instance is output as an immutable structure, meaning none of the components can be changed in any way. Data can only be read, not written.

Once the Dataset Loader has completed its task, the Experimental Loop algorithm (described in Algorithm 10) takes over, and performs a series of repetitions according to an experimental design specified by the passed parameters. Each iteration of the outermost loop is a single repetition of the experiment. A new problem instance is generated by making a mutable copy of the *complete* rows from an immutable Dataset instance that was generated by the Dataset Loader (Algorithm 9), and then ablating it according to the single Ablation Mode parameter's setting. This parameter can currently be configured to either ablate the data according to a distribution learned from the corresponding immutable Dataset instance, or a user-specified vector of probabilities. Both modes are discussed in the next subsection, along with examples of their use.

After the new problem instance has been generated, each member of the list of Social Choice Algorithms is given a deep copy of the problem instance. If the method relies on the generation of imputations, it generates an imputation, and each of the listed Voting Rules is run on the resulting imputation to generate a set of orderings (one per voting rule). If the method does not rely on the generation of an imputation (e.g. Minimax Regret), then it is instead called with each voting rule in turn as an argument to obtain these orderings. Subsection 6.1.5 discusses the implementation details of this part of the system and shows example uses. Additionally, each of the listed voting rules is run once on the true-preferences of the problem instance to obtain the "correct" orderings. Finally, every member of the list of Performance Measures is applied, comparing each ordering produce by a Social Choice Algorithm under each voting rule, to the corresponding "correct" ordering. The results are stored in an immutable database under a composite key that encodes the dataset, Social Choice Algorithm, Voting Rule, and Performance Measure that were used, as well as the repetition number from the main loop. The entire main loop is then repeated. In the final step of the algorithm, the output database is rendered immutable in the final step to ensure that the data are not tampered with by the system inadvertently[3]. The total runtime of the algorithm is highly dependent on the runtimes of the parameters it is passed. If k datasets, all smaller than $|B|$ ballots over $|C|$ candidates are passed, and l voting rules, all taking less than $O(S(B))$ time to evaluate over the most complex dataset, and m Social Choice Algorithms all taking less than $O(M(B))$ time, and n performance

[3]In practice the database is written to disk, and the files marked as read only after the run is complete.

Algorithm 9 An algorithmic description of the Dataset Loader component of the Prefmine system. The Dataset Loader accepts a string corresponding to the name of a dataset, and then downloads the corresponding .soi file from the Preflib repository [Mattei and Walsh, 2013]. The file is then processed into an immutable Data instance which is produced as output.

```
procedure DatasetLoader(DatasetName)
    Let url $\leftarrow$ lookupURL(DatasetName)
    Let $S$ be a stream obtained by opening url.
    Let $|C| \leftarrow$ S.nextLine()
    Let CandMap $= \emptyset$
    for i $= 0$; $i < |C|$; i++ do
        Let {key, name} $=$ split(S.nextLine(),",")
        Let CandMap[key] $=$ name
    end for
    // The last line of the metadata contains the number of ballots.
    Let $|B| \leftarrow$ split(S.nextLine(), ",")[0]
    Dataset output $= \emptyset$
    while S.hasNextLine() do
        //Each remaining line is parsed into a top order.
        tokens $\leftarrow$ split(S.nextLine(),",")
        numBallots $\leftarrow$ tokens[0]
        Datum d $= \emptyset$
        notAssigned $=$ CandMap.keys()
        for i=1; $i <$ |tokens|; i++ do
            d[i] $=$ tokens[i]
            notAssigned $\setminus=$ tokens[i]
        end for
        for i= |tokens|; i $<$ |C|; i++ do
            d[i] $=$ notAssigned
        end for
        for i $= 0$; i $<$ numBallots; i++ do
            output $\leftarrow$ append(output, d.duplicate())
        end for
    end while
     return cast(Immutable Dataset) output
end procedure
```

measures, all taking less than $O(P(o_1, o_2))$ time, are passed, then the total runtime will be in $O(\text{NumReps} \cdot k(\text{ablate}(B) + ml(S(B) + M(B) + n)))$

The final component of the Prefmine system is the Analysis Toolkit, an algorithm that allows the analysis of results generated by the Experimental Loop. The toolkit allows the user to specify an output database from a run of the Experimental loop, a social choice algorithm and a voting rule, as well as a list of performance measures. The toolkit then generates a tabular summary of the mean performance of that social choice algorithm under that voting rule, with respect to each performance measure. Sample standard deviations are also reported. The table is output both in a human-readable format, and as a Latex tabular environment that can be dropped into a document with minimal editing. The Analysis Toolkit's behaviour is summarized in Algorithm 11. The algorithm's runtime is in $O(\text{NumReps}\cdot kn)$, where k is the number of datasets, and n is the number of performance measures, assuming that printing is a constant time operation, and NumReps is the largest number of entries in the database for any single key.

This concludes the presentation of the Prefmine system at a high level. Note that the algorithms as presented contain some inefficient components to facilitate their presentation. For example, Algorithm 10 repeatedly recomputes Alg(Problem.expressedPrefs.copy()), when in practice this is computed once and cached. Additionally, the algorithms are all presented as sequential when (at least in earlier versions of the system) parallel processing took place. In the past, parallel processing took place within individual algorithms. For example, when training a chained classifier based on logistic regression, many classifiers could be trained in parallel from the same dataset. After changes in the implementation language took place during 2015, this feature ceased to function properly and was removed. Future versions of Prefmine will likely include parallelism in the outermost loop of Algorithm 10 (i.e. the Reps loop) instead. Although this is not always maximally efficient, it will ensure thread safety and should produce a significant speedup when the slowest social choice algorithms are run. Additional parallelism was previously present in the computation of the social choice functions, which adopted a map-reduce paradigm [Dean and Ghemawat, 2008] to efficiently compute the outcome.

The remainder of this section serves as a reference manual for Prefmine, describing its use, features, and how to extend the existing code base, with examples. The features and usage are both presented via the graphical user interface.

Algorithm 10 An algorithmic description of the Experimental Loop component of the Prefmine system. The Experimental Loop accepts an Immutable Dataset (produced using Algorithm 9), an ablation mode, a list of social choice algorithms (e.g. imputation-based, Minimax Regret), a list of voting rules (e.g. **Borda**, **Copeland**, a list of performance measures (e.g. Single Winner Error, First Error Location), and a number of repetitions for the experiment. For each repetition of the experiment, a new problem instance is generated using the specified ablation mode. Then every social choice algorithm is run under every voting rule to produce an outcome. The outcome is compared to the ground truth outcome for the voting rule under consideration using every performance measure. The results are written to an output database, which is rendered read-only as the final step.

```
procedure ExperimentalLoop(ImmutableDatasets, AblationMode, ListOfSCAlgs,
ListOfVotingRules, ListOfPrefMeasures, NumReps)
    Let Output = $\emptyset$
    for Rep = 0; Rep < NumReps; Rep++ do
        for all Dataset in ImmutableDatasets do
            Let Problem = AblationMode(Dataset.copy())
            for all Alg in ListOfSCAlgs do
                for all Rule in ListOfVotingRules do
                    if Alg uses Imputation then
                        Outcome = Rule(Alg(Problem.expressedPrefs.copy()))
                    else
                        Outcome = Alg(Problem.expressedPrefs.copy(), Rule)
                    end if
                    CorrectOutcome = Rule(Problem.truePrefs)
                    for all Measure in ListOfPerfMeasures do
                        Key = concatenateNames(Dataset, Alg,Rule,Measure,Rep)
                        Output[Key] = Measure(Outcome, CorrectOutcome)
                    end for
                end for
            end for
        end for
    end for
     return cast(Immutable) Output
end procedure
```

Algorithm 11 An algorithmic description of the Analysis Toolkit component of the Prefmine system. The Analysis Toolkit accepts an output database produced by the Experimental Loop component of the system (Algorithm 10). The user also specifies a particular social choice algorithm and voting rule, as well as a list of datasets and performance measures. The toolkit computes a table where each row corresponds to a dataset, and each column to a performance measure. The value in a particular table cell will be the mean and standard deviation of the selected social choice algorithm under the selected voting rule, with respect to the corresponding performance measure (i.e. column) and dataset (i.e. row). The resulting table is then output both in a human readable format and as a Latex tabular environment.

```
procedure AnalysisToolkit(OutputDatabase, SCAlg, VotingRule, ListOfPrefMeasures, ListOfDatasets)
    Let Table = $\emptyset$
    for all Dataset in ListOfDatasets do
        for all Measure in ListOfPerfMeasures do
            Key = concatenateNames(Dataset,SCAlg,VotingRule,Measure, *)
            Results = OutputDatabase[key]
            Mean = computeMean(Results)
            Stdev = computerStandardDeviation(Results)
            Table[Dataset][Measure]["mean"] = Mean
            Table[Dataset][Measure]["stdev"] = Stdev
        end for
    end for
    Print(Table)
    PrintLatex(Table)
end procedure
```

6.1.3 Using Prefmine

Prefmine is implemented in the D programming language. To compile Prefmine, users must install the Digital Mars D compiler `dmd`[4], and the `dub` package manager[5]. After obtaining the Prefmine source code from the author, users can run `dub` in the top-level directory of the project source to compile and execute the code. `dub` will automatically fetch and install all other required libraries. `dub` will produce an executable named `prefmine` in the top-level directory of the project source code after a successful compilation. Running the `prefmine` executable will open a small window, shown in Figure 6.2. Users can elect to either start a new experiment or run the analysis toolkit, by selecting the corresponding options from the drop down menu, as shown.

Starting a new experiment will display the window shown in Figure 6.3, which allows the user to configure which datasets will be loaded, and to configure the inputs to both the Dataset Loader and Experimental Loop portions of the system. The configuration of the experiment is detailed later in this section. Additionally, the user can configure the directory to write the experimental loop's output database to, and indicate whether to overwrite or append to any existing database present at that location. After selecting the settings they prefer (e.g. Figure 6.4), the user presses the "Create" button. Datasets are loaded, and the experimental loop begins running. Two progress bars in the lower right indicate the progress of the experimental loop, and the large textbox provides a summary of this progress (Figure 6.5). When the experiment is complete, the user should close the application.

The process for analysis of the data is similar. Instead of selecting "Experiment" from the File menu at the start, one selects "Analysis". A very similar window will be constructed, displayed in Figure 6.6. Radio buttons limit the user to the selection of a single imputation method and a single voting rule, while the familiar checkbox-style interface allows the selection of multiple performance measures. All datasets stored in the results database at the specified path will be presented in the results table.

6.1.4 Dataset Generation

In Prefmine, the selection of datasets for the Dataset Loader to fetch is accomplished through a dropdown menu at the top of the experiment window, as seen in Figure 6.7. The menu is used to avoid any possible confusion on the part of the experimenter about

[4] http://www.digitalmars.com/d/1.0/dmd-linux.html

[5] For Debian-based operating systems: http://d-apt.sourceforge.net/

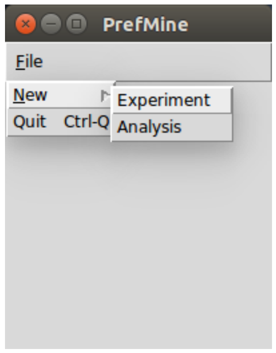
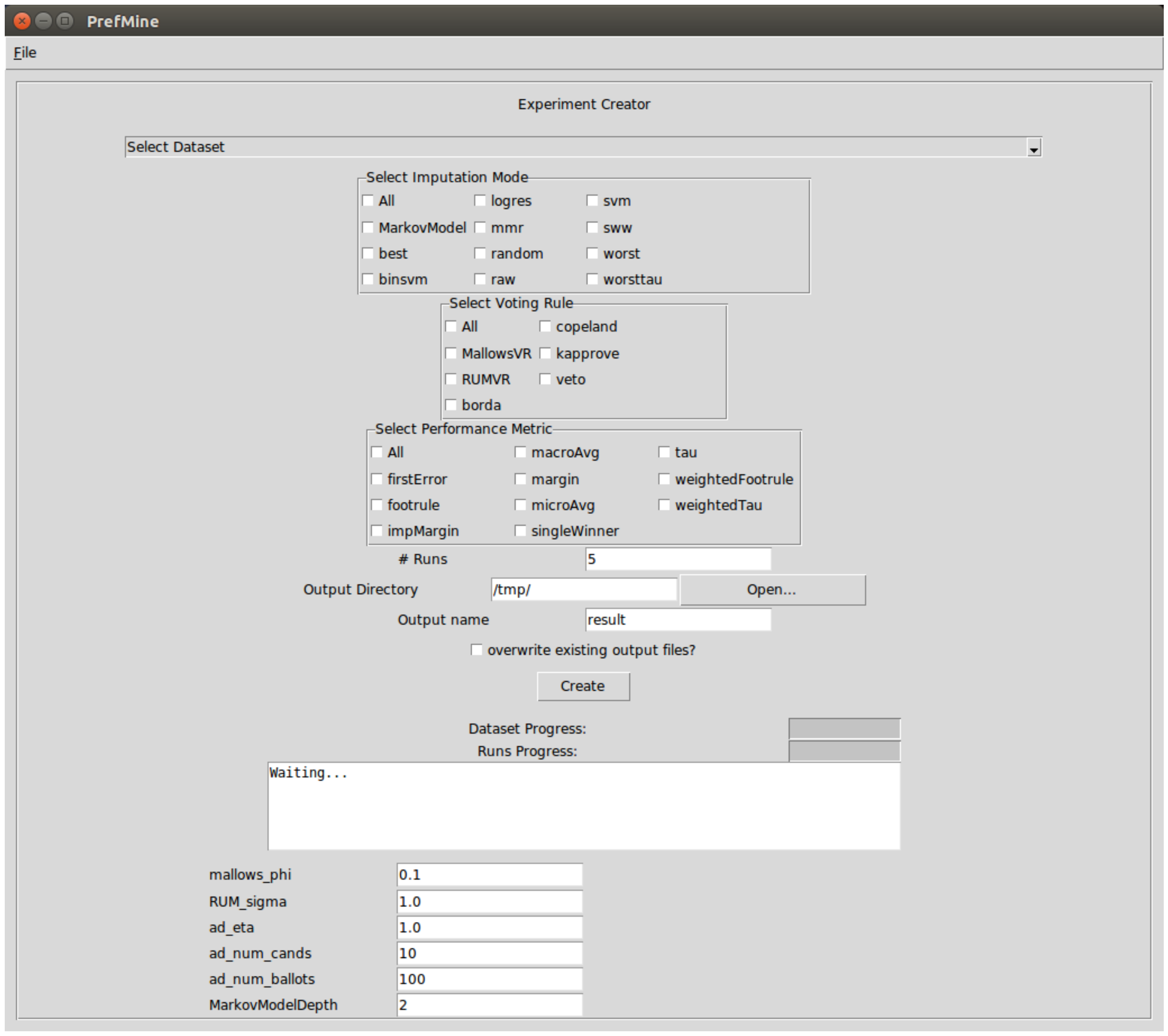


Figure 6.2: The initial Prefmine window, from which users can launch a new experiment, or run the analysis toolkit.

which dataset an experiment is being run on: selection of the specified name ensures that data is fetched from the corresponding URL. The menu is populated dynamically from the Preflib repository's website, so all available datasets are listed[6]. In addition to each individual set of electoral data, it is possible to run a Prefmine experiment on any set of related elections at one, if they are stored together on Preflib as a single collection of data. For example, the 7 Debian elections and 3 Irish elections are grouped together, and so experiments can be run on these 3 or 7 sets all at once. Additionally, two synthetic dataset generators are listed in the dropdown menu: RUM and NoisedMallows. The RUM option is an implementation of a Random Utility Model [Luce, 1959] for generating problem instances, based on Azari et al. [2012]. This model assumes that each voter's utility for a given candidate is sampled from a Gaussian distribution. All voters sample utilities from the same Gaussian distribution for a particular candidate, but each candidate has its own distribution. The Gaussian distribution for each candidate has mean sampled uniformly from $(0, |C|)$, and standard deviation equal to the `RUM_sigma` parameter setting, which is configured near the bottom of the experiment window. A voter's true preferences are implied by the utility they have sampled for each candidate: the candidate with the highest sampled utility is ranked first, and the candidate with the lowest sampled utility is ranked last.

The other synthetic data generator is simply a standard Mallows model [Mallows, 1957]. The central ordering is a permutation of the candidates selected uniformly at random. The dispursion parameter ϕ is set using the `mallows_phi` parameter near the bottom of the

[6]In the example windows, this feature has been disabled by the author to avoid searching through such a large list.



Experiment Creator

Select Dataset

Select Imputation Mode

☐ All	☐ logres	☐ svm
☐ MarkovModel	☐ mmr	☐ sww
☐ best	☐ random	☐ worst
☐ binsvm	☐ raw	☐ worsttau

Select Voting Rule

☐ All	☐ copeland
☐ MallowsVR	☐ kapprove
☐ RUMVR	☐ veto
☐ borda	

Select Performance Metric

☐ All	☐ macroAvg	☐ tau
☐ firstError	☐ margin	☐ weightedFootrule
☐ footrule	☐ microAvg	☐ weightedTau
☐ impMargin	☐ singleWinner	

# Runs	5
Output Directory	/tmp/
Output name	result

☐ overwrite existing output files?

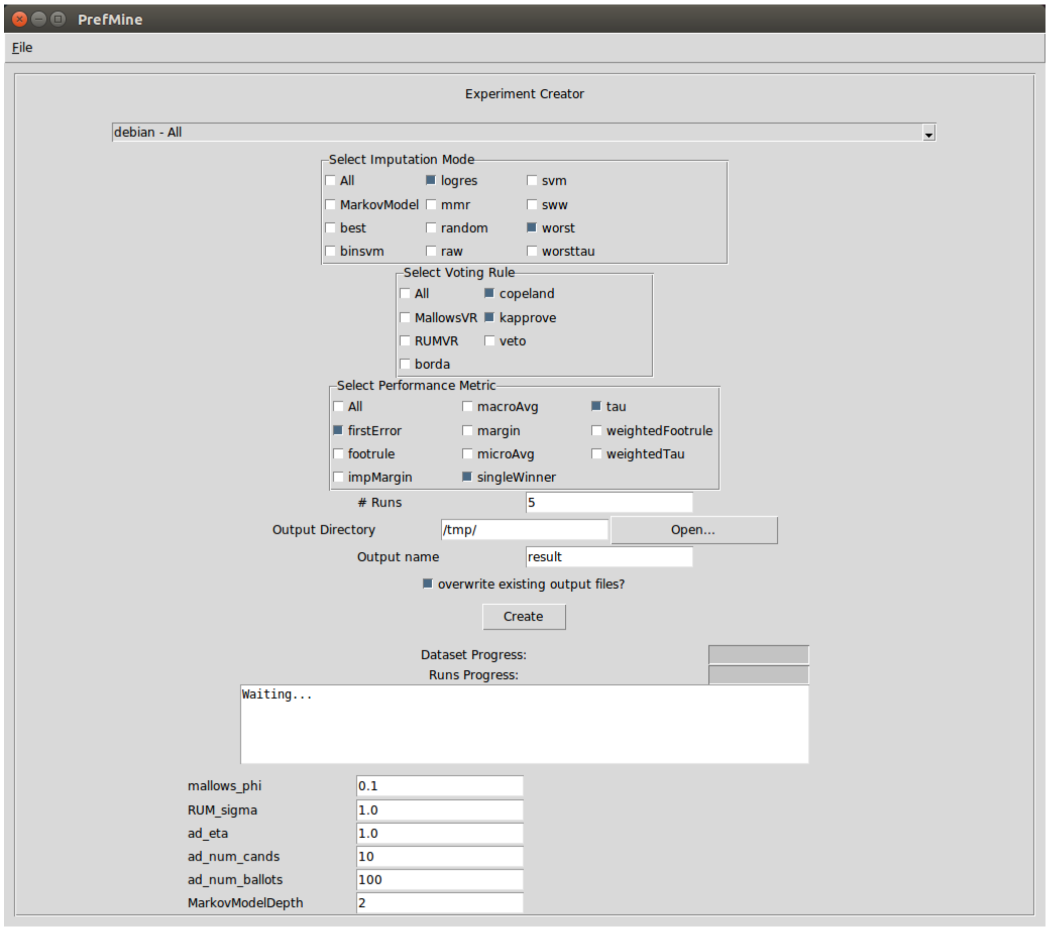
Dataset Progress:

Runs Progress:

```
Waiting...
```

mallows_phi	0.1
RUM_sigma	1.0
ad_eta	1.0
ad_num_cands	10
ad_num_ballots	100
MarkovModelDepth	2

Figure 6.3: The experiment configuration window in Prefmine, from which the user can configure input parameters for both the Dataset Loader and Experimental Loop portions of the system.



Experiment Creator

debian - All

Select Imputation Mode

All	[✓ logres]	svm
MarkovModel	mmr	sww
best	random	[✓ worst]
binsvm	raw	worsttau

Select Voting Rule

All	[✓ copeland]
MallowsVR	[✓ kapprove]
RUMVR	veto
borda	

Select Performance Metric

All	macroAvg	[✓ tau]
[✓ firstError]	margin	weightedFootrule
footrule	microAvg	weightedTau
impMargin	[✓ singleWinner]	

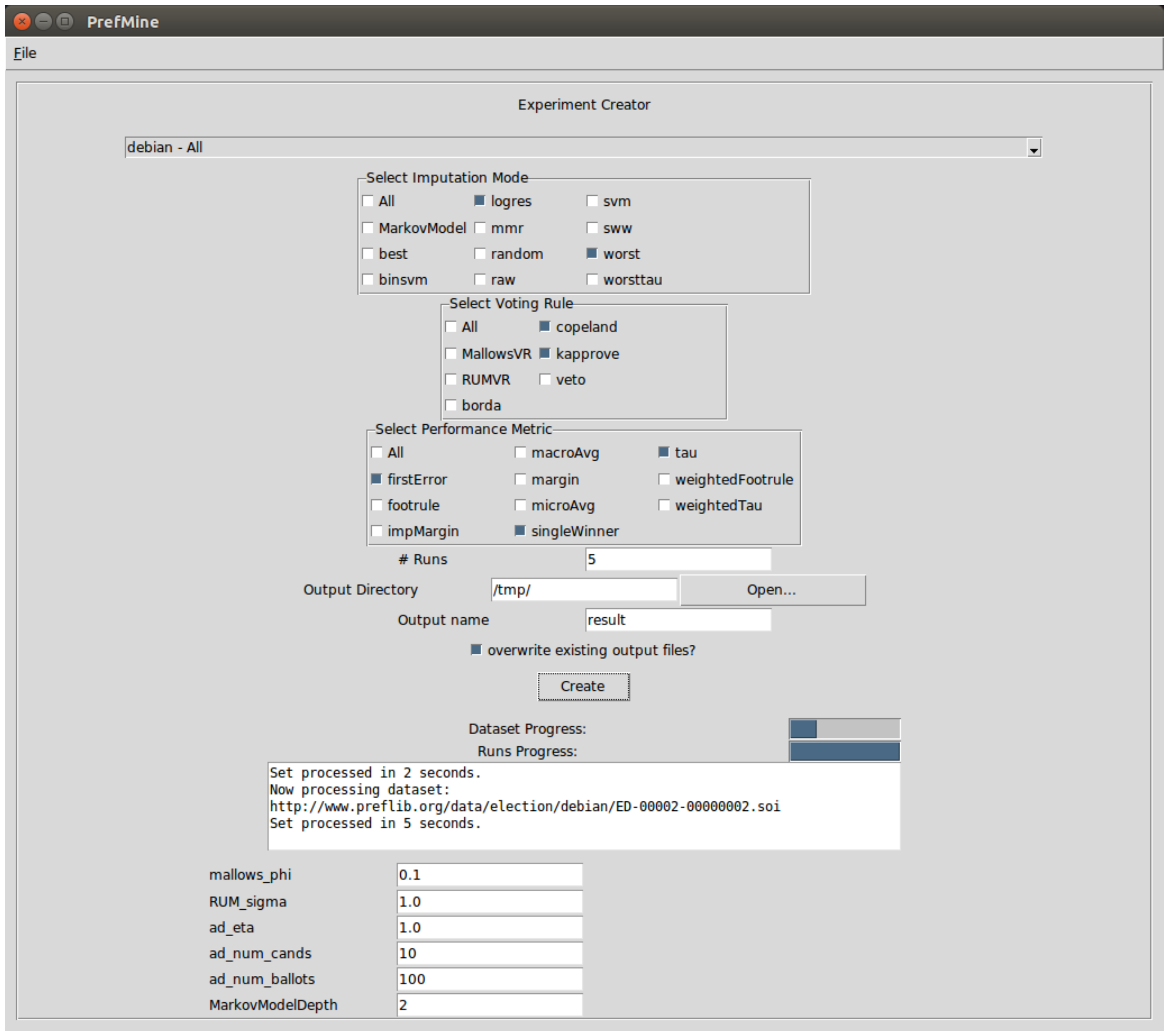
# Runs	5
Output Directory	/tmp/
Output name	result

[✓ overwrite existing output files?]

```
Waiting...
```

mallows_phi	0.1
RUM_sigma	1.0
ad_eta	1.0
ad_num_cands	10
ad_num_ballots	100
MarkovModelDepth	2

Figure 6.4: An example of an experiment configured using Prefmine's experiment configuration window. The experiment will run over the seven Debian datasets from the Preflib repository. Logistic Regression and a Worst Case Imputation approach will be applied to each problem instance, and the **Copeland** voting rule will be used to decide outcomes. Performance will be assessed under the Single Winner Error, First Error Location, and Kendall Correlation (τ) performance measures. 5 replications will be performed, and the output database will be stored in /tmp/, overwriting any existing results there. Since more than one dataset is being processed, the output filename parameter is ignored. The parameters below the output textbox ("Waiting...") are used to configure synthetic data generators or imputation methods that are not used, and so are ignored. Pressing the "Create" button will start the experiment.



Experiment Creator

debian - All

Select Imputation Mode

All [✓ logres] svm
MarkovModel mmr sww
best random [✓ worst]
binsvm raw worsttau

Select Voting Rule

All [✓ copeland]
MallowsVR [✓ kapprove]
RUMVR veto
borda

Select Performance Metric

All macroAvg [✓ tau]
[✓ firstError] margin weightedFootrule
footrule microAvg weightedTau
impMargin [✓ singleWinner]

# Runs	5
Output Directory	/tmp/
Output name	result

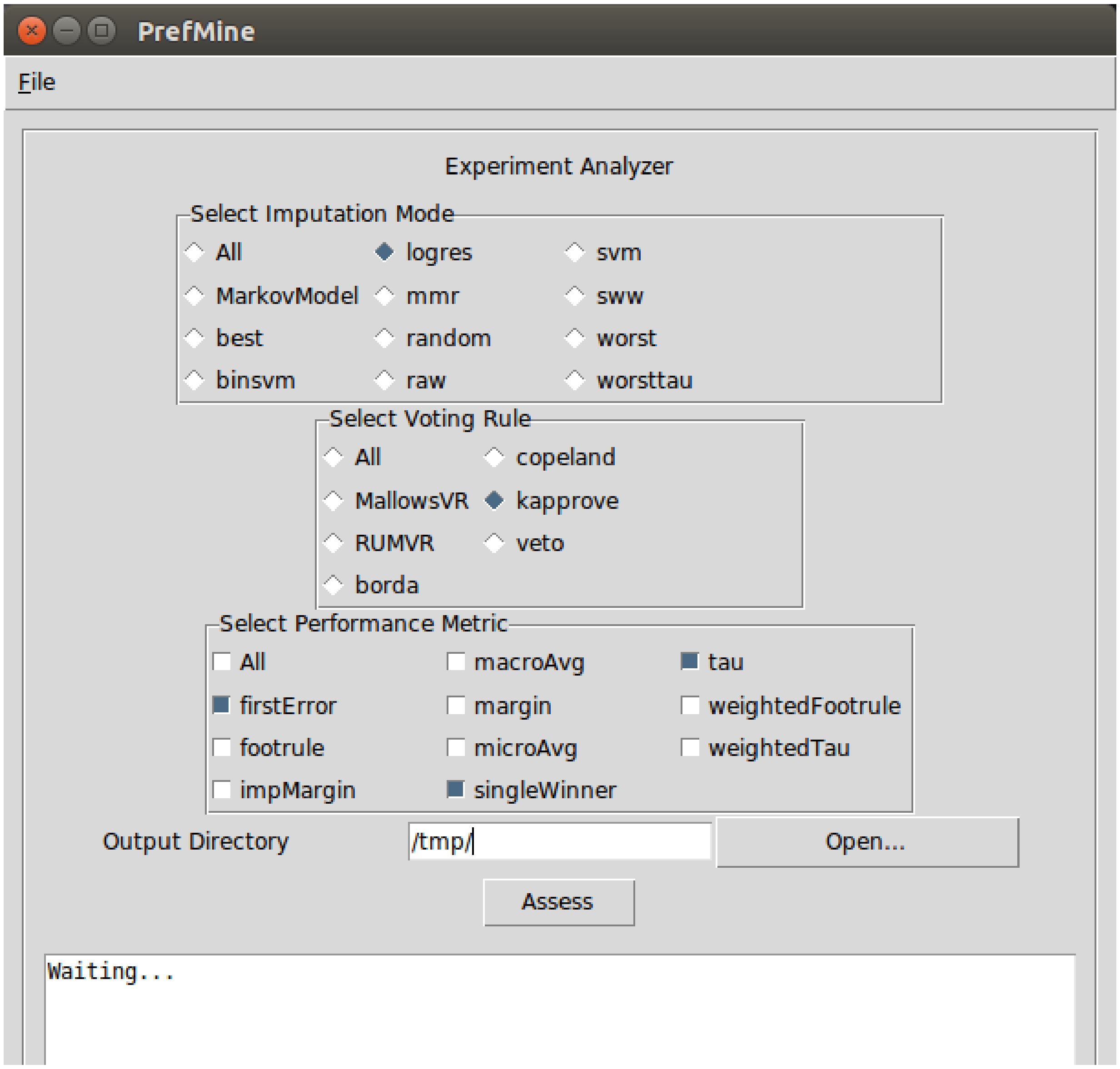
[✓ overwrite existing output files?]

Dataset Progress:
Runs Progress:

```
Set processed in 2 seconds.
Now processing dataset:
http://www.preflib.org/data/election/debian/ED-00002-00000002.soi
Set processed in 5 seconds.
```

mallows_phi	0.1
RUM_sigma	1.0
ad_eta	1.0
ad_num_cands	10
ad_num_ballots	100
MarkovModelDepth	2

Figure 6.5: An example of a Prefmine experiment in progress, using the settings from Figure 6.4. Note the progress bars showing the fraction of datsets processed (top) and runs completed on this dataset (bottom). The Textbox summarizes progress, including runtimes required to complete each dataset.



Experiment Analyzer

Select Imputation Mode

All	[✓ logres]	svm
MarkovModel	mmr	sww
best	random	worst
binsvm	raw	worsttau

Select Voting Rule

All	copeland
MallowsVR	[✓ kapprove]
RUMVR	veto
borda	

Select Performance Metric

All	macroAvg	[✓ tau]
[✓ firstError]	margin	weightedFootrule
footrule	microAvg	weightedTau
impMargin	[✓ singleWinner]	

Output Directory /tmp/

```
Waiting...
```

Figure 6.6: The Prefmine analysis window. The window is similar to the experiment window, but with a reduced set of options, and larger space to view the output.

experimenter window. Synthetic ballots are sampled from the Mallows model using a re-implementation of the efficient algorithm from [Lu and Boutilier, 2011a].

Both the Mallows and RUM synthetic data generators share a number of parameters set within the Prefmine experiment window. Collectively these are called the *artificial data* parameters, and are denoted with the prefix `ad_`. All artificial data parameters must be set in order to use a synthetic data generator. The `ad_num_cands` and `ad_num_ballots` parameters respectively specify the number of candidates who will compete in the synthetic election, and the number of ballots that will be cast in the election. The `ad_eta` parameter allows the construction of simple top-ordered ballots, rather than the totally-ordered ballots that are generated by the two methods. The probability of generating a top-ordered ballot of length at least k is given by η^{k-1} (i.e. in expectation a fraction η of ballots rank at least two preferences, η^2 rank at least 3, and so on).

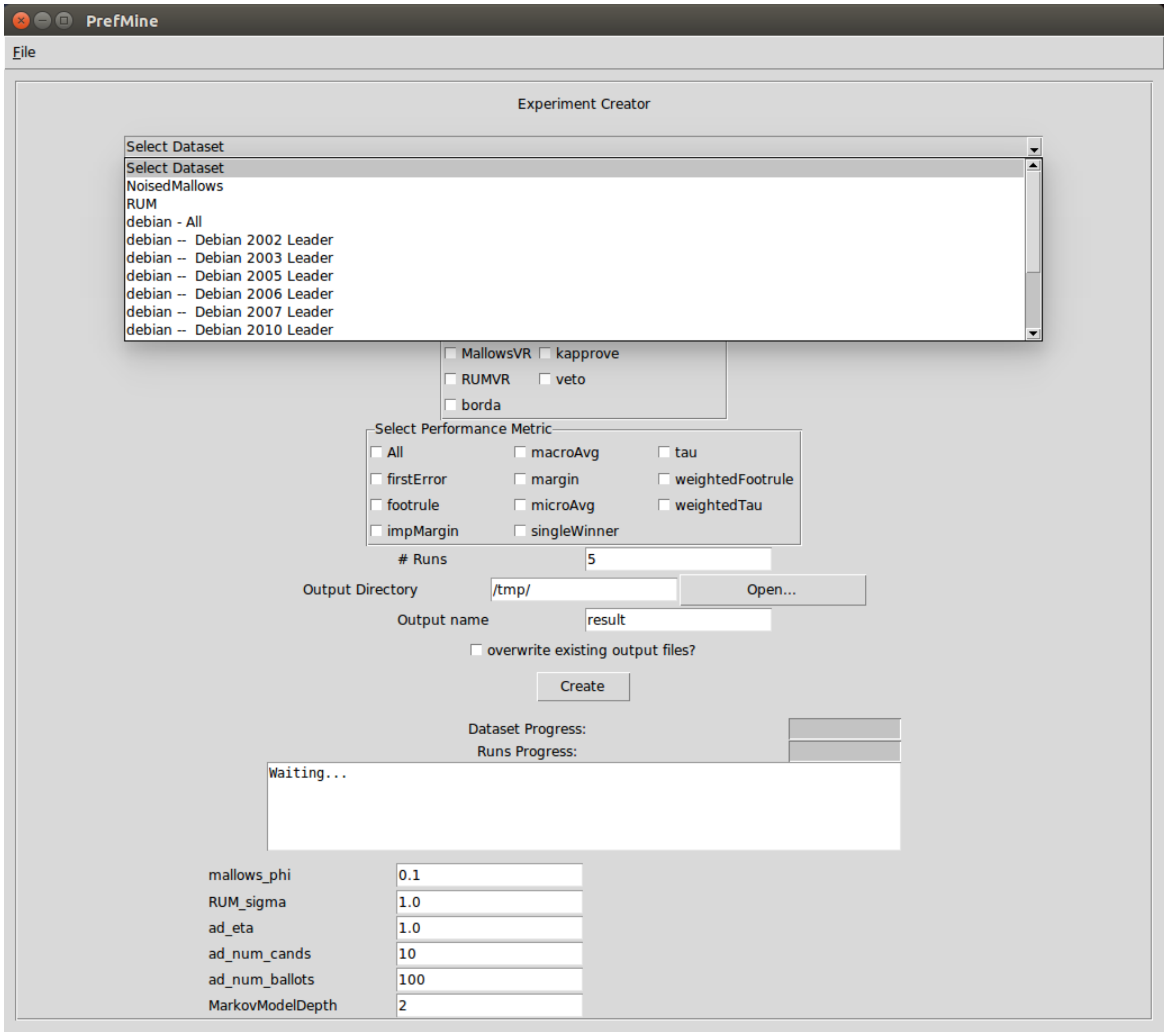
In addition to selecting datasets, Prefmine offers two methods for ablating datasets to generate problem instances[7]. In the first, a user specifies a cumulative density function for the probability that a ballot has at least k candidates ranked, for every $1 < k \leq |C|$, and the system ablates the ballots such that this distribution is observed in expectation. In the other, such a distribution is learned empirically from the original dataset, and the ablation process is identical.

6.1.5 Imputation Modes

After selecting a dataset or synthetic data generation method, the Prefmine user should select one or more "Imputation Modes" from the array of checkboxes located directly beneath the dataset selection menu. To select more than one imputation mode at a time, the user simple checks the boxes beside multiple methods. Figure 6.8 shows a closeup of the array of checkboxes, which currently includes twelve different methods. Some of these are actually comparison algorithms and do not use imputation, despite the name. The details of the methods are itemized below.

- **All**: Behaves identically to selecting every other checkbox in the array.
- **MarkovModel**: This is the Markov-Tree based approach described in Chapter 7.
- **best**: The "best case" imputation method. The method peeks at the true preferences of users, and imputes each expressed ballot with the correct values. It can be useful

[7]The interface shown in this chapter does not allow selection between them, as this created additional clutter. The choice is easily made using the commandline interface to Prefmine.

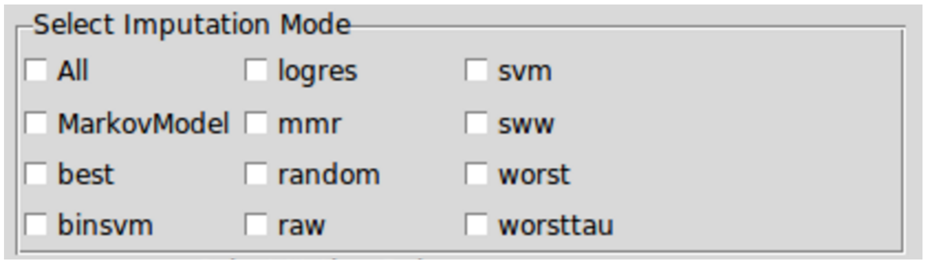


# Runs	5
Output Directory	/tmp/
Output name	result

```
Waiting...
```

mallows_phi	0.1
RUM_sigma	1.0
ad_eta	1.0
ad_num_cands	10
ad_num_ballots	100
MarkovModelDepth	2

Figure 6.7: The location of the dataset selection dropdown menu in Prefmine's experiment window.


Select Imputation Mode

☐ All	☐ logres	☐ svm
☐ MarkovModel	☐ mmr	☐ sww
☐ best	☐ random	☐ worst
☐ binsvm	☐ raw	☐ worsttau

Figure 6.8: The Imputation Method selection box in Prefmine's experiment window.

to detect whether ties are present in the orderings produced by the true preferences of users, an especially common problem when the **Copeland** voting rule is used.

- **binsvm**: The standard imputation-based approach to social choice described in Chapter 4, using a binary Support Vector Machine [Cortes and Vapnik, 1995] as the base classifier. A set of SVMs are learned to predict the candidate that should be imputed at each position. Since this is a multi-class classification problem, one SVM is trained to predict membership in each class, a one-versus-all approach [Rifkin and Klautau, 2004]. Contrast with the **svm** option, which uses a one-versus-one approach. The individual SVM models are trained using the popular libsvm C++ implementation [Chang and Lin, 2011], which is linked directly with Prefmine's D code via a custom interface. Parameters for the SVM are selected using cross validation, with Linear, Polynomial (degree 3), and Radial Basis Function kernels considered alongside every combination of parameter values for $C \in (2^{-5}, 2^{15})$ and $\gamma \in (2^{-15}, 2^{5})$, spaced at equal *factors* of 2^4 (i.e. 2^{-15}, 2^{-11}, 2^{-7}, and so on). In the current implementation this can take a tremendous amount of time compared with other imputation methods. A subsampling approach maybe implemented in future to remedy this.

- **logres**: The standard imputation-based approach to social choice described in Chapter 4, using logistic regression as the base classifier. This is identical to the model described in Chapter 5, including the choice of features, feature selection algorithms, and parameterization. The implementation is a custom version of the conjugate gradient descent algorithm [Hestenes and Stiefel, 1952] designed to operate efficiently over the standard Prefmine data format.

- **mmr**: An implementation of the Minimax Regret algorithm [Lu and Boutilier, 2011b], used as a comparison method and described in Chapter 5. This implementation handles partial orders when used with scoring vector-based voting rules (e.g. **Borda**), but only with top-orders on **Copeland**.

- **random**: The randomized imputation approach described in Chapter 5, in the spirit of the MLE approach [Xia and Conitzer, 2008]. Starting from the top of the ballot, each position that could be held by more than one candidate is considered (i.e. each position n where the voter has not definitely ranked a candidate in n^{th} place). A candidate is selected uniformly at random from the set of candidates eligible for this position, and this candidate is ruled out for all other positions. The process is then repeated until every candidate has been assigned to a position. For top-orders, this selects a suffix for each ballot uniformly at random.

- **raw**: No imputation is performed. The ballots are passed directly through to the voting rules, in their original, incomplete, forms. This is useful primarily when testing a new ablation model, but could also be used to decide elections directly as a comparison.

- **svm**: As binsvm above (including parameter selection procedures and runtimes), but using libsvm's default one-versus-one approach to multiclass classification instead of the one-versus-all approach used by binsvm and logres.

- **SWW**: Single Winner Worstcase: A comparison method like MMR that computes the set of *possible* winners, per [Xia and Conitzer, 2008], and assigns a score of 1 to members of the set, and 0 to all other members. Essentially it computes an upper bound on the worst possible single winner distance. Current implementations work only for monotonic scoring rules, since other voting rules are NP-Hard to compute possible winners for.

- **worst**: Peeks at voters' true preferences, and imputes the *reverse* ordering (i.e. each candidate goes as far from its true position as possible, while still producing an ordering consistent with the voter's expressed preferences).

- **worsttau**: The system used as a benchmark for the difficulty of different datasets in Chapter 5. It imputes each ballot with the opposite of the correct *aggregate ordering* under **Borda**. That is, with the opposite of the ordering that is returned by **Borda** under consideration when run on the true preferences of the voters, which the method peeks at. Contrast with **worst** and **SWW** above. This creates a profile where candidates that were ranked highly *on average* are ranked low consistently, and vice-versa. It is a good heuristic approximation of the SWW method for many voting rules, and is a lower-bound on the worst possible distances, rather than an

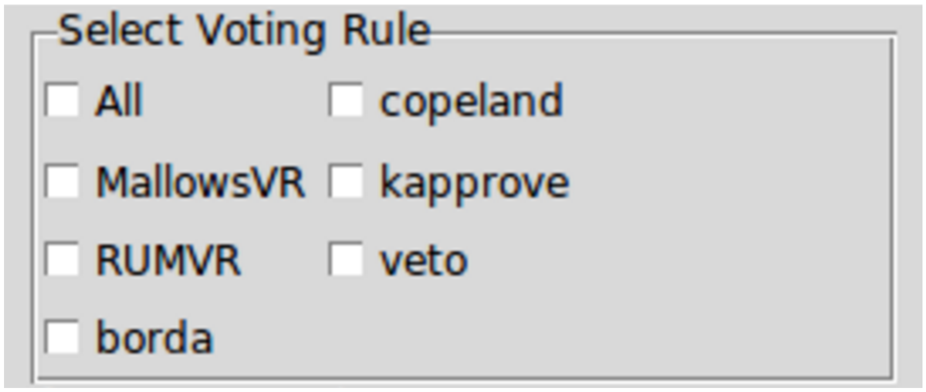


Figure 6.9: The Voting Rules selection box in Prefmine's experiment window.

upper bound (i.e. there may exist profiles that are even worse, especially for non-monotonic rules).

Extending the set of "Imputation Methods" is a straightforward proposition. An imputation method must accept a profile of incomplete ballots and produce a completed one. A comparison method must accept both a profile of incomplete ballots and a list of voting rules. It must output a score for each candidate under each rule, such that candidates with higher scores under a given rule come before those with lower scores. In both cases, after writing a function satisfying the above constraints, users can add its name to a list which is used to dynamically generate the box depicted in Figure 6.8. The system will automatically generate appropriate calls to the function in the experimental loop, storage of its results, and so on.

6.1.6 Voting Rules

After picking one or more imputation methods, Prefmine users should pick one or more voting rules for their experiments. The Voting Rule selection box is an array of checkboxes directly below the Imputation Methods box. The current implementation of Prefmine contains seven voting rules, two of which are used only for synthetic data. As with the Imputation Method box, a user who wants to sequentially run several voting rules on the same data simply checks more than one box. The Voting Rule selection box is shown in Figure 6.9. The various options are itemized below.

- **All**: Equivalent to checking all other boxes.
- **MallowsVR**: A "voting rule" to use in conjunction with synthetic data generated by the Mallows Model. When selected, models with the ability to estimate aggregated rankings directly (like the Markov Model of Chapter 7, or MMR) will have their estimated rankings directly compared to the true central ranking of the Mallows that was used to generate the current problem instance, rather than using an ordinary voting rule. Activating this function when a dataset other than **NoisedMallows** is selected will trigger a fault.
- **RUMVR**: A "voting rule" to use in conjunction with synthetic data generated by the Random Utilty Model. When selected, models with the ability to estimate aggregated rankings directly (like the Markov Model of Chapter 7, or MMR) will have their estimated rankings directly compared to the true central ranking of the RUM that was used to generate the current problem instance, rather than using an ordinary voting rule. Activating this function when a dataset other than **RUM** is selected will trigger a fault.
- **borda**: An implementation of the **Borda** count voting rule. A candidate receives exactly k points for appearing ahead of k other candidates on a ballot.
- **copeland**: An implementation of the **Copeland** voting rule. A candidate receives 1 point for each pairwise runoff contest it wins against other candidates.
- **kapprove**: An implementation of $\frac{|C|}{2}$-approval. A candidate receives 1 point for appearing in the top half of the positions on a ballot (round down), and 0 otherwise.
- **veto**: An implementation of the Veto voting rule. A candidate receives 1 point for every ballot on which it is not ranked last.

Note that none of these voting rules express behaviour in the case of ties. This is because when Prefmine is used for experimental purposes, detection of ties is important (e.g. if the correct ordering contains no ties, but the winning candidate selected by an imputation method was tied with someone else, then this result may have a different interpretation even if the method has picked the correct winner via tie-breaking). Additionally, all voting rules assign a score to each candidate. This is to facilitate measurement of the errors in candidate's scores under rules like Borda. Voting rules that are not based on assigning a score to each candidate can simply assign each candidate a score equal to the number of candidates behind them in whatever ranking is decided upon. Indeed, this is precisely what is done for the current implementations of the MallowsVR and RUMVR rules.

Additionally, note that the implementations of these rules use a simple Map-Reduce [Dean and Ghemawat, 2008] approach for the borda, kapprove, veto, and Copeland options. In each case, since the ballots in a profile express independent information, the set of ballots can be partitioned into independent subsets, one for each available CPU core. Aggregate scores can be computed (in the case of Copeland, these are computed for a given pair-wise contest, not the overall ranking) by separate threads for each subset. The aggregate scores for each thread can then be combined to obtain the final result. This feature was present in an earlier version of Prefmine, but has been disabled following changes in the implementation language in 2015.

As with the addition of new Imputation Methods, Prefmine's plugin-style architecture allows users to readily add new voting rules of their own. A voting rule must accept a profile of complete ballots (i.e. total orderings). It may optionally accept a profile of partial orderings as well. It must produce a mapping from each candidate to a score, such that candidates with higher scores are ahead of those with lower scores.

6.1.7 Performance Measures

After selecting the desired dataset, imputation methods, and voting rules, the user need only select one or more performance measures before starting the experiment (other parameters have functional defaults). Prefmine currently implements eleven different performance measures. Users select desired performance measures from the Performance Metric selection box, located directly below the Voting Rule selection box in the experiment window. This box is pictured in Figure 6.10. As with the other selection boxes, the Performance Metric selection box contains an array of checkboxes. Users can click on a checkbox to select a performance measure. Clicking on multiple checkboxes allows the user to select several methods. Note that although the name of the box suggests that all options are Metrics, this is not formally true. For example, the **tau** options is a correlation. The eleven possible options are summarized below. Throughout the summary, o_1 is the overall ordering of the candidates under an imputation method of interest, and o_2 is the ordering under the true preferences of voters. Additionally, S_1 and S_2 are used to denote the *scores* for the candidates, as output by the voting rules described in the previous subsection.

- **All**: Equivalent to selecting all other options simultaneously.
- **firstError**: The First Error Location measurement, defined as $\delta_{\text{FE}}(o_1, o_2) = \text{argmax}_i$ $\forall j < i, o_{1,j} = o_{2,j}$. Returns the location of the highest ranked candidate in o_1 that has been placed in an incorrect position. Proportionate to the length of the prefix

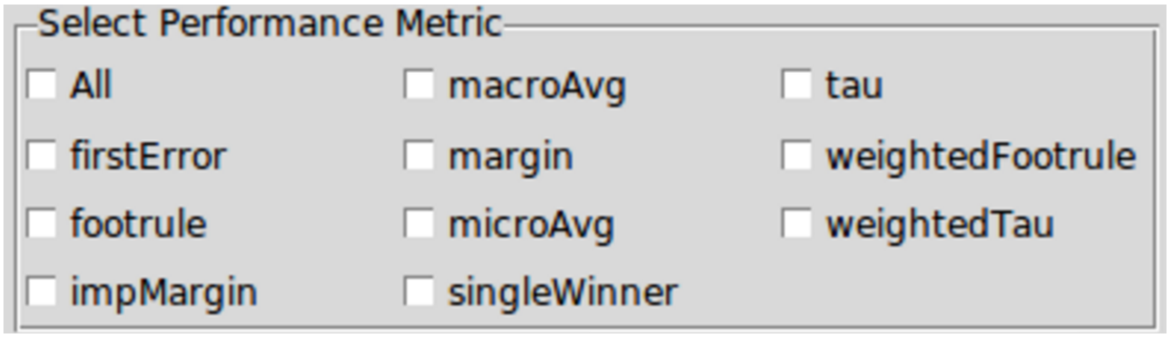


Figure 6.10: The Performance Metric selection box in Prefmine's experiment window.

under which two sequences agree. This measurement is useful in conjunction with measurements like **tau** below, which give an overall view of the similarities between two sequences.

- **footrule**: Computes Spearman's Footrule distance [Diaconis and Graham, 1977; Spearman, 1906], the underlying (but un-normalized measurement used in the non-parametric Spearman correlation. Formally, this is $\delta_{SF}(o_1, o_2) = \sum_{i=1}^{|C|} |\text{Pos}(o_1, c_i) - \text{Pos}(o_2, c_i)|$, the summed number of positions between where a candidate ought to be placed (i.e. the position of the candidate in sequence o_2), and where the candidate has actually been placed (i.e. its position in o_1). This is also similar to the Single Winner Error, but computed for every candidate, not just the winner. It is a less commonly used alternative for τ.

- **impMargin**: Computes the margin by which the winner picked by the imputation method loses to the true winner, in the ground truth elections. That is: $\delta_{IM}(o_1, S_2) = \text{argmax}_{c \in C} S_2(c) - S_2(o_1)$. Used to measure the margins of victory, which are predictors of performance for some imputation methods.

- **macroAvg**: Computes the average percentage error in the score of a candidate: $\delta_{mac}(S_1, S_2) = \frac{1}{|C|} \sum_{c \in C} |(S_1(c)/S_2(c)) - 1|$. Can be used to compare the average bias towards different candidates, independent of their popularity.

- **margin**: As **impMargin** above, but instead computes the margin by which the true winner has lost to the winner picked by the imputation method, in the imputation method's election. That is: $\delta_{Marg}(S_1, o_2) = \text{argmax}_{c\ inC} S_1(c) - S_1(o_2)$. Used to measure the margins of victory, which are predictors of performance for some imputation methods.

- **microAvg**: Computes the overall percentage error in the scores of candidates collectively:
$$\delta_{mic}(S_1, S_2) = \frac{\sum_{c \in C} |S_1(c) - S_2(c)|}{\sum_{c \in C} S_1(c)}$$
Can be used to compare the total error in the imputation, but will favour methods that are more accurate on popular candidates (which will have much higher scores) even if they are less accurate on unpopular candidates (which will have smaller scores).

- **singleWinner**: Computes the Single Winner Error for the imputation method: the rank of the winner chosen by the imputation method in the ground truth election. $\delta_w(o_1, o_2) = |\{c \in C | c \ o_2 \ o_1(1)\}$, where $o_1(1)$ is the candidate ranked first in the imputation method's ordering.

- **tau**: Computes the Kendall Correlation τ between the two outcome orderings [Kendall, 1938].
$$\tau(o_1, o_2) = \frac{\sum_{c_i \in C} \sum_{c_j \in C \setminus c_i} I((c_i \ o_1 \ c_j) \ = \ (c_i \ o_2 \ c_j) \wedge (c_j \ o_1 \ c_i) \ = \ (c_j \ o_2 \ c_i))}{|C|(|C| - 1)}$$
where I is a binary indicator variable. This measures the fraction of pairwise comparisons on which the two orders agree.

- **weightedFootrule**: A variant of **footrule** above, where errors are weighted by the (correct) position of each candidate. This effectively penalizes methods that make mistakes in highly ranked candidates, and benefits those that make mistakes only in the ordering of lower ranked candidates. Formally: $\delta_{SF}(o_1, o_2) = \sum_{i=1}^{|C|} |\mathrm{Pos}(o_1, c_i) - \mathrm{Pos}(o_2, c_i)| \times \mathrm{Pos}(o_2, c_i)$.

- **weightedTau**: A variant of **tau** above, where errors are weighted by the position of each candidate.
$$\tau(o_1, o_2) =$$
$$2\frac{\sum_{c_i, c_j \in C \times (C \setminus c_i)} I((c_i \ o_1 \ c_j) \ = \ (c_i \ o_2 \ c_j) \wedge (c_j \ o_1 \ c_i) \ = \ (c_j \ o_2 \ c_i)) \times \mathrm{Pos}(o_1, c_i)}{|C|(|C| - 1)(|C| + 1)}$$

As with the other features of Prefmine, the set of performance measures is easily extended. A performance measure must accept a mapping of candidates to scores, as output by a voting rule, and must output a scalar value. The code base readily facilitates conversion between scores and orderings. After writing a new performance measure, adding it to the user interface simply entails adding its name to a list, which will be used to populate the Performance Metric selection box at runtime.

6.1.8 Extending the System

Prefmine is envisioned as an experimental platform for other researchers to extended, facilitating the application of machine learning algorithms to the extensive Preflib repository. At present there are no plans to add further imputation methods, voting rules, or performance measures to Prefmine, although these additions would be easily accomplished. Instead, future work will focus on adding additional features to the software, to further streamline its application and to improve runtimes.

At present, users of Prefmine may add new approaches written in the D programming language directly. C and C++ code can also be linked directly to Prefmine, but requires users to write a simple interface file. Automating the construction of such an interface would broaden the appeal of Prefmine significantly, as C and C++ are much more commonly used languages.

Prefmine also lacks graphing capabilities, requiring users transport their results to external software in order to visualize the differences between the different methods. Adding in direct support for graphing through a third-party application like the `plot.ly` cloud service or Python's matplotlib could allow users to quickly compare the results of different methods, and perhaps even construct custom graphs in inside a Prefmine instance.

Finally, Prefmine currently lacks proper parallel processing facilities, and consequently is much slower than it ought to be. While in the short term this will be addressed by enabling parallel repetitions of the experiment, in the longer run general purpose frameworks like OpenCL [Stone et al., 2010] could be incorporated into Prefmine, allowing much faster computation.

Prefmine is not currently available to the public, pending further improvements to the user experience and the reintroduction of parallel processing. Interested parties should contact the author[8].

[8]j3doucet@uwaterloo.ca

6.2 Lessons for Practitioners

The previous section described the Prefmine experimental testbed, which was used to generate most of the results presented in this thesis. The testbed was carefully constructed to minimize the potential for experimental error, and to streamline the addition of new imputation methods, voting rules, and performance measures. Prior to the creation of Prefmine, experiments were performed using a multi-part experimental framework, which performed many of the same steps, but which also suffered from systematic problems with the replication and storage of results and data, parallel processing of data, and extensions of the system to incorporate additional techniques. Despite this, these earlier systems provided a number of important findings that were later incorporated into Prefmine, particularly the choice of learning algorithms available in Prefmine (logistic regression and two SVM variants), as well as the internal algorithm for feature selection (information gain with a fixed feature set size). This section describes first the deficiencies of the earlier experimental setup that were solved by Prefmine (briefly), and then the experiment design and results that were used to select the learning and feature selection algorithms used within the Prefmine system.

6.2.1 Experimental Robustness

A major advantage of Prefmine over the earlier system is the experimental robustness. The earlier system relied on a large number of modular programs written in three languages, and invoked by a series of interconnecting scripts. The result was a brittle system, where changes in one component often produced subtle errors in other, distantly related, parts. The lack of a comprehensive testing framework meant that often errors were uncovered only longer after their introduction, invalidating a large number of earlier runs. Prefmine overcomes this deficiency by combining a plugin-style architecture with integrated unit tests in the core experimental framework, incorporating the lessons learned during the initial system's development. The plugin-style architecture is also the feature that facilitates Prefmine's extensability, allowing it to support many times more experimental settings than the earlier system, and to easily integrate yet more.

The earlier system also taught important lessons regarding data integrity. Since the system worked on locally cached copies of the Preflib datasets, and since the introduction of new features frequently produced incorrect behaviour in unrelated parts of the system, data were often corrupted without the experimenter's knowledge. Eventual discovery of these errors necessitated rerunning earlier experiments, and dramatically increased development times. In contrast, Prefmine stores no data locally: fresh copies are procured from the Preflib repository at the start of each experiment. Additionally, the first step in any experiment is to render the stored copied of the data immutable (i.e. read-only). The last step is for all results to be written out into an immutable database. This prevents systematic corruption of the data, and further de-couples the different components of the system. Each algorithm is provided with its own read-write copy of a problem instance, which ensures that when a new algorithm is developed, the benchmark results from older ones are not influenced, even if the new algorithm is corrupting its own data. As a result, development times are greatly reduced, and adding new features to the system has taken less (rather than more) time as the capabilities of the framework have expanded.

The earlier system also provided lessons in the importance of integrating parallelism directly into the experimental system, rather than adding it piecemeal throughout its components. The original system frequently suffered race conditions resulting from the use of multiple components accessing the same (mutable) datasets simultaneously, and the only parallelism that was manageable in later versions of the system was through entirely separate installations that were started as separate processes. In contrast, Prefmine was implemented in a language that supported extensive static analysis, immutable data types, and simple parallel structures (D), which facilitated simple, error free, parallel processing in earlier Prefmine versions (though not in the current version).

Although these lessons seem relatively obvious in retrospect, they constitute an important reason for future experimenters to consider using (or extending) Prefmine, rather than constructing their own testbeds. At present there is no unified testbed for use with Preflib, and correctly managing the data can be a laborious process, prone to many of the issues outlined above. By extending Prefmine, practitioners instead can jump straight to the implementation of their new algorithm or voting rule, confident in the integrity and efficiency of the resulting experiment design.

6.2.2 Feature Selection and Algorithm Choice

Although the earlier system was error-prone and laborious to construct, it was used for a number of initial experiments on the imputation-based approach to social choice, some of which have not since been replicated with the more reliable Prefmine system. The experiment most likely to be of interest to readers is one that set out to answer the questions "Which conventional machine learning algorithm is best suited to predicting voters' unstated preferences?" and "How large should the feature set for the imputation-based approach to social choice need to be, and how should it be selected?". The results from this experiment later influenced the choice of classification algorithms, feature sets, and feature selection algorithms used in Prefmine[9]

To answer these questions, an experiment design much like the one used in Prefmine was adopted. 10 datasets were selected (the eight Debian sets from Preflib, and Dublin North and West from the Irish sets). These sets were picked for the same reason they were used in the experiments of Chapter 5. They represent real-world preferences that humans expressed as top-orders for use in a ranked ballot voting system. Chapter 5 provides details of the datasets in question. Problem instances were generated by discarding incomplete ballots, and then ablating the completed ones in a fashion consistent with the distribution of missingness in the original dataset, exactly as is done in Prefmine's standard ablation mode. Only the **Borda** voting rule was considered in these experiments.

The experiments of Chapter 5 measured performance in terms of the correctness of the outcomes, which was a sensible choice for comparing dissimilar approaches like MMR and the imputation-based approach. However, since this is essentially comparing competing variants of the same approach (i.e. variants of the imputation-based approach with different classifiers used), a different set of measurements were adopted. Two measurements were used. The first was the error in the **Borda** scores of the candidates after the imputation took place, a measurement that amounts to the microAvg setting in Prefmine, discussed above. Let S_1 and S_1 be vectors indexing candidates to Borda scores (i.e. positive integers). The Borda Count Error (BCE) of S_1 with respect to S_2 is given by:

$$\mathrm{BCE}(S_1, S_2) = \frac{\sum_{c \in C} |S_1(c) - S_2(c)|}{\sum_{c \in C} S_2(c)}$$

If the Borda scores (i.e. average ballot positions) are computed for an imputed preferences of voters, and for the voters' true preferences, then the BCE of the scores for the imputed preferences with respect to the scores for the true preferences is a measure of how *accurately* the imputation method has predicted the preferences of voters. Note that there exist other methods, like simply counting the error rates of the classification methods, that could be adopted instead. However, a classifier making errors that cancel out in aggregate is generally preferable to one with a lower absolute error rate making errors that do not cancel out, a concept captured nicely by looking at the errors in the aggregate totals, rather

[9]The content of this section is based on an earlier technical report drafted by the author for Dan Lizotte's graduate Applied Machine Learning Seminar. Some details of the experiment design, and the results, are reproduced verbatim, or with only minor adjustments for consistency with the notation of the thesis.

than on individual ballots. For succinctness, $\mathrm{BCE}(S_1, S_2)$ will be written simply as BCE, denoting the error between the aggregates of a method and the ground truth preferences.

Related to this preference for methods that are less *biased* in the mistakes they make, the second measurement used in this experiment is the popularity bias of the methods, a feature that was found to be problematic in some earlier, smaller scale, experiments. In particular, earlier experiments indicated that imputation-based methods tended to underestimate the aggregate scores for unpopular candidates, and overestimate them for popular ones. If a method tends to make errors that penalize unpopular candidates, then even if it performs well, its use might be questionable when the goal is to make "fair" decisions, because candidates will be treated unequally *by the system*, not merely by voters. Of course, this is a matter of degree. Certainly unpopular candidates may also view **Plurality** as a system that discriminates against them, though in a rather different manner.

In some cases bias of this kind may be a desirable feature: candidates none of the voters know much about may after all be a poor choice. However, in other applications the fact that voters do not know much about a given alternative is coincidental. For example, in the case of the robotic mining swarm, firms that do not know about a given region may do so because they are recent entrants to the area, not because the region is a fringe or undesirable alternative. In any case, methods with less bias clearly ought to be preferred to those with more, since the former are usable in a broader class of applications. Formally, bias is defined as the Pearson Correlation between the true (i.e. "correct") scores of a candidate, and the error in the candidate's score:

$$\mathrm{Bias}(S_1, S_2) = \mathrm{cor}(S_1 - S_2, S_2)$$

that is, the correlation between the elementwise difference (n.b. not the absolute value of the difference) of the two scoring vectors, and the second scoring vector. Again, Bias is frequently used without arguements to denote $\mathrm{Bias}(S_1, S_2)$, where S_1 is the Borda scores of the candidates under the imputed preferences, and S_2 is the true Borda scores of the candidates.

The experiment compared three different imputation algorithms, each under three different feature selection treatments. The three algorithms were a multinomial logistic regression model, a support vector machine, and a Naive Bayes model. The multinomial regression model works much like logistic regression, but with a multinomial output instead of a binary one. It trains a single multinomial log-linear model using a neural network from the nnet package in R [Ripley and Venables, 2011] The model has no important parameters to tune, and operates more or less automatically. Predictions from the model were generated with the predict function in R, using the undocumented type="prob" argument to provide a distribution over all the classes for each record.

The second model was a simple Naive Bayes classifier, taken from the e1071 package in R [Dimitriadou et al., 2008]. The model was configured to use Laplace smoothing with value 1, and predicted new values using the R predict function, with the type="raw" argument. To ensure the creation of a valid model, several additional preprocessing steps were performed before providing data to Naive Bayes. First, a second copy of every record was added, to avoid the situation where a class has only one example, which produced strange behaviours. Second, a very small amount of uniformly random noise ($\in \pm 10^-4$) was added to every feature for every ballot. This avoided the situation where attributes had nonzero variance overall, but zero variance when conditioned on a particular class, which produced unhandled exceptions in the model. We selected this model because of its extreme simplicity, and also because of the intuitive notion that models based on effectively counting the votes were likely to produce a good performance. This thought was eventually borne out in the Markov Tree models of Chapter 7, though in a rather more principled way, by designing a model that learns patterns in the votes by counting them, but with a clearer semantic meaning to the counts.

The final model was the support vector machine (SVM), using the libsvm implementation [Chang and Lin, 2011]. The standard svm-train and svm-predict tools were wrapped with a Perl script. The script performed a search over the SVM parameter space for each training set, and then produced a model using the parameterization with the highest cross validation accuracy. The search used the polynomial (degree 3), RBF, and sigmoid kernels. For each kernel, the search performed a grid search over the parameter space of $C \in (2^{-5}, 2^{15})$ and $\gamma \in (2^{-15}, 2^{-5})$, in steps of 2^4 . A large step size was used because of limited computational resources. Parameter selection used a randomly selected subset of 500 data points to improve run times. Additionally, the Perl script monitored the polynomial kernel, which failed to converge on certain datasets. Experimentation with the parameter ranges did not change this behaviour, so the script was configured to stop searching with the polynomial kernel after the first instance where it failed to converge on any given dataset. This provided approximately a twenty-fold decrease in run times for the SVM. Overall, this method served as the inspiration for the **svm** Imputation Method in Prefmine (not **binsvm**). There are slight differences in the Prefmine implementation however.

Since all three learning algorithms were to be used in the imputation-based approach to social choice, features over the ballot needed to be constructed. The base feature set for every run was very similar to the set described in the example near the end of Chapter 3. Features included the rank (position) of each candidate on the ballot, if known; a three valued indicator variable for each pair of candidates (i, j), indicating whether i appeared before j, after j, or the ordering of the two candidates was unknown; and a set of variables indicating the magnitude of the difference in position of two candidates on the voter's ballot. For instance, if candidate i is the voter's fifth preference, and candidate j is the voter's eighth preference, then the distance between them was $5 - 8 = -3$. Features were generated using a number of Perl and bash scripts that operated over the ballots[10]. Files containing the resulting features were stored for later use.

Two alternative feature sets were generated as well. The first set was constructed by running principal component analysis (PCA) on the original feature set [Dunteman, 1989]. PCA is a dimensionality reduction technique, in which the original feature set is compressed into a smaller set of features, each of which is a linear combination of the original feature set. Using PCA allows much (often most) of the original feature set's information to be represented in a smaller set of features, which can in turn reduce the runtime of classification algorithms applied to the data.

Before applying PCA to a dataset represented using the original feature set, values that indicated missing data were replaced with the mean of the column in which they appeared. This is a standard stem to avoid bias. If a column was entirely comprised of missing values, or if it had variance 0, it was removed entirely prior to performing PCA, to avoid faults in the PCA implementation that was used. For each training file, all components with magnitudes at least 10% of the first principal component (i.e. containing at least 10% of the information content of the most informative component) were captured, and created an alternative file containing only the reconstructions of each row within the resulting subspace. The PCA calls subtracted the mean from each column automatically, and also normalized the data prior to computing the components. R was used [Team, 2014] for the preprocessing described in this step, and the R `prcomp` implementation of PCA was then applied.

The second feature set was obtained by applying an information gain filter to the data, and selecting just the ten most predictive features under this metric, using the FSelector package in R [Romanski, 2009], and with the same preprocessing steps as the PCA feature selection. In a classification context, the information gain of a feature is the change in entropy produced by partitioning the data on that feature. It is based on the Kullback-Leibler divergence measure [Kullback and Leibler, 1951]. Let $H(Y)$ be the overall entropy a dataset with features X and labels Y[11]. Then the information gain of feature X_i is:

[10]These scripts were highly error prone, and produced different representations of the same data for different methods.

[11]i.e. a measure of how much variety there is in the labels of the dataset: higher if the dataset is split more or less evenly among all the classes, lower if one class dominates the set.

$$\text{IG}(X_i, X, Y) = H(Y) - \sum_{v \in \text{values}(X_i)} \frac{|\{x_j \in X | X_{ij} = v\}|}{|X|} H(\{y \in Y | X_{ij} = v\})$$

The 10 original features with the highest information gain were selected and used as the third feature treatment. This constant size set of features was included to explore the possibility of speeding up the machine learning algorithms with a greatly reduced version of the original feature space, rather than trying to construct new features as with the PCA approach. An advantage of the information gain approach is that, since the original features already had clear human-interpretable meanings, models learned from subsets of this set of feature ought to be easy to interpret as well. In contrast, models learned using the PCA feature set might be more difficult to interpret. As a final step before providing data to a classifier, any remaining missing values were imputed with the mean of the corresponding column. The data were also centered and normalized, and any columns with 0 variance were dropped. These steps were taken to ensure that the classifiers were able to process the data, as the support vector machine proved especially temperamental when given datasets that violated any of the mentioned properties. These techniques were automatically integrated into the support vector machine implementation that appears in Prefmine.

Each of the three algorithms (SVM, Naive Bayes, and Multinomial Logistic Regression) was run under each of the three feature treatments for each of the ten datasets considered. Every combination of dataset, feature set and algorithm was run with ten different problem instances. This small number of replications was used because the system was very slow (especially when PCA was used, and because of the parameter selection component of the SVM). Performance is summarized in Tables 6.1 and 6.2. Bold values indicate the best performance under each dataset. If there are multiple bolded values for a single set, this indicates that the two values are statistically indistinguishable. To compare measurements between two machine learning methods, a Student's t–test was used over the paired differences between the value of the measurements for the two methods on the same training and test data, with the null hypothesis that the mean difference between the measurements is zero. It was not immediately clear that the assumptions required for use of the t-test would be satisfied with such a small dataset. In particular, the normality assumption was not assured. To mitigate this, a Shapiro-Wilk test was applied to the distribution of differences, prior to analysis, and a non-parametric test was used instead if the data were not consistent with a normal distribution.

In terms of the Borda Count Error, the combination of the SVM and information gain feature selection method appears best. The SVM has statistically significantly better

	SVM			NB			MN		
	PCA	IG	plain	PCA	IG	plain	PCA	IG	plain
Deb. 2002	0.008	0.007	0.008	**0.003**	**0.003**	**0.003**	**0.003**	**0.003**	**0.003**
Deb. 2003	0.009	**0.005**	0.009	0.009	0.009	0.009	0.009	0.009	0.009
Deb. 2005	**0.014**	**0.013**	**0.016**	0.031	0.031	0.031	0.031	0.031	0.031
Deb. 2006	**0.013**	**0.013**	**0.015**	0.031	0.031	0.031	0.031	0.031	0.031
Deb. 2007	**0.033**	**0.031**	**0.033**	0.043	0.043	0.043	0.043	0.043	0.043
Deb. 2010	**0.004**	**0.004**	**0.004**	0.005	0.005	0.005	0.005	0.005	0.005
Deb. 2012	0.011	0.011	0.011	**0.003**	**0.003**	**0.003**	**0.003**	**0.003**	**0.003**
Deb. Logo	**0.006**	**0.008**	**0.006**	0.011	0.011	0.011	0.011	0.011	0.011
North	1.084	**0.923**	1.08	1.546	1.546	1.546	1.546	1.546	1.546
West	**0.549**	**0.458**	0.575	0.654	0.654	0.654	0.654	0.654	0.654

Table 6.1: A summary of the results from the preliminary experiment comparing feature selection methods and classifiers for use with the imputation based approach to resolving social choice with incomplete information. Performance is reported under the Borda Error measure, which is related to the classifier's accuracy in imputing ballots. Performance which is statistically indistinguishable from the best on any given dataset has been rendered in bold.

performance than the other two methods on every set except Debian 2002 and Debian 2012, which were both shown to have minimal room for errors in any case in Chapter 5. The SVM performance using the information gain selected features was generally at least as good as using the other two sets, and on Dublin North and Debian 2003, was statistically better. This was true despite the information gain feature set requiring the lowest runtimes overall, on account of having the smallest feature spaces.

The Bias results in Table 6.2 provide an interesting contrast. The non-SVM methods consistently exhibit lower bias than the SVM across the easy Debian sets, though they are not statistically different on the Debian Logo and Dublin North sets. The SVM results are better on Debian West. Bias also appears to be slightly worse for the SVM when used with the information gain feature sets. It appears the SVM+IG combination is imputing with a bias for more popular candidates, which leads to a more accurate imputation over all. It is also interesting to note that Dublin North in general exhibits much higher bias values than any of the other sets. Performance when ordering the less popular candidates on this set was also lower for the imputation-based approach than for other methods in the experiments of Chapter 5. It appears that the patterns present in the data lead methods

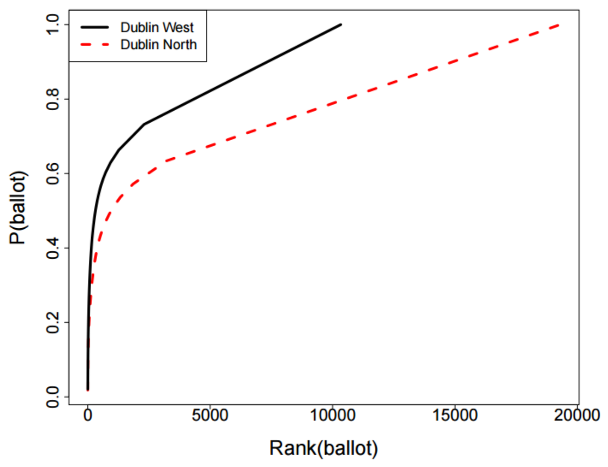


Figure 6.11: Empirical cumulative density functions for unique ballots in Dublin North and Dublin West. The x-axis shows the ranking of ballots from most to least common. The y-axis shows the cumulative proportion of voters who cast each ballot.

to impute more popular candidates. This is further illustrated by measuring the diversity of the sets of ballots in Dublin North and Dublin West, shown in Figure 6.11. The ballots of the Dublin North set exhibit much more diversity of opinion, meaning more patterns must be learned to impute them accurately.

Overall, these results suggested only a small number of features were needed to ensure good performance from machine learning models, and that using information gain to select them was a reasonable approach. As a result, all classifier-based imputation methods in Prefmine use information gain feature selection, though with a value of 30 rather than 10, since this is still adequately fast for most applications, and provides a slight performance boost on some sets. SVM was added to Prefmine, while the Naive Bayes classifier and the multinomial model were not, on the basis of these performance results.

6.3 Summary

This chapter described the Prefmine system, a general testbed for evaluating different approaches to the problem of social choice with partial information. The testbed system implements many different approaches to this problem, many different voting rules, and many different performance measures, allowing practitioners to easily evaluate algorithms

	SVM			NB			MN		
	PCA	IG	plain	PCA	IG	plain	PCA	IG	plain
Deb. 2002	0.151	0.155	0.159	0.109	0.109	0.109	0.109	0.109	0.109
Deb. 2003	0.088	**0.030**	0.051	**0.009**	**0.009**	**0.009**	**0.009**	**0.009**	**0.009**
Deb. 2005	0.255	0.182	0.309	**0.031**	**0.031**	**0.031**	**0.031**	**0.031**	**0.031**
Deb. 2006	0.171	**0.040**	0.203	**0.011**	**0.011**	**0.011**	**0.011**	**0.011**	**0.011**
Deb. 2007	0.280	**0.170**	0.296	**0.065**	**0.065**	**0.065**	**0.065**	**0.065**	**0.065**
Deb. 2010	**0.078**	0.162	**0.064**	**0.099**	**0.099**	**0.099**	**0.099**	**0.099**	**0.099**
Deb. 2012	0.34	0.35	0.32	**0.12**	**0.12**	**0.12**	**0.12**	**0.12**	**0.12**
Deb. Logo	0.056	0.034	0.026	0.007	0.007	0.007	0.007	0.007	0.007
North	0.331	0.323	0.322	0.391	0.391	0.391	0.391	0.391	0.391
West	**0.020**	0.101	**0.026**	0.041	0.041	0.041	0.041	0.041	0.041

Table 6.2: A summary of the results from the preliminary experiment comparing feature selection methods and classifiers for use with the imputation based approach to resolving social choice with incomplete information. Performance is reported under the Bias measure, which is the correlation between the classifier's error in imputing a given candidate and the popularity of that candidate. Performance which is statistically indistinguishable from the best on any given dataset has been rendered in bold.

on their own datasets. Additionally, the system's plugin-style architecture allows future experimenters to easily add a new approach to Prefmine, and then compare it with existing methods, without needing to implement their own, error-prone, code to obtain, pre-process, or evaluate data. Prefmine also provides intuitive and easy to read results summaries for the user, and efficiently makes use of parallel computing resources.

In addition to showcasing Prefmine, the chapter contains a detailed description of the system, including descriptions of all user settings, and pictures illustrating how to use Prefmine. An initial study described near the end of the chapter demonstrates the benefits of many of Prefmine's features, and provides insights into the design of Prefmine, and why future users might prefer it to creating their own systems.

Prefmine was used to generate the results described in earlier chapters of the thesis, but later chapters also use Prefmine with the same experimental designs. For simplicity, future chapters may simply describe the parameter settings used in a Prefmine experiment, rather than describing the experiment design in full detail. The interested reader can then consult the user manual portion of this chapter for details on the meanings of each specified parameter setting.

While Chapter 3 described the problem to be addressed, and Chapters 4 and 5 outlined and benchmarked an initial version of the imputation-based solution, in the next chapter (Chapter 7) a novel learning algorithm tailored specifically to the problem is presented, and evaluated within the Prefmine framework. This algorithm maintains many of the qualities that made the imputation-based approach appealing in the previous chapter, including a very accurate selection of the true winner in elections, but improves on these approaches by providing both rapid runtimes and an improvement in the ability to predict the orders that less popular candidates should appear in.

Chapter 7

Markov Tree Approach

Trees sprout up just about everywhere in computer science...

Donald Knuth [2013]

This chapter presents a new algorithm for predicting and imputing the preferences of voters, based on Markov Trees. It is intended to address the problem of social choice with incomplete preferences, as discussed in Chapter 3, and can be used as part of the imputation-based approach described in Chapter 4. However, unlike the systems described earlier in this thesis that used ordinary classification algorithms, the Markov Tree approach can also be used to select the outcome of the election directly, if desired, in a principled fashion. The Markov Tree approach is also efficient, highly parallelizable, and supported by theoretical guarantees on its performance, both in terms of its convergence rate, and via eventual consistency with two commonly used preference distributions. The Markov Tree approach is also readily interpreted, and can provide useful information about the behaviours of the voters in a given set.

This chapter begins by highlighting the need for a customized learning algorithm to use when imputing voters' preferences. It then introduces the idea of different degrees of detail when modelling voter preferences, and shows how a Markov Tree of variable depth can represent more or less detailed distributions. The remainder of the chapter develops this idea in detail, showing how to learn such models effectively, providing consistency guarantees for the convergence rate and consistency properties of the algorithm, and showing empirically that the use of Markov Trees offers a performance improvement over the approaches discussed earlier in the thesis on the real-world voting data that is considered. As discussed in Chapter 4, other learning algorithms, used in conjunction with the imputation-based approach, may be better suited to other problems, but there are reasons to suppose many real world problems align well with the assumptions made by the Markov Tree approach.

7.1 Motivation

As discussed in Chapter 4, deciding on the outcome of a social choice problem with incomplete preferences is analogous to machine learning. The extent to which a method performs well depends on the agreement between the method's underlying assumptions, and the actual behaviours and desires of the voters. The imputation-based approach at large assumes voters are similar to one another, and that trends in the missing information are easily learned from the information that is present. The results from Chapter 5 demonstrated that these assumptions are reasonably consistent with the data, especially with respect to selecting the overall winner of the election. However, the resulting models also made occasional mistakes, notably in the selection of the winner on the Meath set under the **Copeland** voting rule, and in the selection of the overall ordering on several datasets under rules like **Borda** and **K-Approval**. On these sets the imputation-based approach performed well, but not quite as well as competing approaches.

One possible reason for the shortcomings of the classification-based models is that they are not modelling an explicit distribution of preferences, but instead try to solve a number of simple classification tasks and merge the results together. To avoid this, one might prefer a model designed explicitly to learn the full distribution of preferences. In particular, such a model should be able to capture arbitrary dependencies in the ordering of the candidates on different ballots. For example, imagine that if a voter ranked $c_1 \succ c_2 \succ c_3 \succ c_4$, then the probability of ranking candidate c_5 at the end of their ballot is reduced or increased by a specific amount, and that this pattern turns out to be quite important in imputing the missing information on other ballots. Clearly an effective learning method must be able to identify the pattern in question, but these sorts of complex relationships are not well captured by the features that were used in the classifiers of the earlier chapters. Unfortunately, there is an inherent tradeoff between the level of detail such a model can learn and the amount of data needed to ensure that the learned model is reasonably correct.

The least restrictive way to model the distribution of all voters' preferences is to assume that they are drawn from an arbitrary joint distribution over the set of possible preference profiles. This model is quite appealing, because it can capture arbitrary relationships in the data. However, although this model is powerful, it is not at all useful for tasks like the imputation-based approach to social choice, because only one profile pertinent to the task at hand is observed, while there are an enormous number of possible preference profiles. This means there is not enough data to learn anything meaningful. Further, as voters are individuals, it is not unreasonable to suppose that their preferences are generated independently of one another. Certainly in something like the Debian or Irish national elections, which can contain thousands or tens of thousands of voters, any two voters selected at random are unlikely to have influenced each other directly. A reasonable simplifying assumption then is to assume that voters' ballots were generated independently and identically from a distribution of some kind.

Among distributions over (complete) preferences, the most straightforward and non-restrictive model is a general joint distribution over the set of possible total orderings. Voters' preferences are modelled as a set $\mathbf{R}$ of random variables $R_1, ..., R_{|C|}$, which all have domain C, the set of candidates or alternatives, and which take on mutually exclusive values (i.e. $R_i = c \leftrightarrow R_j \neq c,\ \forall\ 1 \leq i \neq j \leq |C|$). If, in a voter's preferences, candidate c is preceded by exactly $i-1$ candidates, then in the corresponding assignment of values to the random variables in $\mathbf{R}$, $R_i = c$. The model is defined by a probability distribution $P(R_1, ..., R_{|C|})$ over possible ballots. Given such a model, it becomes very easy to impute voter's preferences. For example, if a voter states that they like c_1 the most, and c_2 second most, one could simply consult the model, find the most common completion of ballots that started with c_1 and c_2, and impute that completion. If the model was learned from actual ballots, following the imputation-based approach described in Chapter 4, then this idea amounts to the notion of looking for other voters who also ranked c_1 and c_2 first, and imputing based on the preferences of these other voters. In fact, this is the method labelled Imputation Plurality in that chapter. Although the algorithm for accomplishing this inference is straightforward, the size of the input to the algorithm can be prohibitively large, because the probability distribution may contain enormous amounts of detail. There are $|C|!$ possible ballots, and in theory, the probabilities for each can be separately encoded. If most of the ballot is incomplete, then reasoning over the set of possible completions is similarly difficult.

In addition to the problem of reasoning over the joint distribution, learning the distribution in the first place is a difficult task. If the distribution makes no simplifying assumptions, and encodes a separate and unrelated probability for every possible ordering, then a vast number of parameters must be learned ($|C|!$), and a similarly vast number must be processed to impute votes. For example, the model must represent the probability of generating a ballot that begins $c_1 \succ c_2 \succ c_3 \succ c_4 \succ c_5$, and one that begins $c_1 \succ c_2 \succ c_3 \succ c_5 \succ c_4$, and so on for every permutation. Doing this will clearly require at least one observation for each permutation, and perhaps many more. Thus an election with 10 alternatives would need millions of ballots to learn a full joint distribution of this kind, which is clearly infeasible for most applications. If there are not enough data to learn a reasonable estimate of the parameters in a full joint distribution (i.e. $P(R_1, ..., R_{|C|})$), then imputations generated from the estimated distribution will be inaccurate even if it is computationally tractable to compute them. The combination of the difficulties in learning and imputation stem from a problem known as the "curse of dimensionality". As the number of candidates (i.e. random variables) in the model increases, the complexity of the resulting joint distribution increases exponentially, rendering it impossible to use or learn from.

Despite the intractability of learning high quality, accurate, joint distributions like the Imputation Plurality model, the key idea of the model (filling in ballots by looking at what completions were most common) is sound. If a voter states that c_1 is their first preference, and c_2 their second, then in many domains they really have provided enough information to complete the remainder of their preferences. If some further assumptions can be made on the structure of the joint distribution, then this idea can be maintained, but in a form that is tractable to learn and reason over.

An extremely strong simplifying assumption in the distribution of preferences would be to assume that the variables $R_1, ..., R_{|C|}$ are in fact independent of each other, apart from the constraint that they must have mutually exclusive values. This model is equivalent to orderings produced under Luce's axiom [Luce, 1959][1], and the model is a zeroth-order version of Plackett's model, often called Plackett-Luce [Plackett, 1975]. The probability of of assigning a given element of the $\mathbf{R}$ to candidate c_i from among the candidate set C is given by

$$P(R_k = c_i | R_{1,..,j} = \{R_j | 1 \leq j < k\}) = \frac{w_{c_i}}{\sum_{c \in C \setminus R_{1,...,j}} w_c}$$

where w_i is some weight or value associated with candidate c_i, that is independent of the position in which c_i is being assigned. Luce's model is perhaps the simplest model satisfying the independence of irrelevant alternatives criterion: The probability of picking some candidate c_i *instead of* some other candidate c_j does not depend on what other candidates are present in the set $C \setminus R_{1,...,j}$ that is to be chosen over.

In this greatly simplified model, the joint distribution is decomposed according to the following equation:

[1]Luce's Axiom is essentially a restatement of the I.I.A. criterion, that whether or not a voter prefers a to b ought not to depend on whether or not some alternative c is also available.

$$P(\mathbf{R}) = \begin{cases} \prod_{R_k \in \mathbf{R}} P(R_k | \{R_j | 1 \leq j < k\}) & \text{iff } R_k \neq R_j,\ \forall 1 \leq k \neq j \leq |C| \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

which is to say, simply the product obtained from first drawing one candidate according to Luce's model (for position R_1), and then another applying the model but over the set of remaining candidates (i.e. $C \setminus R_1$), and so on until all candidates have been picked. To impute a ballot that starts with candidates c_1 and c_2 (i.e. in which $R_1 = c_1$ and $R_2 = c_2$), one simply samples from $P(R_k = c_i | \{R_j | 1 \leq j < k\})$ repeatedly until the voter's entire ranking is completed.

This model assumes that the probability of candidates appearing in different places on the ballot is independent of the positions of the other candidates (except insofar as they have already been selected), which means it cannot model concepts like even a standard left-right political contest. For example, in many contests it might be expected that $P(R_1 = \text{Conservatives}, R_2 = \text{Socialists}) \ll P(R_1 = \text{Conservatives})P(R_2 = \text{Socialists})$, that is, a voter who ranks the Conservative candidate highly tends not to also rank the Socialist candidate highly, even though both candidates are popular overall (i.e. a high share of the electorate rank either of them highly). In exchange for the extreme simplicity of the model however, comes extremely simple learning and reasoning tasks.

This chapter adopts the idea that the order in which candidates should be imputed on a ballot should be one that maximizes the product of a set of probabilities corresponding to the probability of generating contiguous sub-sequences of the candidates. A sub-sequence of candidates is simply a group of candidates that appear next to one another in a voter's ranking (e.g. the i^{th}, $i + 1^{th}$ and $i + 2^{nd}$ most preferred choices of the voter). Under Luce's model, the probability of any sub-sequence appearing in a voter's ranking is simply the product of $|C|$ copies of re-normalized versions of the same categorical distribution. The model has only $|C|$ parameters $w_{c_1}, ..., w_{c_{|C|}}$ to learn, and inference is straightforward (the most probable sequence always being the one that places candidates with higher weight earlier than those with lower weight). Luce's model is thus easy to apply, and requires a very small amount of data to learn, provided that the set of candidates is not large. However, the model may not have much predictive power because even if the model's parameters are learned on plenty of data, it is not capable of modelling realistic nuances of real world preference profiles, like left-right political spectra.

In between the two extremes of the fully joint distribution, with millions of parameters to learn, and Luce's model, with only $|C|$, there already exist a number of alternative models, including mixtures of simple models, and nested models. Most notably, Plackett [Plackett, 1975] shows that a nested series of logistic models can be used to learn higher order sequences of patterns, an idea very similar to the one adopted in the Markov-Tree learner later in this chapter. A number of other authors consider efficient ways of learning the parameters of a Plackett-Luce model, under different assumptions about the distributions of the parameters of the model (e.g. [Azari et al., 2012; Azari Soufiani et al., 2013; Cheng et al., 2010; Caron et al., 2014; Guiver and Snelson, 2009]). Some of these models, notably those of Azari-Soufiani et al., assume that individual voters will have different subjectively-drawn impressions of the qualities of the candidates, and then attempt to infer the means of the distributions in question, as a way of deciding the outcomes of the elections. This represents a simplified version of Plackett's nested models, which were capable of capturing distributions in which voters did not rank candidates on a single-dimension.

There is also a considerable amount of work on mixtures of Mallows models. Melia and Chen [2010] describe an algorithm for learning several Mallows models [Mallows, 1957] simultaneously to model more complex preference distributions, while Lu and Boutilier [2011a] generalize the Mallows to a "Riffle Insertion Model", which, much like Plackett's nested model, can be expanded to provide greater detail in some parts of the distribution than in others.

7.2 Ballots as Sequences

One way of representing a probability distribution over a sequence of values for random variables is with a Markov Model, which stores a probability distribution for the next value in the sequence given some number of previous values, as well as a distribution over starting values for the sequence. A Markov Model assumes that the value of the variable at the present step is dependent only on the value of the variable in the previous step of the sequence, or perhaps in some (finite) number of additional sequence steps in the past. This assumption allows it to factor the joint distribution over a sequence of values $\mathbf{R}$ into a series of simpler distributions. For example, a first-order Markov Model computes the probability of a sequence as:

$$P(\mathbf{R}) = P(R_1) \prod_{i \in 2,\ldots,|R|} P(R_i|R_{i-1})$$

where $P(R_1)$ is a distribution over the value of the first variable in the sequence, and $P(R_i|R_{i-1})$ is the probability of the next variable in the sequence taking on value R_i given that it had value R_{i-1} in the previous step.

The relationships between elements of the sequence in a Markov Model can be presented pictorially using the standard notation for Bayesian Networks. Figure 7.1 shows several examples of such figures for different values of k, the number of previous steps on which future steps depend. All of the models are over sequences of length exactly 6, and the six nodes shown correspond to the value of the variables in the sequence at each of six steps. Edges show dependencies between the nodes. A probability distribution is needed at each node, showing the probability for the value of the sequence at this step, given the node's parents, $P(R_i|R_{i-j}, ..., R_{i-1})$. The grey nodes in each model all share a probability distribution, since they all have the same number of parents. The assumption that all nodes beyond the j^{th} share a probability distribution is called the Markovian assumption.

A Markov Tree is a particular way of storing, representing, and utilizing the probability distributions that are contained in a Markov Model. In a Markov Tree, the root stores the probability distribution over values of the initial variable R_1, much like the first node in a Markov Model. Both the root of a Markov Tree, and the first node of a Markov Model store the probability distribution $P(R_1)$. The root of a Markov Tree has one child for each possible value that the variable could take initially. For example, in a second-order Markov Tree, each node at the second level of the tree stores the probability distribution for R_2 given that R_1 took on a specific value in the previous step (i.e. $P(R_2|R_1 = x)$ for some x), and likewise has a child for each possible value that R_2 could take on. Collectively, *all* the nodes at the second level of the tree store the information contained in $P(R_2|R_1)$, which is the probability distribution stored at the second node in a Markov Model. Similarly, subsequent non-leaf levels of the tree will collectively store the distributions of subsequent nodes in a Markov Model. For instance, in a third-order Markov Tree, the third level of the tree collectively stores $P(R_3|R_1, R_2)$. Each node will store $P(R_3|R_1 = x, R_2 = y)$ for some specific values x and y. The leaves of the tree are special, and correspond to the entire set of grey nodes shown in Figure 7.1. In a Markov Tree of order k (i.e. a tree modelling a distribution where the i^{th} preference depends on the preceding k preferences), each leaf of the tree stores a distribution $P(R_i|R_{i-k}, ..., R_{i-1} = S)$ where S is a sequence of length k and $i > k$, meaning the leaves[2] collectively store the distribution $P(R_i|R_{i-k}, ..., R_{i-1})$. Figure 7.2 shows a Markov Tree that encodes all the information required to represent a second-order Markov Model assuming that the variables can take on only 3 distinct values at each step, denoted $\{c_1, c_2, c_3\}$. Markov Trees are used in this chapter to represent Markov Models, because a special restriction must be applied to a Markov Model in order

[2]Because of the Markov assumption, the distribution stored at each of the grey nodes is identical, and so only needs to be recorded once in a generic form, and can be reused when reasoning about all values of R_i for $i > k$.

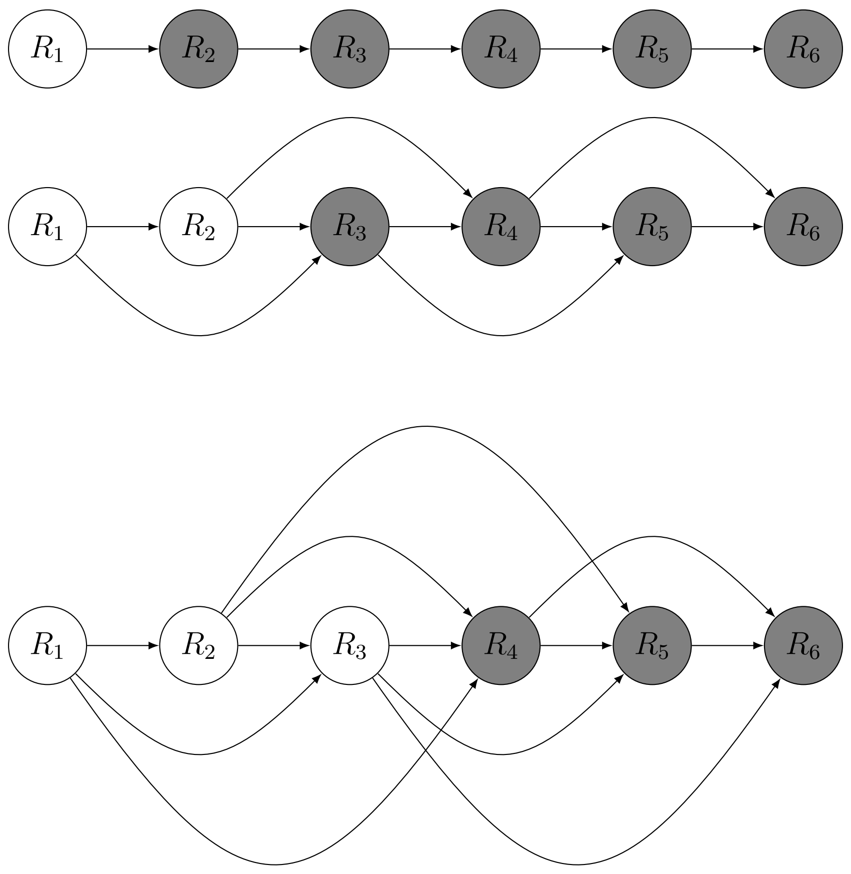


Figure 7.1: A graphical representation of first, second, and third order Markov Models for a sequence of length six. Each node corresponds to the state of a random variable at the corresponding step in the sequence. An edge from R_i to R_j shows that the probability distribution for R_j depends on the value of R_i. Gray shaded nodes have identical probability distributions (by the Markov assumption).

to represent distributions over *ballots*, and this restriction is more easily encoded into a Markov Tree. These restricted Markov Trees are discussed formally and at length in the next section.

7.2.1 An Interpretation of the Markov Trees

Before proceeding, it may be useful to discuss one possible interpretation of the vote generating process modelled by a Markov Tree. This does not concern the actual use of the Trees in the imputation-based approach, but may offer insights into what the Trees represent, and where they will and will not work effectively, a topic that comes up later in the chapter.

One possible way to view a voter's ballot is as a *trajectory* through a space in which the candidates are embedded. The notion of embedding candidates in a space is a very natural one in some domains. For example, in politics, candidates can often be sorted into a 1D (the left-right spectrum) or 2D (the economic/social policy axes) space such that nearly every voter agrees on the positions of the candidates in the space, if not on which candidates are best. The starting point for a trajectory through this space is the location of a voter's first, or most preferred, alternative. The trajectory then proceeds to the location of the second most preferred alternative, and so on, forming a series of linear segments in the space in which the candidates are embedded. Intuitively, if the earlier portions of a trajectory are known, later portions might reasonably be inferred via extrapolation. Insofar as some trajectories are more common or popular than others, the relative quality of different imputations of the candidates can then be inferred.

From a generative perspective, this could model the idea that ranked data are produced based on the candidates a voter has thought about in the recent past. This would be applicable to domains where human voters are confronted with excessive amounts of choice, like ordering the set of all possible films, or all possible meals. In practice, such a voter probably does not construct a full ranking of the alternatives in their head. Instead, the voter will first rank a few options that come to mind immediately. Thinking of these options will suggest certain successors, which in turn will suggest other successors. More formally, assume voters' preferences begin with a *seed ranking* of size k, a top order drawn from a distribution over the set of top-k rankings of C. Following that, a voter's next most preferred alternative is drawn from a distribution that depends only on the immediately preceding k items. The process is then repeated, with some probability of stopping at each step, to yield a top-order over some (potentially large) set of candidates.

A Markov Tree of order k is nearly (though not quite) sufficient to represent the process

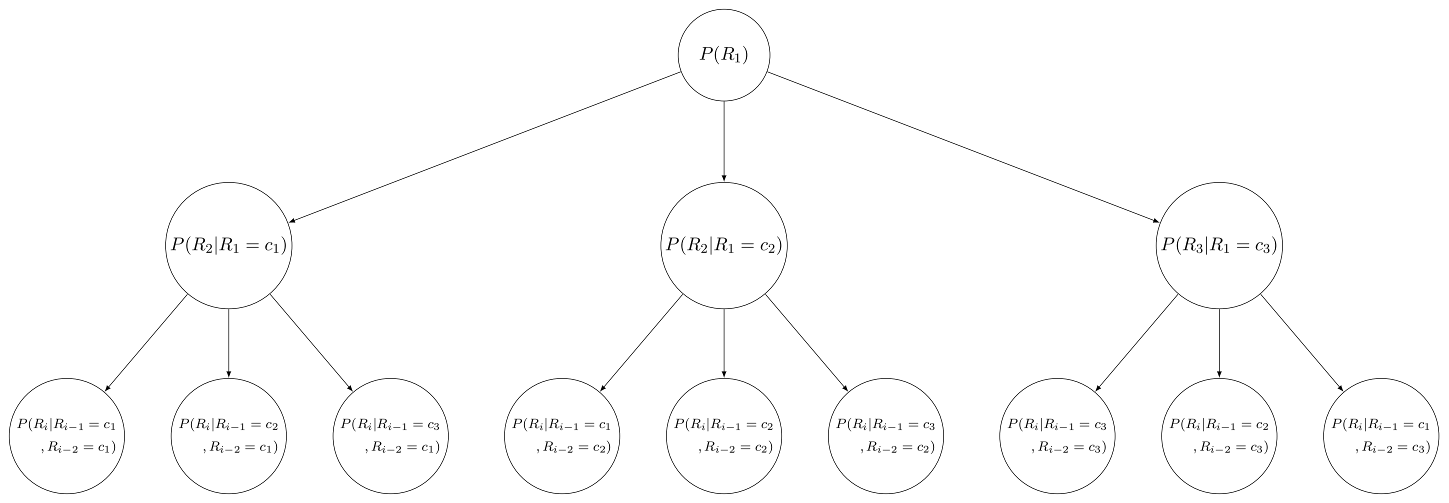

$P(R_1)$
$P(R_2|R_1 = c_1)$
$P(R_2|R_1 = c_2)$
$P(R_3|R_1 = c_3)$
$P(R_i|R_{i-1} = c_1, R_{i-2} = c_1)$
$P(R_i|R_{i-1} = c_2, R_{i-2} = c_1)$
$P(R_i|R_{i-1} = c_3, R_{i-2} = c_1)$
$P(R_i|R_{i-1} = c_1, R_{i-2} = c_2)$
$P(R_i|R_{i-1} = c_2, R_{i-2} = c_2)$
$P(R_i|R_{i-1} = c_3, R_{i-2} = c_2)$
$P(R_i|R_{i-1} = c_3, R_{i-2} = c_3)$
$P(R_i|R_{i-1} = c_2, R_{i-2} = c_3)$
$P(R_i|R_{i-1} = c_1, R_{i-2} = c_3)$

Figure 7.2: A graphical representation of a second order Markov Tree. The internal node of the tree (the root) stores information about the distribution of values for R_1 and R_2, while the leaves store information about the distribution of R_i given the previous $k = 2$ values of the variable, for every $i > k$.

for generating trajectories of candidates in the fashion mentioned above[3]. Since a Markov tree is fairly compact, provided that k is small, and has a readily interpretable representation, it may be a useful and informative alternative to more complex existing models for learning user preferences, at least in domains with fairly homogenous preference trajectories (i.e. where the candidates are embedded in a low-dimensional space). The problem with using a Markov Tree to generate a trajectory of candidates is that a voter's trajectory is in fact a permutation over the set of candidates, implying that the values taken on by a variable R_i must be mutually exclusive with those of of all other variables R_j in the sequence. Another way of looking at this is that a valid trajectory passes through each candidate exactly once. This is a clear violation of the Markovian assumption, because the probability of selecting a particular candidate will depend on *all* of the previous nodes in the sequence. To address this, a simplifying assumption is made via Luce's axiom, detailed in the next section.

7.3 Learning Markov Models

This section will show a restricted version of a Markov Tree, capable of imputing voters' preferences, and how it can be learned from incomplete preference profiles. Once a Tree has been learned on the basis of a set of ballots, it can be used to impute any missing preferences in that set, per the imputation-based approach described in Algorithm 1 of Chapter 4. The section begins with a formal, mathematical description of the family of models that could be learned, and the algorithm that is used to estimate the parameters of such models. The learning, imputation, and generation processes are then described algorithmically using pseudocode in Algorithms 12-14. Algorithm 12 corresponds to the parameter M of Algorithm 1 from Chapter 4, while Algorithm 13 corresponds to $m(B)$ in that algorithm.

Formally, the proposed model is to use a **restricted** Markov Tree, in which a voter's j^{th} most preferred alternative is modelled as a discrete random variable R_j with domain equal to the set of alternatives that are to be ranked, C. R_j is distributed according to a categorical distribution with parameters dependent on k preceding variables R_i for $j - k \leq i < j$, and conditionally independent of the votes of any other voter given the parameters of the model itself (i.e. votes are drawn i.i.d.). As mentioned above, a Markov tree is not quite capable of modelling the generative process that was described. This is because each candidate can be ranked only once, whereas in a true Markov tree of order k,

[3]Note: a tree of order k has depth $k + 1$.

the $k+2^{nd}$ candidate cannot depend on the first. To address this, the following **constraints** are imposed: $P(R_j = c|R_y = c \wedge y \neq j) = 0$, and also $P(R_j = c|\{R_y \mid y < j \wedge R_y \neq c\})$ $= P(R_j = c|\{R_y \mid j - k \leq y < j\})$ where k is said to be the *order* of the restricted tree. The resulting model is a Markov Tree of order k, but with mutual exclusivity constraints encoded. Such a model can be compactly represented as an ordinary Markov tree of order k. When reasoning about R_j, one only needs to restrict the domain of R_j to those elements of C that did not appear previously in the ranking, renormalizing the probabilities of the remaining elements of C in accordance with Luce's axiom.

The parameters of the model are encoded in several sets. The first set corresponds to the leaves of the tree, and are used to impute the $k+1^{st}$ candidate given a list of k candidates that are ranked already. For a given sub-sequence S of k candidates drawn from C, $\theta_{S,c,k+1}$ denotes the value of the parameter corresponding to $P(R_j = c|\{R_y = S_{y-(j-k)+1} \mid j - k \leq y < j\})$, for any $j > k$. $P(\theta_{S,c,k+1})$ is then the prior probability distribution over values of $\theta_{S,c,k+1}$ in the model, and the complete set of parameters corresponding to multinomial distribution over values of R_j given the observation of each possible sub-sequence S is denoted with $\Theta_{C,k+1} = \{\theta_{S,c,k+1} \mid S \in \rho(C,k) \wedge c \in C\}$, where $\rho(C,k)$ is the set of all permutations of k elements of C. Additionally, the model will have $k-1$ further sets of parameters for the initial levels of nodes in the Markov tree, corresponding collectively to the joint distribution over the first k candidates ranked, which naturally cannot be predicted by a distribution that requires k prior rankings. These parameters are denoted with $\Theta_{C,y} = \{\theta_{S,c,y} \mid S \in \rho(C,y) \wedge c \in C\}$, and are defined for every $1 \leq y \leq k$. Each of the parameters $\theta_{S,c,y}$ corresponds to $P(R_j = c|\{R_x = S_{x-(j-y)+1} \mid j - y \leq x < j\})$, predicting the y^{th} candidate on the basis of the preceding $y-1$. Thus, for example, a first order tree has two sets of parameters, $\Theta_{C,2}$, stored in the leaves, and $\Theta_{C,1}$, stored in the root node. In a slight abuse of notation, the set of all parameters for a complete k^{th} order model is denoted with $\Theta_{C,\vec{k}}$. Throughout, there are occasional abuses of notation by treating S as though it were a set. S is always a sub-sequence rather than a set, but $S \setminus s$ may be used to indicate the sequence with its *last* element (s) truncated. Similarly, $|S|$ may denote the length of S.

Consider for example a first order model (i.e. $k = 1$) describing the distribution over a set of four candidates $C = \{c_1, c_2, c_3, c_4\}$. The model will have four parameters in $\Theta_{C,1}$, and 16 in $\Theta_{C,2}$ (4 in each of the 4 leaf distributions). Example values for the parameters are summarized in Table 7.1, which provides some intuition about the notation used in the model. The left hand table shows the distribution over initial elements of the sequence, in this case, over voters' most preferred candidates. The right hand table shows the distribution of a voter's second preference, conditioned on their first. A voter's third preference will obey an identical distribution, conditioned on their second, but re-normalized to remove

c	$\theta_{S,c,1}$	Value
c_1	$\theta_{\emptyset,c_1,1}$	0.1
c_2	$\theta_{\emptyset,c_2,1}$	0.4
c_3	$\theta_{\emptyset,c_3,1}$	0.4
c_4	$\theta_{\emptyset,c_4,1}$	0.1

S	c	$\theta_{S,c,2}$	Value
c_1	c_1	$\theta_{c_1,c_1,2}$	0
	c_2	$\theta_{c_1,c_2,2}$	0.4
	c_3	$\theta_{c_1,c_3,2}$	0.3
	c_4	$\theta_{c_1,c_4,2}$	0.3
c_2	c_1	$\theta_{c_2,c_1,2}$	0.8
	c_2	$\theta_{c_2,c_2,2}$	0
	c_3	$\theta_{c_2,c_3,2}$	0.2
	c_4	$\theta_{c_2,c_4,2}$	0
c_3	c_1	$\theta_{c_3,c_1,2}$	0.1
	c_2	$\theta_{c_3,c_2,2}$	0.1
	c_3	$\theta_{c_3,c_3,2}$	0
	c_4	$\theta_{c_3,c_4,2}$	0.8
c_4	c_1	$\theta_{c_4,c_1,2}$	0.2
	c_2	$\theta_{c_4,c_2,2}$	0.5
	c_3	$\theta_{c_4,c_3,2}$	0.3
	c_4	$\theta_{c_4,c_4,2}$	0

Table 7.1: Tables showing an example of the probability distributions used in a first-order Markov Tree. The distribution over initial states R_1 is shown on the left, while the transition probabilities (i.e. the distribution of R_i given R_{i-1}) are shown in the table on the right. The red numbers correspond to the probabilities that are re-normalized and used to determine a voter's third preference, given that their first preference was c_2, and their second was c_3.

their first preference from consideration. For example, suppose that a voter's first preference is c_2, and second preference is c_3. Looking in the right hand table, the probability of selecting c_1 as the voter's third preference is 0.1, given by $\theta_{c_3,c_1,2}$. Likewise, the probability of selecting c_4 as the voter's third preference is 0.8, as indicated by $\theta_{c_3,c_4,2}$. Since c_2 has already been assigned as the voter's first preference, these probabilities do not sum to 1. By applying Luce's Axiom, the probabilities are renormalized, reflecting the idea that, apart from the mutual exclusivity constraint, the distribution depends only on the immediately preceding preference. Therefore, c_4 will be selected as the voter's next preference with probability $\frac{0.8}{0.1+0.8} \approx 0.89$ and c_1 will be selected with probability $\frac{0.1}{0.1+0.8} \approx 0.11$.

Having described the model, it now remains to describe how its parameters should be learnt. Learning the parameterization of $\Theta_{C,k}$ is the most difficult. This requires a set of *contiguous* partial rankings over the subsets of k candidates in C. A contiguous ranking over a set C is defined by a successor function $\succ_{\text{contig}}$ such that $\forall\ a, b \in C,\ a \succ_{\text{contig}} b \iff \text{rank}(b) \leq \text{rank}(a) \wedge \ \not\exists\ c \in C\ s.t.\ \text{rank}(b) \leq \text{rank}(c) \leq \text{rank}(a)$, where $\text{rank}(x)$ is the number of candidates that are preceded by x in the ranking. That is, it is a partial ordering where certain pairs (those where $a \succ_{\text{contig}} b$ is defined) are *known* to be contiguous in the ranking, independent of the fact that certain other candidates may be unranked. Although the same symbol ($\succ$) is used, this should not be confused with the more general ideas of a partial or total order. This restriction at first appears substantial, but note that any total order over the candidate set can be represented with a contiguous ranking, and so can any top-k orders, and partial orders in which relatively few pairs of candidates are incomparable (e.g. $a \succ b \succ c \succ (d \sim e)$ is a contiguous ranking where a, b, and c appear contiguously in the first three positions, ahead of d and e, but the relative order of d and e is undetermined).

Given a single total ordering of the candidates $\succ$, learning proceeds by splitting the ordered list corresponding to $\succ$ into a series of sub-sequences, used as observations to adjust the parameters of $\Theta_{C,\vec{k}}$. The candidate ranked first S_1 is an observation for $R_1 = S_1$, which consequently influences the posterior distribution of $\theta_{\emptyset,R_1,1}$. Similarly, the first j ranked candidates serve as an observation of $R_j = S_j | \{R_i = S_i, i < j\}$ for each $j < k$, which influences the posterior distribution of $\theta_{S_{1..j-1},S_j,j}$. After the first k candidates ranked in $\succ$, each subsequent ranked candidate defines an observation of $R_j = S_j | \{R_i = S_i, j-k \leq i < j\}$. Given a contiguous ranking that is not total, a similar process is followed. All contiguous subsequences of length $k+1$ can be extracted and used as observations of $R_j = S_j | \{R_i = S_i, j-k \leq i < j\}$, and any top-order of length $j < k$ can also be used to provide observations for the other model parameters. Collectively, these sequences constitute the training data. To estimate the parameters from the training data, standard statistical estimation techniques are adopted. In particular, given a set of ballots that all begin with the same sub-sequence S and rank at least one more candidate c_i following S, the parameters in $\{\theta_{S,c_i,|S|} \mid c_i \in C \setminus S\}$ can be learned via the Maximum Aposteriori estimate for a categorical distribution:

$$\hat{\theta}_{S,c_i,|S|} = \frac{\alpha_i + N_{S,c_i,|S|}}{\sum_{c_j \in C \setminus S} \alpha_j + N_{S,c_j,|S|}}$$

where $N_{S,c_i,|S|}$ is the number of observations which contain sequence S followed by c_i, and α_i is the number of *prior* observations of c_i following S (i.e. α collectively parameterize a Dirichlet prior for the categorical distribution defined by $\theta_{S,c_i,|S|}$ for every $c_i \in C \setminus S$).

For example, suppose that a first order Markov Tree over four candidates $C = \{c_1, c_2, c_3, c_4\}$

is to be trained. Initially the model effectively consists of 5 categorical distributions, respectively modelling $P(R_1)$ for the root, and for the leaves, $P(R_i|R_{i-1} = c_1)$, $P(R_i|R_{i-1} = c_2)$, $P(R_i|R_{i-1} = c_3)$ and $P(R_i|R_{i-1} = c_4)$. Suppose we begin with the assumption of a single observation for each candidate in each distribution. The resulting Dirichlet distributions are summarized in Table 7.2. Suppose that the training data consist of the sequences in Figure 7.3. These are converted into observations as follows. For the $P(R_1)$ distribution, there are 9 observations of sequences that start with c_1, 4 that start with c_2, 12 that start with c_3, and 8 that start with c_4. For distribution $P(R_i|R_{i-1} = c_1)$, there are 17 observations of c_2 (contributed from the two middle sets of ballots), and no other observations at all. For the distribution $P(R_i|R_{i-1} = c_2)$ there are 21 observations of c_4, and 4 observations of c_3. For the distribution $P(R_i|R_{i-1} = c_3)$, there are 12 observations of c_2, 8 of c_1, and 4 of c_4. Finally, for $P(R_i|R_{i-1} = c_4)$, there are 16 observations of c_1 and 17 of c_3. Adding these totals to the initial counts yields the observation totals for each parameter. Normalizing by the sum of the observations in each of the categorical distributions then produces an estimate of the parameters, as shown in Table 7.3.

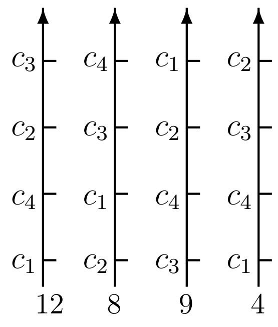


Figure 7.3: A preference profile to be used in the example training of a First Order Markov Tree.

It is easy to see that a tree of this class, properly parameterized, can generate any sequence of alternatives. The full distribution over the first k candidates is represented fully, and with proper parameterizations, it can generate any initial sequence. The remainder of the sequence will be sampled from the $k+1^{th}$ node, which will initially assign probabilities to each previously unranked candidate. After selecting a candidate, the distribution conditional on the second through $k+1^{th}$ candidate will be renormalized by restricting all $k+1$ candidates thus far selected. This will proceed until the final candidate is selected, provided that all candidates have non-zero probabilities of appearing in all conditionings of the $k+1^{th}$ node. The distribution as represented is not proper, because the representation is compact. If the tree were expanded out fully, so that each node beyond the k^{th}

c	$Parameter$	Obs
c_1	$\theta_{\emptyset,c_1,1}$	1
c_2	$\theta_{\emptyset,c_2,1}$	1
c_3	$\theta_{\emptyset,c_3,1}$	1
c_4	$\theta_{\emptyset,c_4,1}$	1

S	c	Parameter	Obs
c_1	c_2	$\theta_{c_1,c_2,2}$	1
	c_3	$\theta_{c_1,c_3,2}$	1
	c_4	$\theta_{c_1,c_4,2}$	1
c_2	c_1	$\theta_{c_2,c_1,2}$	1
	c_3	$\theta_{c_2,c_3,2}$	1
	c_4	$\theta_{c_2,c_4,2}$	1
c_3	c_1	$\theta_{c_3,c_1,2}$	1
	c_2	$\theta_{c_3,c_2,2}$	1
	c_4	$\theta_{c_3,c_4,2}$	1
c_4	c_1	$\theta_{c_4,c_1,2}$	1
	c_2	$\theta_{c_4,c_2,2}$	1
	c_3	$\theta_{c_4,c_3,2}$	1

Table 7.2: Tables showing the prior observations for an example First-Order Markov Tree before being updated with training data. The model can be viewed as five categorical distributions, one in the table on the left, and four in the table on the right. Rather than showing the proportions in the distribution, the number of observations for each of the candidates is shown.

c	*Parameter*	Obs	$\hat{\theta}$
c_1	$\theta_{\emptyset,c_1,1}$	1+9	$\frac{10}{37}$
c_2	$\theta_{\emptyset,c_2,1}$	1+4	$\frac{5}{37}$
c_3	$\theta_{\emptyset,c_3,1}$	1+12	$\frac{13}{37}$
c_4	$\theta_{\emptyset,c_4,1}$	1+8	$\frac{9}{37}$

S	c	Parameter	Obs	$\hat{\theta}$
c_1	c_2	$\theta_{c_1,c_2,2}$	1+17	$\frac{18}{20}$
	c_3	$\theta_{c_1,c_3,2}$	1	$\frac{1}{20}$
	c_4	$\theta_{c_1,c_4,2}$	1	$\frac{1}{20}$
c_2	c_1	$\theta_{c_2,c_1,2}$	1	$\frac{1}{28}$
	c_3	$\theta_{c_2,c_3,2}$	1+4	$\frac{5}{28}$
	c_4	$\theta_{c_2,c_4,2}$	1+21	$\frac{22}{28}$
c_3	c_1	$\theta_{c_3,c_1,2}$	1+8	$\frac{9}{27}$
	c_2	$\theta_{c_3,c_2,2}$	1+12	$\frac{13}{27}$
	c_4	$\theta_{c_3,c_4,2}$	1+4	$\frac{5}{27}$
c_4	c_1	$\theta_{c_4,c_1,2}$	1+16	$\frac{17}{36}$
	c_2	$\theta_{c_4,c_2,2}$	1	$\frac{1}{36}$
	c_3	$\theta_{c_4,c_3,2}$	1+17	$\frac{18}{36}$

Table 7.3: Tables showing the posterior observation counts and parameter estimates for an example First-Order Markov Tree after being updated with training data. The model can be viewed as five categorical distributions, one in the table on the left, and four in the table on the right.

corresponds to a particular restriction of the k^{th} node, then a proper distribution could be obtained. As it is, if the compact distribution is sampled from with appropriate restrictions, then the product of the *renormalized* parameters used at each sampling step will yield a proper probability, even if the raw parameter values would not.

Once a model has been learned, it is a straightforward process to impute a ballot. A top-order S can be extended by sampling a candidate from $C \setminus S$ from the appropriate learned categorical distribution. For instance, using the model from the previous example, given a top-ordered ballot $b_i = \emptyset$ that ranks no candidates at all, then after learning has been completed, the model will extend the ballot by imputing the voter's first preference as c_1 with probability $\frac{10}{37}$, c_2 with probability $\frac{5}{37}$, c_3 with probability $\frac{13}{37}$, and c_4 with probability $\frac{9}{37}$. If b_i already contains at least one candidate, and the last candidate ranked on b_i is c_i, then the sequence is extended by sampling from the categorical distribution with parameters $\theta_{c_i,c_j,2}$, for all values of j such that c_j is not ranked on b_i. For example, if b_i ranks c_1 first, and c_2 second, then the model imputes c_3 with probability $\frac{5}{27}$, and c_4 with probability $\frac{22}{27}$, the renormalized values for $\theta_{c_2,c_3,2}$ and $\theta c_2, c_4, 2$ from Table 7.3[4]. The imputation of a partial order can be accomplished using this method as well.

7.3.1 Convergence Rates

Suppose a set of rankings are generated from a restricted Markov Tree of order k with unknown parameter values. It is reasonable to suppose that another restricted Markov Tree of order k could be constructed, and that by observing the rankings generated from the first tree, could be trained to have parameter estimates that would converge to those of the original tree as more data was observed. A bound on the amount of data required to learn the parameters of the above model to within a given tolerance ϵ would be useful however. For simplicity, the learning process analyzed to develop a bound is maximum likelihood estimation rather than maximum aposterori estimation.

Theorem 4. *Given a set of candidates C and a restricted Markov Tree T of order k describing a distribution over candidate sequences of C, the error in a learned estimate $\hat{\theta}_{S,c,|S|}$ of a parameter $\theta_{S,c,|S|}$ in a second restricted Markov Tree T' for a given sequence $S \in C^k$ and $c \in C$ is in $O(\theta_{S,c,|S|}\sqrt{\epsilon})$ with probability at least $(1-\alpha)^k$ after observing $N = \frac{-\ln((\alpha/|C|)^2)}{\theta_{S,c,|S|}} \prod_{0<j<k}(\theta_{S_{1,j-1},S_j,j}(1-4\sqrt{\epsilon})^{-1})$ sequences drawn from T.*

[4]Alternatively, instead of imputing values according to their probabilities, the candidate with the highest number of observations can be picked directly to yield the (locally) most probable completion. Alternatively, the globally most likely completion completion can be found via the application of a standard variable elimination algorithm, though not necessarily efficiently (because the tree is restricted, the Markov property assumed by more efficient algorithms like the Vitirbi algorithm, does not hold).

The gist of the proof is that for parameters in the top level of the tree (e.g. $\theta_{\emptyset,c,1}$), Quesenberry and Hurst's bound for the distribution of a multinomial proportion[5] [Quesenberry and Hurst, 1964] can be used to guarantee that $\hat{\theta}_{S,c,|S|}$ lies within an interval centred at $\theta_{S,c,|S|}$ with probability $(1-\alpha)$. The interval can be expressed using Alzer's inequality [Alzer, 1997] for the χ^2 cumulative density function in a closed form depending only on the values of ϵ, $\theta_{S,c,|S|}$ and N. Some algebra then yields the desired bound. For parameters lying deeper in the tree, the same approach is used, but only the subset of the rankings drawn that begin with sequence S can be used to learn the value of deeper parameter $\theta_{S,c,|S|}$. This results in a blowup of $\prod_{0<j<k}(\theta_{S_{1,j-1},S_j,j}(1-4\sqrt{\epsilon})^{-1})$ in the amount of data needed to produce the same bound as in the data for the parameters at the top level, and this is shown inductively.

Proof. Suppose that N data points (i.e. total orderings) are sampled from a restricted Markov Tree T of order k as described above, parameterized with $\Theta_{C,\vec{k}}$. Let $N_{c,1}$ be a random variable representing the number of rankings sampled from T that rank candidate c highest. Let $\theta_{\emptyset,c,1}$ be the parameter in T corresponding to the true probability of generating a sequence that starts with c. Now, $N_{c,1}/N = \hat{\theta_{\emptyset,c,1}}$ is effectively distributed as a multinomial proportion. By Quesenberry and Hurst's (conservative) bound on the distribution of of a multinomial proportion [Quesenberry and Hurst, 1964], with probability $1-\alpha$,

$$\frac{N_{c,1}}{N} \in \frac{2\theta_{\emptyset,c,1}N + \chi^2_{1,\alpha/|C|} \pm \sqrt{\gamma}}{2(N+\chi^2_{1,\alpha/|C|})}$$

where

$$\gamma = \chi^2_{1,\alpha/|C|}(\chi^2_{1,\alpha/|C|} + 4N\theta_{\emptyset,c,1}(1-\theta_{\emptyset,c,1}))$$

and where $\chi^2_{1,\alpha/|C|}$ is the value at which the probability mass in the right tail of a χ^2 distribution with 1 degree of freedom is exactly $\alpha/|C|$. It follows that:

$$\frac{N_{c,1}}{N} \in \frac{\theta_{\emptyset,c,1} + \frac{\chi^2_{1,\alpha/|C|}}{2N} \pm \sqrt{\gamma/(4N^2)}}{1+\frac{\chi^2_{1,\alpha/|C|}}{2N}}$$

[5]i.e. the expected proportion of samples taking a specific value drawn from a multinomial

Alzer's inequality [Alzer, 1997], states that $\chi^2_{1,\alpha/|C|} < -2\ln((\alpha/|C|)^2)$. Let $\beta = \frac{-\ln((\alpha/|C|)^2)}{N}$. Then, with probability $(1-\alpha)$:

$$\frac{N_{c,1}}{N} \in \frac{\theta_{\emptyset,c,1} + \beta \pm \sqrt{\gamma/(4N^2)}}{1+\beta}$$

from which it is easy to show that:

$$\frac{N_{c,1}}{N} - \theta_{\emptyset,c,1} \in \beta(1 - \frac{N_{c,1}}{N}) \pm \sqrt{\gamma/(4N^2)}$$

$$|\frac{N_{c,1}}{N} - \theta_{\emptyset,c,1}| < \beta(1 - \frac{N_{c,1}}{N}) + \sqrt{\gamma/(4N^2)}$$

$$|\frac{N_{c,1}}{N} - \theta_{\emptyset,c,1}| < \beta + \sqrt{\gamma/(4N^2)}$$

Now, assume that $\beta < \epsilon\theta_{\emptyset,c,1}$, which will be true for some $0 < \epsilon < 1$ for large enough N. Then:

$$|\frac{N_{c,1}}{N} - \theta_{\emptyset,c,1}| < \epsilon\theta_{\emptyset,c,1} + \sqrt{\epsilon^2\theta^2_{\emptyset,c,1} + 2\epsilon\theta^2_{\emptyset,c,1}(1-\theta_{\emptyset,c,1})}$$

$$|\frac{N_{c,1}}{N} - \theta_{\emptyset,c,1}| < \epsilon\theta_{\emptyset,c,1} + \theta_{\emptyset,c,1}\epsilon^{\frac{1}{2}}\sqrt{\epsilon + 2(1-\theta_{\emptyset,c,1})}$$

$$|\frac{N_{c,1}}{N} - \theta_{\emptyset,c,1}| < 4\theta_{\emptyset,c,1}\sqrt{\epsilon}$$

which shows the desired bound for the case of the $P(R_i, 1)$.

The general case is now proven inductively. Suppose that the desired bound holds for a parameter $\theta_{S\setminus s,s,k-1}$. The goal is to show that it also holds for a parameter $\theta_{S,c,k}$, where $c \in C \setminus S$ is some other candidate. Let N_S be the total number of rankings drawn that begin with sequence S. By the inductive hypothesis,

$$|\theta_{S\setminus s,s,k-1} - \frac{N_S}{N_{S\setminus s}}| < 4\theta_{S\setminus s,s,k-1}\sqrt{\epsilon}$$

with probability at least $(1-\alpha)^{k-1}$, for some $\frac{-\ln((\alpha/|C|)^2)}{N_{S\backslash s}\theta_{S\backslash s,s,k-1}} < \epsilon$. It follows that:

$$\theta_{S\backslash s,s,k-1} - \frac{N_S}{N_{S\backslash s}} \in \pm 4\theta_{S\backslash s,s,k-1}\sqrt{\epsilon}$$

$$\frac{N_S}{N_{S\backslash s}} \in \pm\theta_{S\backslash s,s,k-1}(4\sqrt{\epsilon}-1)$$

$$N_S < N_{S\backslash s}\theta_{S\backslash s,s,k-1}(4\sqrt{\epsilon}-1)$$

$$N_S > N_{S\backslash s}\theta_{S\backslash s,s,k-1}(1-4\sqrt{\epsilon})$$

with probability $(1-\alpha)^{k-1}$ also. Following the argument for the basecase above, if it is allowed that $\beta < \epsilon\theta_{S,c,k}$, then $|\theta_{S,c,k} - \frac{N_{S\cup c}}{N_S}| < 4\theta_{S,c,k}\sqrt{\epsilon}$ with probability at least $(1-\alpha)^k$. Since $N_S > N_{S\backslash s}\theta_{S\backslash s,s,k-1}(1-4\sqrt{\epsilon})$ it follows that N (the total number of sequences sampled) must also be larger by a factor of at least $\frac{1}{\theta_{S\backslash s,s,k-1}(1-4\sqrt{\epsilon})}$ than was required for the bound to hold for $\theta_{S\backslash s,s,k-1}$.

$\square$

In more concrete terms, this result indicates that the total number of samples that must be drawn to accurately infer the true parameters of T grows exponentially in the depth of k, but linearly in the total number of parameters (i.e. in $|\Theta_{C,\vec{k}}|$). Further, errors will be concentrated in the least common sequences, which are unimportant for many applications.

It may be helpful to see an example that illustrates the orders of magnitude for the data requirements. Suppose a second order Markov Tree is to be trained on ballots for an election with 10 candidates. It is expected *a priori* that all candidates have at least a 5% chance of following any two other candidates in a given ranking, and that ballots were generated by a second order Markov Tree or a similar process. If the error in the estimate of any parameter's value is desired to be at most 1% with probability 95%, then data requirements would be:

$$N = \frac{-\ln(0.005^2)}{(0.05)^2(1-4\sqrt{0.01})^{-1}}$$

or about $7,000$ example sequences of length $k+1$. This corresponds to about 875 complete rankings drawn from T, because a complete ranking of 10 candidates contains 8 subsequences of length 3. However, if some of the sequences are much more likely than others, these more likely sequences could be learned with far less data (since those sequences will appear disproportionately often in the data).

The runtime for parameter estimation is linear in the amount of data given (i.e. the number of contiguous subsequences provided), but there may be an enormous number of parameters in a deep model, such that simply enumerating the parameters requires more time than a pass through the data. To avoid this, the implementation renders training time independent of the total size of the tree being represented via the use of lazy evaluation and hash maps. The Markov Tree is represented as a nested data structure. The root holds a hashmap from C to the values of $\theta_{\emptyset,c_i,1}$ for all $c_i \in C$. Additionally, it holds a map from C to instances of the data structure that store distributions for the second candidate in sequence. Each of those nested data structures holds a mapping to distributions over the third candidate selected, and so on up to the k^{th}. However, vitally, the nested structures are only constructed if actual data is observed that necessitates their construction. Otherwise, they can be left un-constructed, and during inference and generation, an interchangable uniform distribution can be substituted anywhere that an undefined value is used. Since constructing a new node is a constant-time operation, even in the worst case, the total time required to construct and learn a model is linear in the number of observations (i.e. $O(kn)$ if n subsequences of length k are available for training). This process is summarized in Algorithm 12. The generation and imputation processes for top-orders using data structures of this form are described in Algorithms 14 and 13, with runtimes that amount to $O(k|B||C|)$, or a constant cost of k per preference imputed. The result is that learning the model can be an exceptionally fast process. Contrast this with approaches that need to solve complex optimization problems when used with some voting rules [Lu and Boutilier, 2011a], or need to make multiple passes through the data like logistic regression (i.e. during each iterative update of the model's weights). Note also that the structure can be readily parallelized by placing a locking semaphore on each TreeNode structure. If the model is deep then the chance of collisions should be very low.

7.4 Consistency Results

The results of the previous section demonstrate consistency in the limit between a learned restricted Markov Tree T of order k, and a distribution from the same family, since with enough (i.e. infinite) data, the error between the estimated and true parameter values will go to zero with probability 1. There are reasons to suppose at least some natural processes generate rankings in this manner. As mentioned earlier, if voters are confronted with a large set of objects, they might order a few that immediately come to mind (i.e. the first k), and

Algorithm 12 An algorithmic description of the learning process for the Markov Tree model of a sequences of length k over a set of candidates C, using a set of ballots B. Note the use of lazy evaluation to avoid building out parts of the tree that are not required.

```
procedure TrainMarkovTree($C$, $k$, $B$)
    Let a TreeNode be a structure containing two hash maps: $\theta : C \rightarrow \mathbb{N}$ and Children : $C \rightarrow$ TreeNodes, with both maps being initially empty.
    Let $T_1$ = new TreeNode()
    for all $b \in B$ do
        for $i = 0; i \leq |C| - k; i++$ do
            Let $T_j = T_1$
            for $j = i; j < i + k; j++$ do
                Let $c_j$ be the candidate ranked $j^{th}$ on $b$
                if $c_j \notin T_j.\theta$.keys then
                    $T_j.\theta[c_j] = 1$
                    $T_j$.Children$[c_j]$ = new TreeNode()
                end if
                $T_j.\theta[c_j] = T_j.\theta[c_j] + 1$
                $T_j = T_j$.Children$[c_j]$
            end for
        end for
    end for
    return $T_1$
end procedure
```

Algorithm 13 An algorithmic description of the imputation process for a top-order ballot b using a Markov Tree model T_1 of sequences of length k over a set of candidates C. The main MarkovTreeImpute procedure first computes the members of candidate set that remain to be imputed, and uses the Trace procedure to find the distribution that ought to be used for the first step. Then it repeatedly samples a candidate, and updates the remaining candidate set before finding the distribution parameters to use for the next sample.

```
procedure TRACE($T_i$, $S$)
    for $i = 1; i \leq |S|; i++$ do
        Let $c_i$ be the candidate ranked $i^{th}$ in $S$
        if $T_i$ is NULL or $c_i \notin T_i.\theta$.keys then
            $T_i = NULL$
        else
            $T_i = T_i$.Children[$c_i$]
        end if
    end for
     return $T_i$
end procedure

procedure MARKOVTREEIMPUTE($C$, $k$, $T_1$, $b$)
    Let $S$ be the top-order sequence over candidates implied by $b$.
    Let $|b|$ be the length of $S$.
    Set $C \leftarrow C \setminus S$
    if $|S| > k$, set $S$ to the *last* $k$ elements of $S$
    Let $T_i$ = Trace($T_1, S$)
    for $i = |b| + 1; i \leq |C|; i++$  do
        if $T_i$ is NULL then
            Let $c_i$ be sampled from Categorical(Uniform(|C|))
        else
            Let $c_i$ be sampled from Categorical($T_i.\theta$)
        end if
        $C \rightarrow C \setminus c_i$
        Set the candidate ranked $i^{th}$ on $b$ to $c_i$.
        Set $S = S.c_i$, and truncate $S$ to the last $k$ elements of $S$.
        Set $T_i$ = Trace($T_1, S$)
    end for
     return $b$
end procedure
```

Algorithm 14 An algorithmic description of the generation process for a top-order ballot b using a Markov Tree model T_1 of sequences of length k over a set of candidates C. The process simply generates a ballot containing a top-order of length 1 by sampling a candidate from the distribution stored at the root of the tree. It then imputes the ballot using Algorithm 13.

```
procedure MarkovTreeGenerate($C$, $k$, $T_1$)
    Let $b$ be an empty ballot.
    Let the first position on $b$ be sampled from Categorical($T_1.\theta$).
     return MarkovTreeImpute($C$,$k$,$T_1$,b)
end procedure
```

then select the next highest ranked object based on the ones they have recently considered (i.e. selecting the $k+1^{th}$ object on the basis of the preceding k). However, the Markov Trees are also capable of correctly learning distributions that were generated by other common processes that are simpler than it is. To demonstrate this, eventual consistency will be shown between the "most probable sequence" (MPS) in a learned restricted Markov Tree and two common families of ranking models, the Mallows Family and the subset of the RUM Family with identical variances across utility distributions.

The "most probable sequence" can be thought of as the sequences most likely to be generated by a model T when Algorithm 14 is applied to it. The sequence is defined in terms of the "most probable alternative" at a given node of T reached by traversing the tree with sequence S. The "most probable alternative" is given by $\hat{c_S} = \text{argmax}_{c \in C} \theta_{S,c,j}$, and the most probable sequence for T at depth j to be $MPS_j(T) = MPS_{j-1}(T), c_{\hat{MPS_{j-1}(T)}}$, with $MPS_1(T) = \hat{c_\emptyset}$.

Theorem 5. *Let M be a Mallows model with centroid ranking $\mu = \mu_1, ..., \mu_{|C|}$ and dispersion parameter ϕ. As the number of total orders sampled from M goes to infinity, the most probable sequence $MPS_{|C|}(T)$ of a restricted Markov Tree T of order k trained on the drawn rankings converges to μ, provided* $0 \leq e^{-\phi} < 1$

Proof. The parameters at the first node of T correspond to the proportions of observed rankings that begin with each candidate C. By definition of the Mallows Model, the total density of rankings that start with μ_i is equal to $\sum_{S \in L(C \setminus \mu_i)} \frac{1}{Z} e^{-\phi d(\{\mu_i, S\}, \mu)}$, where d is the Kendall-Tau distance between the two rankings, and Z is a normalization constant. Note that

$$\frac{\sum_{S \in L(C \setminus \mu_i)} \frac{1}{Z} e^{-\phi\tau(\{\mu_i, S\}, \mu)}}{\sum_{S \in L(C \setminus \mu_j)} \frac{1}{Z} e^{-\phi\tau(\{\mu_j, S\}, \mu)}} = e^{-\phi(j-i)}$$

Therefore, given an infinite sample of rankings drawn from M, a proportion ranking μ_1 first is at least a factor $e^{-\phi}$ larger than the proportion ranking any other candidate first. Therefore, in the estimates of $\theta_{\emptyset,\mu_1,1}$ must also be at least a proportion $e^{-\phi}$ larger than the proportion ranking any other candidate first, in expectation, by the central limit theorem[6].

The remainder of the proof proceeds by induction. Suppose that T's most probable sequence is $\mu_1, ..., \mu_j$. It will be shown that the $j+1^{th}$ element of the most probable sequence will be μ_{j+1}. There are two cases.

In the first case, $j+1 < k$. It is easy to show that, if the first j candidates are fixed to the correct ordering, then by exactly the same line of argument used in the basecase above, candidate μ_{j+1} will appear in the $j+1^{th}$ position a proportion $e^{-\phi}$ more often than other candidates. Observe that,

$$\frac{\sum_{S \in L(C \setminus \{\mu_1, ..., \mu_j\})} \frac{1}{Z} e^{-\phi\tau(\{\mu_1, ..., \mu_j, \mu_{j+1}, S\}, \mu)}}{\sum_{S \in L(C \setminus \{\mu_1, ..., \mu_j, \mu_l\})} \frac{1}{Z} e^{-\phi\tau(\{\mu_1, ..., \mu_j, \mu_l, S\}, \mu)}} = e^{\phi(j+1-i)}$$

where μ_l is here any candidate other than μ_{j+1} that has not already appeared in the most probable sequence. As before, μ_{j+1} will be observed at least a multiple $e^{-\phi}$ more often than any other candidate, and so is the most probable continuation of the most probable sequence.

The second case occurs when $j+1 \geq k$. This case is the most complex. The proof of this case relies on a slightly different formulation of the Mallows model. Mallows [1957] notes that an alternative system for sampling candidates from a Mallows distribution is to sample a ranking by pairwise relations. In the sampled ranking, a pair of candidates c_1, c_2 are ordered such that $c_1 \succ c_2$ with probability p if $c_1 = \mu_i$, and $c_2 = \mu_j$ such that $j > i$, and probability $1-p$ otherwise. $p > 0.5$ by assumption. If an intransitive ranking is sampled, it is rejected and a new ranking is sampled in its place. Mallows [1957] shows this to be equivalent to sampling from a mallows distribution with the same central ordering μ, and a dispursion parameter $\frac{p}{1-p} = e^{-\phi}$.

First, consider the case with a first-order model $k = 1$. By the inductive hypothesis, assume that more instances of μ_1 have been observed than of any other for the start

[6]Note that this holds if α_i is any finite value for all i in the prior parameter distribution, since with enough data the proportions will still converge to within $e^{-\phi}$ in expectation.

of the sequence. For the base case in this inner induction, it will be shown that the subsequence $\mu_i \mu_{i+1}$ occurs more often in the ballots sampled from the Mallows than any other subsequence $\mu_i \mu_l$ for $l \neq i+1$.

Suppose that a partial sequence has been sampled from a Mallows distribution via the pairwise sampling process described above, and that the order of every candidate apart from some candidate μ_l has been fixed with respect to every other. The order of μ_l with respect to every other candidate will now be sampled. Suppose there exist exactly y candidates μ_j such that either $\mu_j \succ \mu_l$ and $\mu_i \succ \mu_j$ or else $\mu_l \succ \mu_j$ and $\mu_j \succ \mu_i$ (i.e. y candidates which lie *between* the μ_l and μ_i in the ground truth ordering). Then the probability of sampling the orderings for μ_l such that μ_l will immediately follow μ_i in the sampled ordering is given by

$$z \cdot \left(\sum_{i=0}^{|C|-y} \binom{|C|-y}{i} p^{2i} (1-p)^{2(|C|-y-i)} \right) \left(\sum_{j=0}^{y} (p(1-p))^y \right)$$

where $z = p$ if $l > i$, and $p - 1$ otherwise. The intuition for this probability is that μ_l will follow μ_i in the sequence if and only if they have identical pairwise orderings with respect to all the other candidates. The first bracketed term is the probability of the two candidates having identical pairwise orderings over the set of candidates that lie in front of or behind both of them in the ground truth. The second bracketed term is the probability of them having identical pairwise orderings over the set of candidates that lie in between them in the ground truth ordering. z is the probability that μ_l follows μ_i. The summations in this expression have closed forms, and so the probability can be simplified to

$$z \cdot (1 - 2p + 2p^2 - p^3)^{|C|-y} (2p - 2p^2)^y$$

Note that if

$$(1 - 2p + 2p^2 - p^3) > (2p - 2p^2)$$

then this equation is maximized when $y = 0$, and when $z = p$. However, the equation

$$(1 - 4p + 4p^2 - p^3) = 0$$

has only one real root, at $p = 1$, and has value 1 at $p = 0$. Therefore, for all permitted values of p, the equation is maximized when $y = 0$ and $z = p$. This occurs for exactly one candidate, μ_{i+1} (the only candidate both adjacent to and following μ_i in the ground truth ordering).

An identical argument can be used when $k > 1$. In this case, an the sequence of interest is $X = \mu_{i-k+1}\mu_i$. Again, the goal is to show that μ_{i+1} will be sampled as a completion of this sequence more often than any other candidate,in expectation. Once again, assume that a partial sequence has already been sampled that contains X, but that has not yet defined any pairwise relationships for some candidate μ_l. The probability of sampling the pairwise relationships for μ_l such that μ_l is placed immediately after X is then

$$z^{|X|} \cdot \left(\sum_{i=0}^{|C|-y-|X|} \binom{|C| - y - |X|}{i} p^{2i}(1-p)^{2(|C|-y-|X|-i)} \right) \left(\sum_{j=0}^{y} (p(1-p))^y \right)$$

Here $z = p$ if $l > i$, and $1 - p$ if $l < i - k + 1$ (note that one of these must be true). Although the quantities have changed slightly, the closed forms for the bracketed expressions are identical:

$$z^{|X|} \cdot (1 - 2p + 2p^2 - p^3)^{|C|-y-|X|}(2p - 2p^2)^y$$

so the probability is again maximized when $y = 0$, and when $z = p$. Once again, this occurs only when $\mu_l = \mu_{i+1}$. □

Theorem 6. *Let M be a Random Utility Model where the utility of candidate c_i is distributed as a Gaussian with mean $\eta_i \in \vec{\eta}$, and standard deviation σ, subject to the constraint that $\eta_i \neq \eta_j$ for all candidates $c_i \neq c_j$. The set of means $\vec{\eta}$ induce a ranking over the candidates $\mu = \mu_1, ..., \mu_{|C|}$, such that if $\mu_i = c_j$ and $\mu_l = c_k$, and $i < l$, then $\eta_j > \eta_k$. As the number of sample rankings drawn from M goes to infinity, the most probable sequence $MPS_{|C|}(T)$ of a restricted Markov Tree T of order k trained on the drawn rankings converges to μ.*

Proof. The proof proceeds by induction. First, consider the distribution of first preferences among rankings drawn from M. For any $\mu_l \neq \mu_1$, the amount by which the subjective utility of μ_1 exceeds the subjective utility of μ_l during the generation of a particular ranking is a random variable distributed as $N(\eta_1 - \eta_l, 2\sigma)$. Since, by definition, $\eta_1 > \eta_l$, this amount is always positive in expectation, so $MPS_1(T)$ will converge to η_1 with infinite data.

Now, suppose that $MPS_j(T) = \mu_j \forall 0 < j < j + 1$. It will be shown that $MPS_{j+1}(T) = \mu_{j+1}$. All $\mu_l \neq \mu_{j+1}$ that have not yet been fixed in $MPS_j(T)$ have the property $\eta_l < \eta_{j+1}$.

If $j+1 < k$, then the amount by which the subjective utility of μ_{j+1} exceeds the subjective utility of μ_l during the generation of a particular ranking is a random variable distributed as $N(\eta_{j+1} - \eta_l, 2\sigma)$, always positive in expectation. □

An interesting implication of both results is that a Markov Tree of any order $k > 1$ will eventually converge so that its most probable sequence matches the centroid of a Mallows or the induced ranking of a RUM, although potentially large amounts of data will be required if there are many candidates involved because the proofs assume infinite data. In the Mallows model for example, with finite data there will be some probability that a candidate other than the correct one appears after some subsequences of the MPS more often than the correct one. If there are many candidates, then the probability that *all* such candidates appear less often than the correct one will be small for small amounts of data. The amount of data needed corresponds to the convergence rate estimated earlier in the chapter, but cannot be determined analytically, because computing the marginals of a Mallows distribution given some evidence is $\#P$-Hard [Lu and Boutilier, 2011a].

7.5 Convergence to Artificial Distributions

The parameter k in the Markov Tree model, which represents the length of the sub-sequence that is used as the basis for imputation, provides explicit control over complexity (i.e. the number of parameters, and the amount of training data needed). As such it is expected that increasing k will allow the model to fit data more precisely, but will also increase the potential for overfitting. In general this parameter must be set according to the difficulty and size of the problem under consideration, but assessing the impact of the parameter on the model in advance will facilitate the configuration of the model in subsequent experiments, and may also provide insights for future practitioners.

To evaluate the impact of k, a number of experiments were performed on synthetic data generated from Mallows Models and RUMs. These experiments used the artificial data generation capabilities of the Prefmine experimental testbed, described in Chapter 6.

Figures 7.4- 7.12 show the Kendall Correlations between the centroid of a Mallows distribution, and the MPS of Markov Trees trained on samples drawn from the distribution. Each point is the average of 400 trees of the same depth, trained on the same number of points drawn from different Mallows models. The horizontal axis shows the number of data points that were drawn (logarithmic scale), and the vertical axis shows the Kendall Correlation. Error bars are not shown, but the largest 95% confidence intervals would have a width of 0.001, which would not visible on the graphs in any case.

Figures 7.4, 7.5 and 7.6 show the convergence results for Mallows distributions with 5 candidates, and $e^{-\phi} = 0.5$, 0.65, and 0.8 respectively. Each graph shows the convergence for four different values of k (1,2,3, and 4), starting with just 10 sequences drawn, and ending with a thousand or more after all or most of the models have fully converged. The figures are best viewed in colour. Figure 7.4 shows a very rapid convergence for all of the models when ballots are drawn from a fairly homogeneous population with a small number of candidates. Simpler models perform less well when there is insufficient data, but converge faster (likely because they have fewer parameters, and because they are able to sample more ballots per sequence). Figure 7.5 shows a similar trend when the ballots are less homogeneous, though more data is required for convergence, and simpler models do not converge as quickly as before. The trend is brought to its logical conclusion in Figure 7.6, where even more data is required for the convergence of the simpler models, and their convergence rates are even slower. However, a clear trend toward convergence is present even in the simplest (i.e. first and second) order models, bearing out the findings in Theorem 5.

Figures 7.7-7.9 display the convergence results for ballots sampled from Mallows distributions with ten candidates instead of 5, for the same parameter settings of $e^{-\phi}$ as in the previous three figures. Five different model depths (1-5) are used. Convergence is much like the cases with 5 candidates. When the dispersion is low (0.5) convergence occurs rapidly. Simple models converge faster, but have lower initial performance than more complex models. When dispersion is higher and the sampled preferences are thus more heterogeneous, as in Figure 7.9, convergence is slower, and the difference in convergence rates is smaller. Also notable in this figure a non-monotonic improvement in performance for the two simplest models considered. This trend is observed much more often when even larger values of $|C|$ are used, and explanations are offered alongside those results.

Figures 7.10-7.12 show similar convergence results when 20 candidates are used. Here models of depth 1,2,3 and 5 are used as in the earlier experiments, as well as much deeper models (19 in the experiments with $e^{-\phi} = 0.5$, 10 in the others). In the simplest case with the most homogeneous preferences, the models appear to converge at rates more akin to the more difficult cases with 10 candidates. It is interesting to note that model depths beyond 5 do not appear to have any meaningful impact on the convergence rates of the model, and even the model of depth 3 appears nearly indistinguishable from these much more complex models. As in the other experiments, the two simplest models started out weakest, but also appear to learn faster than the others. The results for less homogeneous preferences are more surprising. In both sets, the three simpler (i.e. lowest depth) models actually experience performance degradation as more data is added initially, up to a point after which they appear to converge very rapidly, actually reaching the correct MPS before

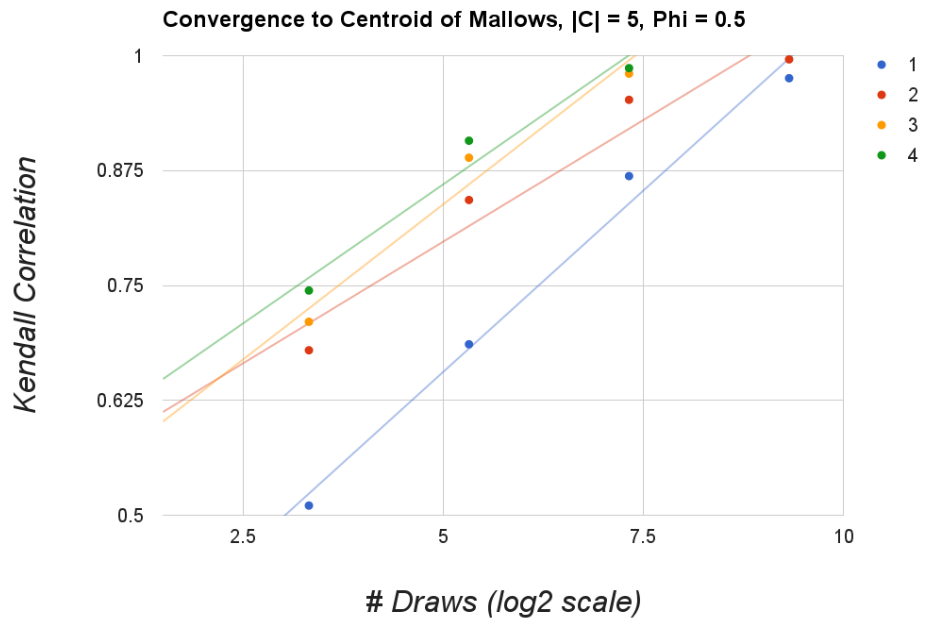


Figure 7.4: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 5$ and $e^{-\Phi} = 0.5$ as more data is drawn from the Mallows.

the more complex models. One possible explanation for this behaviour is that when few ballots are drawn, short sub-sequences are unlikely to appear more than once unless they are correct. However, since the ballots have many possible errors, as more ballots are drawn, there are $20 - k$ possible candidates that might precede the correct one at the end of each portion of the MPS, and the probability that the correct one appears more often then all of the others in the sample is low, even though is expected to appear more often than any individual sequence (i.e. suppose that the probability of the correct ending appearing more often than a particular incorrect one is α. Then the probability of appearing more often than all other endings is something more like $(1 - \alpha)^{|C|-k}$, though α may improve rapidly as more data is drawn). More complex models may avoid this problem because the probability of longer sequences that are incorrect being preserved is exponentially lower than that of shorter sequences. Note that some of the graphs do not run until complete convergence. This is because the larger models can be very time consuming to generate

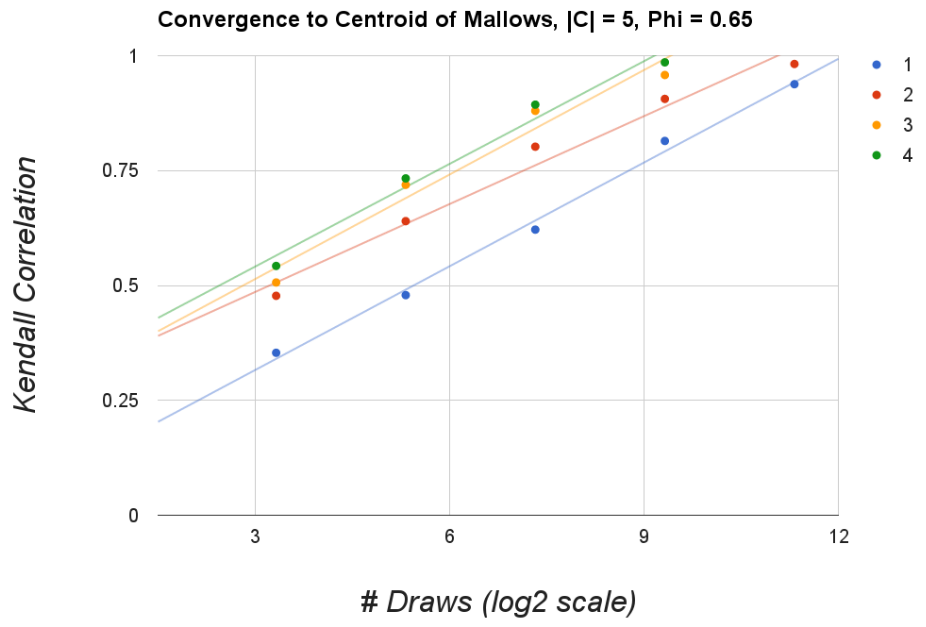


Figure 7.5: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 5$ and $e^{-\Phi} = 0.65$ as more data is drawn from the Mallows.

data for, since the data generation algorithm used is quadratic in the number of candidates.

Finally, Figures 7.13-7.15 show similar convergence result for data sampled from random utility models with different numbers of candidates. As before, the number of rankings is shown in a log scale on the horizontal axis, and the Kendall correlation between the ground-truth ranking and the MPS of various Markov Tree models is shown on the vertical axis. Different lines show the convergence of models of different depths. Convergence appears very similar to the Mallows results, validating the results of Theorem 6 as well.

On the basis of these convergence results, it seems reasonable **to suggest a model depth of 3** for use by practitioners with fewer than 10 or 15 candidates. In all experiments, moving to models that were significantly deeper than this does not appear to provide a very substantial advantage (especially considering the much larger number of parameters, and the much larger memory requirements that can accompany this). Although simpler models ($k = 1$, $k = 2$) appear to make better use of the data and converge faster in some

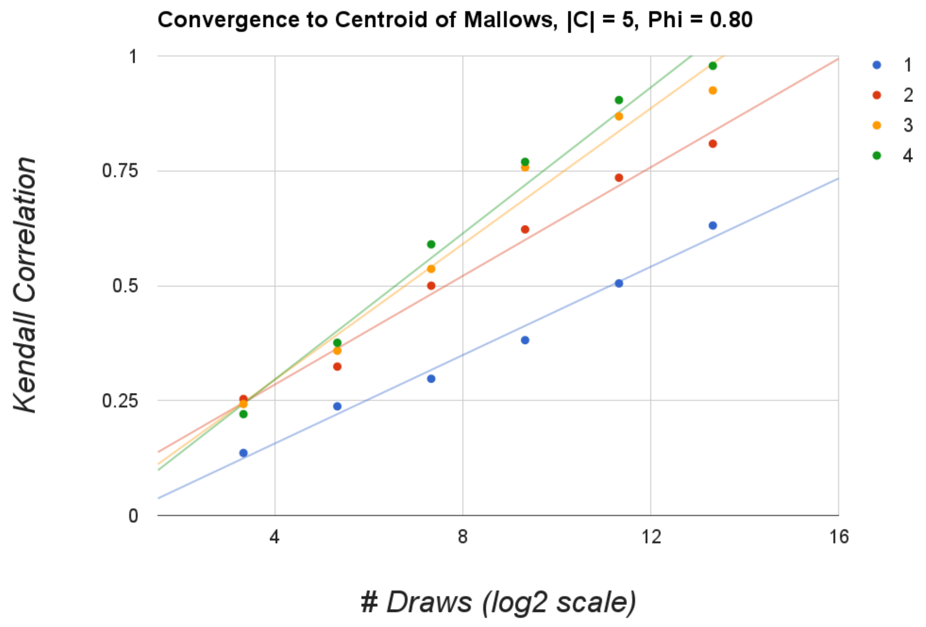


Figure 7.6: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 5$ and $e^{-\Phi} = 0.8$ as more data is drawn from the Mallows.

conditions, they also have much worse performance than a model of depth 3 when the amount of data available is too small. To confirm this, a few trial runs were made with models of different depths on problems generated from each of the Debian and Irish datasets from Preflib [Mattei and Walsh, 2013]. The performance gain for increasing model depth beyond 3 was non-existant for all sets except Debian Logo, a result discussed in detail in the next section.

7.6 Empirical Results

To evaluate Restricted Markov Trees' potential as an imputation method for social choice with partial information, a replication of the experiment design used in Chapter 5 was performed, using the Prefmine system described in Chapter 6. As before, the model was run

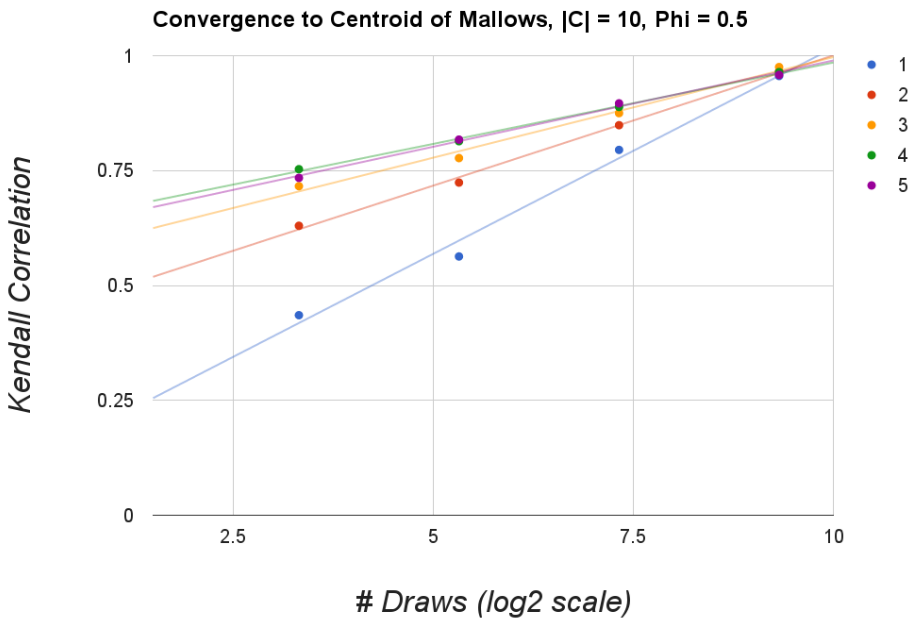


Figure 7.7: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 10$ and $e^{-\Phi} = 0.5$ as more data is drawn from the Mallows.

on each of four voting rules (**K-Approval**, **Borda**, **Copeland**, and **Veto**), and as before performance was measured in terms of the Single Winner Error, Kendall correlation with ground-truth ranking, and First Error Location measurements. For each of the Debian sets, 1,600 problem instances were generated by ablating the complete ballots from each set of electoral data in a fashion that was consistent with the distribution of missing information in the original ballots. 1,600 instances were also generated for the Dublin West set, but only 550 instances for Dublin North, and 100 instances for the Meath set, because of the large number of candidates which made certain parts of the system slower (for instance, the implementation of the **Copeland** rule scales quadratically in the number of candidates). A model depth of $k = 3$ was used for all experiments, as well as a uniform Dirchlet prior for the multinomial distributions at each node in the tree, with a single observation for each candidate. Results are presented in two forms. Tables are used to show the exact values obtained for each performance measure on each dataset, and a statistical analysis

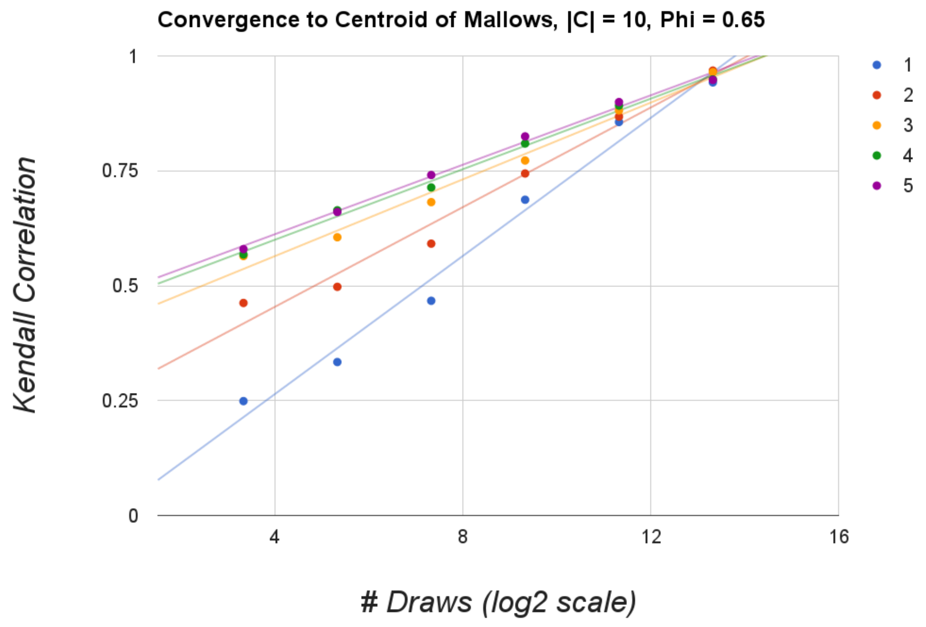


Figure 7.8: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depth to the centroid of a Mallows distribution with $|C| = 10$ and $e^{-\Phi} = 0.65$ as more data is drawn from the Mallows.

provides a definitive comparison between the different methods.

The mean and standard deviation of the observed performance of the Markov-Tree model under each dataset is summarized in Tables 7.4-7.7. Table 7.4 shows performance under the **K-Approval** voting rule. Notable here are a very small number of mistakes when picking the winner on the Debian Logo, Meath, and Debian 2005 sets. In comparison, the logistic regression method made no errors are all under this rule, while the two comparison methods (MMR and the random imputation method) made systematic mistakes on one set. Kendall correlation is also very high, except under Debian Logo. A similar pattern is present under the **Borda** rule results, shown in Table 7.5, where single winner errors are again slightly more frequent than they are when using logistic regression (though much better than the competitors), and the Kendall correlations are all very high, except for Debian Logo, which is lower. Table 7.6 shows a similar pattern, except here single winner performance is actually rather better than logistic regression, which made

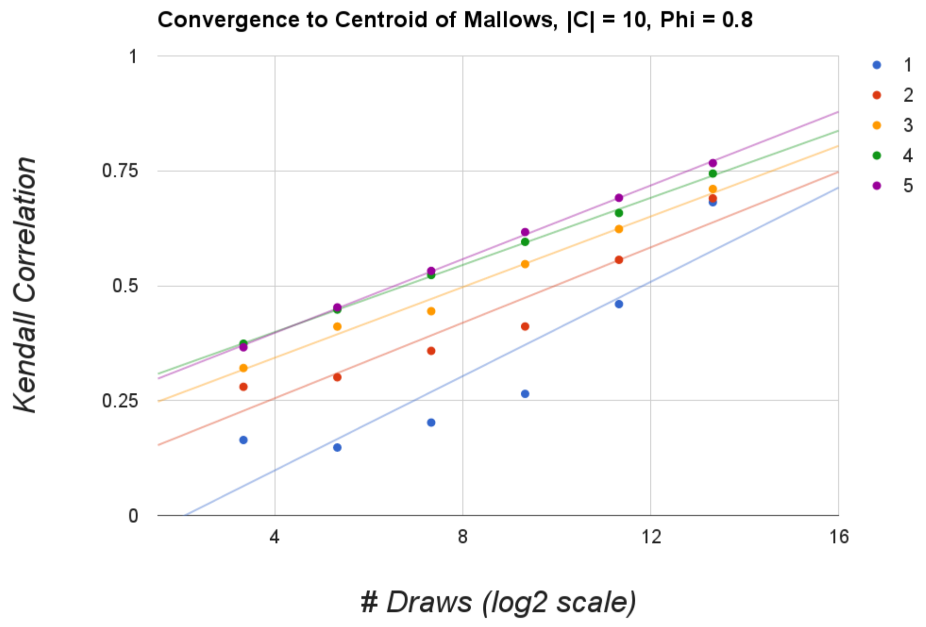


Figure 7.9: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 10$ and $e^{-\Phi} = 0.8$ as more data is drawn from the Mallows.

some substantial mistakes under this rule, particularly on the Meath set. Again, Debian Logo has the lowest Kendall correlation by a significant margin, lower even than the values of sets that are generally harder, like the three Irish sets. Results under **Veto**, shown in Table 7.7, preserve this pattern, with a small advantage in single winner performance, and generally strong Kendall correlations, except for Debian Logo. Although Meath has lower Kendall correlation here than Debian Logo, Meath is also a much harder dataset (i.e. more candidates, greater missingness, and lower worst-case performance possible, as summarized in Chapter 5).

As in Chapter 5, a statistical analysis was performed to determine whether there was a general advantage to using the Markov Tree models instead of the earlier logistic regression models, or either of the two competitors (minimax regret and the random imputation model). For each measurement (Single Winner Error, Kendall correlation, and First Error Location), an analysis of variance (ANOVA) was performed over data from the Markov

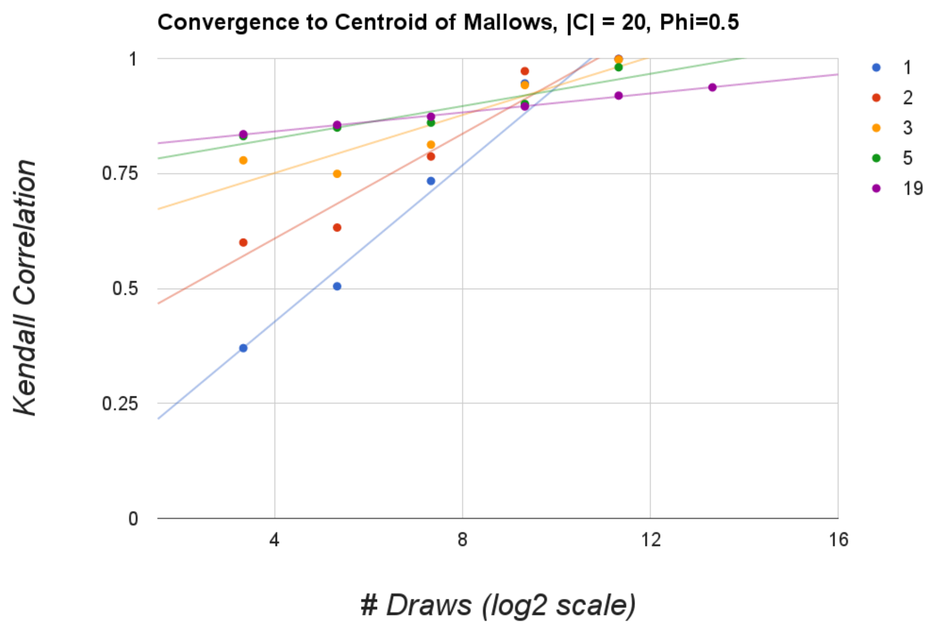


Figure 7.10: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 20$ and $e^{\Phi} = 0.5$ as more data is drawn from the Mallows.

Tree's runs, and the runs using logistic regression, minimax regret, and random imputations. Three factors were used: the choice of method for deciding the election, the voting rule used, and the dataset on which the runs were performed. Additionally, interactions between all factors were considered. In the presence of the other factors (and the interaction terms), there was a highly significant effect from the choice of method ($p < 2e - 16$). Following a detailed exposition of the statistical analyses, a figure is presented to summarize the findings.

For the Single Winner Error measurement, the Markov-Tree method was found to have the lowest error overall, corresponding to an advantage of 0.025-0.056 points over the logistic regression approach, an advantage of 0.097-0.13 points over the random imputation approach, and an advantage of 0.11-0.14 points over minimax regret (all ranges are 99.99% confidence intervals for the mean difference in performance across all datasets and all voting rules). These values correspond to picking the correct winner instead of the second

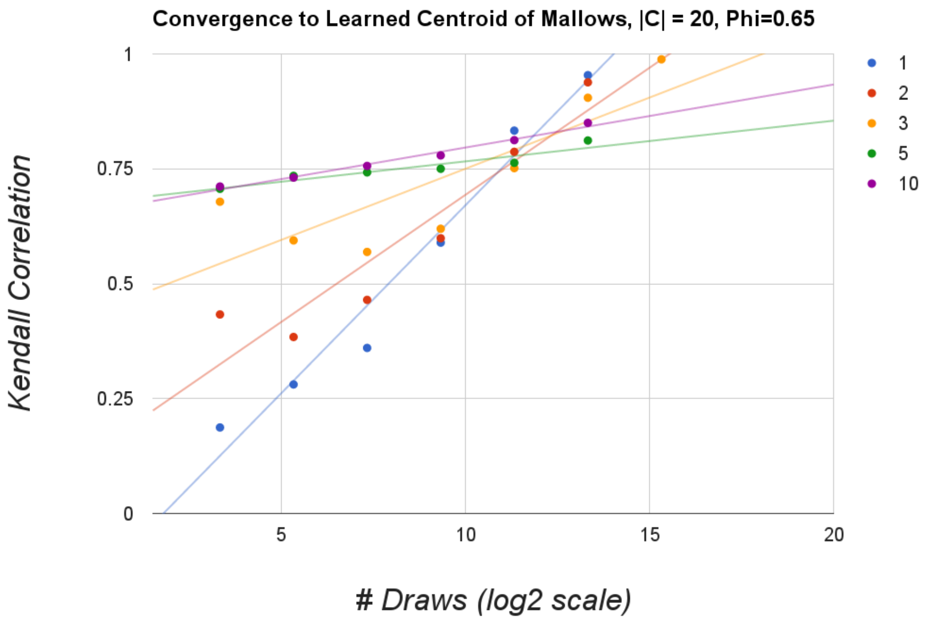


Figure 7.11: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 20$ and $e^{\Phi} = 0.65$ as more data is drawn from the Mallows.

place candidate about 1 in 25 elections more often than the logistic regression approach, 1 in 10 elections more often than the random approach, and 1 in 8 elections more often than MMR. In terms of the results on individual voting rules, under **K-Approval** the Markov Tree approach does slightly worse than logistic regression, with 99% confidence interval of 0.0066-0.013, making its performance identical to that of random or MMR. Under **Borda** it's 0.16-0.18 points better than MMR, 0.15-0.17 better than the random random method, but 0.026-0.034 worse than using logistic regression. Under **Copeland** the Markov Tree approach comes into its own, with an advantage of 0.037-0.051 points over logistic regression, 0.015-0.026 points over MMR, and 0.038-0.050 over the random random approach. Finally, under **Veto** the Markov Tree approach has a very large advantage of 0.13-0.18 over logistic regression, 0.29-0.34 over MMR, and a modest advantage of 0.022-0.026 over the random approach. In addition to besting the other methods overall, it is interesting to note the largest advantage for the Markov Tree approach in picking the winner

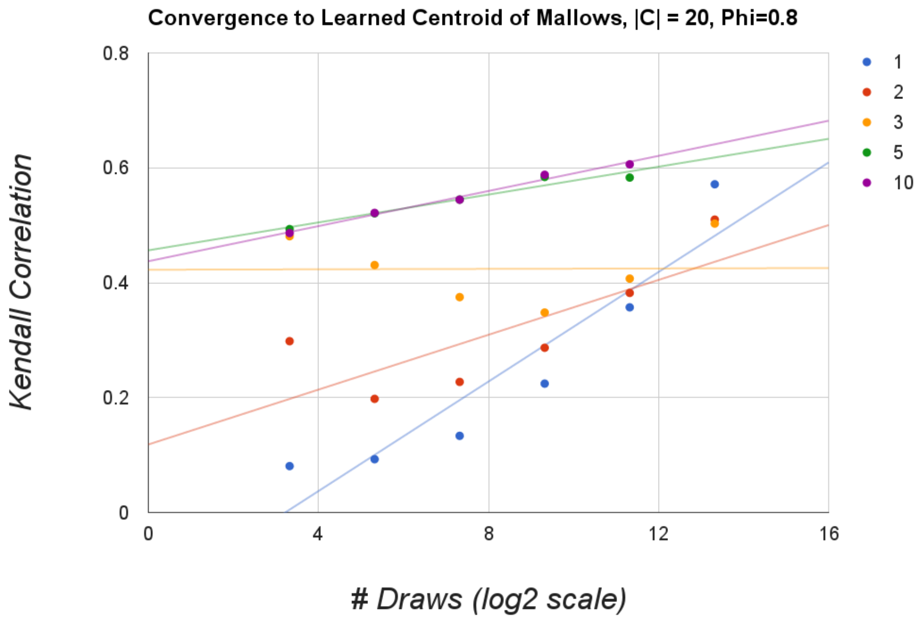


Figure 7.12: Convergence of $MPS_{|C|}(T)$ of Markov Trees of different depths to the centroid of a Mallows distribution with $|C| = 20$ and $e^{\Phi} = 0.8$ as more data is drawn from the Mallows.

of the election is to be found under the harder **Copeland** and **Veto** rules, which are less forgiving of errors in imputed ballots than **Borda** or **K-Approval** are. To impute ballots correctly under Veto, the Markov Tree approach needs to correctly impute all of the missing information. Note also that although the absolute magnitude of the advantage for using the Marove Tree approach may seem small (only about 4% of a point over the logistic regression approach, for instance), this is actually a rather large fraction of the potential improvement. The comparison methods all have reasonably high average performance already, with the logistic regression approach having a mean Single Winner Error of just 0.22 per run. An advantage of 0.04 points is thus 18% of the maximum possible improvement, a significant gain.

Results in terms of the Markov Tree's ability to recover the entire sequence have a very interesting component in the form of the Debian Logo set. As noted earlier, the Kendall correlation of the model on the Logo set is consistently worse than on other sets.

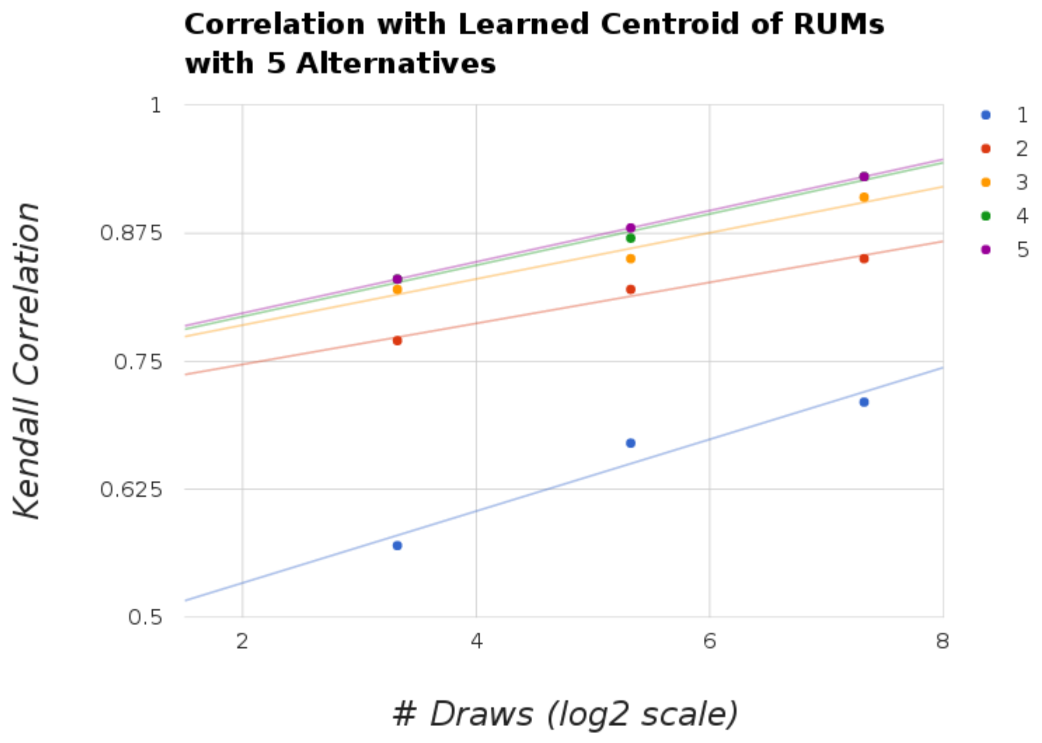


Figure 7.13: Convergence of $MPS_{|C|}(T)$ to the induced ranking of a RUM with **5 candidates**. Number of rankings drawn increases along the x-axis, while the y axis shows the normalized Kendall-Tau distance between the MPS and the induced ranking. Different lines correspond to different values of k (i.e. model depth) for the Restricted Markov Tree.

Reasons for this will be discussed at the end of the section. When Debian Logo is included in the analysis, the Markov Tree approach scores 1.8%-2.6% better than MMR but is worse than the random and logistic regression approaches overall by 1.2%-1.9% 0.9%-1.6% respectively (all reported ranges are 99.99% confidence intervals for the mean difference over all datasets and voting rules). Under **K-Approval**, performance is 0.5%-1.4% worse than logistic regression, 3.2%-3.9% worse than random, and even 0.32%-0.4% worse than MMR, an astonishingly poor performance. Under **Borda** the Markov Tree approach is about 1% worse than all three competing methods. Under **Copeland** however the model does a bit better than all three competitors, outperforming logistic regression by 0.02%-1%, MMR by 10%-12%, and the random imputation approach by 0.36%-0.65%. Finally,

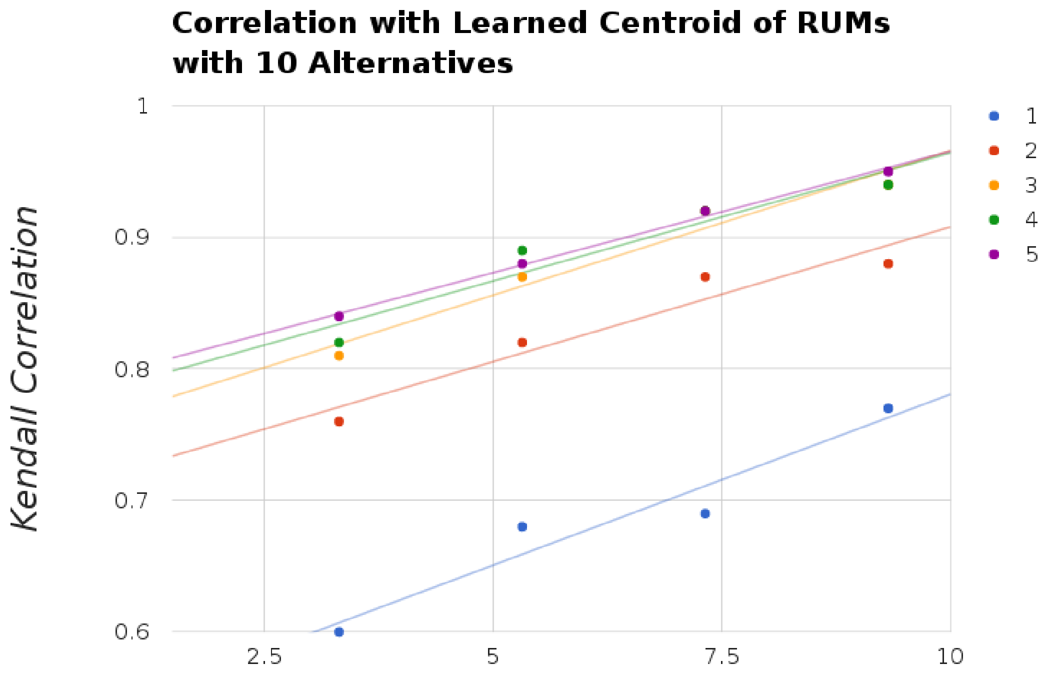


Figure 7.14: Convergence of $MPS_{|C|}(T)$ to the induced ranking of a RUM with **10 candidates**. Formatting is identical to Figure 7.13.

under **Veto** the method again does worse than the random imputation approach by 0.62%-2.4%, worse than the logistic regression approach by 2.2%-4.2%, and better than MMR by 1%-3%. This is a somewhat surprising result, as one might intuitively expect the Markov Tree approach to do better than the classifier approach in terms of recovering the entire sequence. Most advantage for competing approaches seems to be from the **K-Approval** rule, while the Markov Tree approach actually performs best of all on **Copeland**, which is a harder rule as demonstrated by MMR's very poor performance.

Important for later discussion is the finding that if the results for Debian Logo are omitted from the analysis, the Markov Model instead comes within a mean value of 0.002 (i.e. 0.2%) of the random imputation and logistic regression models overall. Although this difference is still statistically significant, it is extremely small, corresponding to a difference of about one extra inversion in the order of adjacent candidates over every five runs on datasets with 14 candidates, and an even smaller effect on sets with fewer

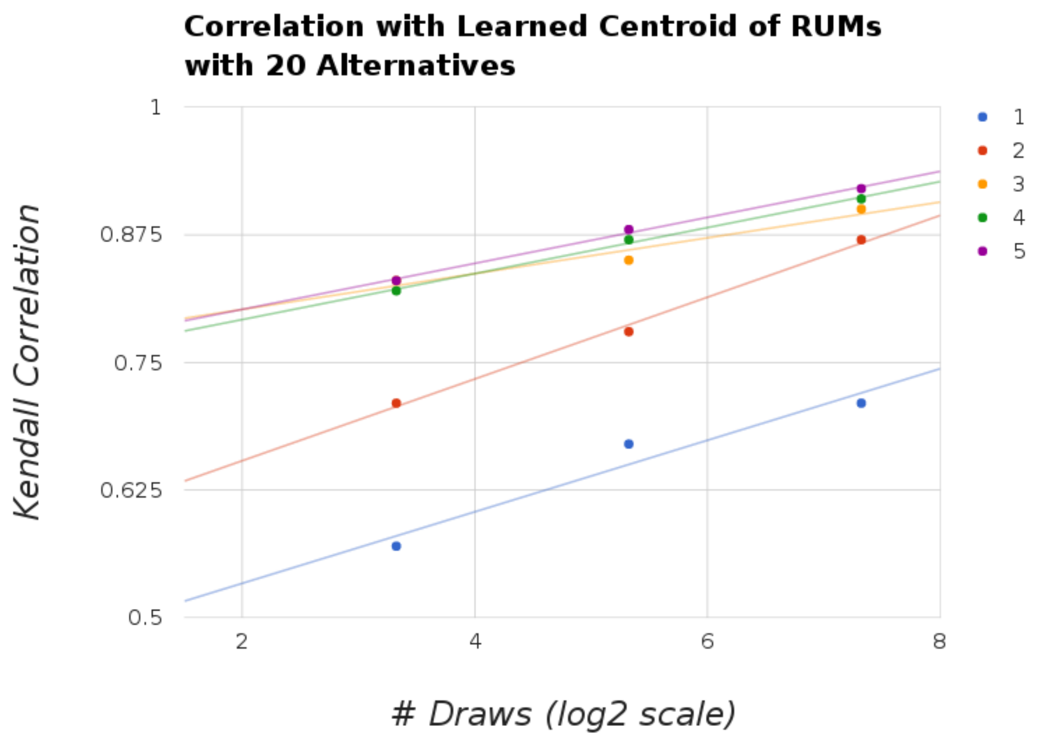


Figure 7.15: Convergence of $MPS_{|C|}(T)$ to the induced ranking of a RUM with **20 candidates**. Formatting is identical to Figure 7.13.

candidates. This suggests that **apart from the Debian Logo set, the Markov Tree approach has substantially better performance when selecting the winner of the election, and essentially identical performance to the best performing methods when ordering the candidates overall**, and should therefore be preferred if the poor performance on Logo can be explained. Figure 7.16 provides a visual summary of these conclusions, showing performance measures averaged across all datasets except Debian Logo, and all voting rules.

Interestingly, there is a plausible explanation for why performance was lower on the Debian Logo set than on the others. While for all other sets, decreasing *or* increasing k from 3 reduced overall performance on the provided data, this was not true on the Debian Logo set. On Debian Logo performance increases as k is increased up to 3, but then continues increasing (slowly) with k until reaching a Kendall correlation of 0.89 at $k = 6$,

	First Error	Single Winner	τ	$\|C\|$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.89 ± 0.80	0.02 ± 0.12	1.00 ± 0.01	7	15.5
Debian 2006	6.90 ± 1.34	0.00 ± 0.00	0.97 ± 0.04	8	14.8
Debian 2007	2.73 ± 1.84	0.00 ± 0.00	0.90 ± 0.04	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	2.24 ± 0.85	0.07 ± 0.26	0.71 ± 0.11	8	40.0
Dublin North	4.02 ± 0.17	0.00 ± 0.00	0.89 ± 0.02	12	58.5
Dublin West	2.51 ± 1.02	0.00 ± 0.00	0.82 ± 0.09	9	50.8
Meath	2.80 ± 0.55	0.08 ± 0.31	0.82 ± 0.04	14	66.8

Table 7.4: Table showing the mean First Error Location, Single Winner Error, and Kendall correlation for a Markov Tree of order 3, averaged across many runs on each of ten different datasets under the **K-Approval** voting rule. The ranges shown with $\pm$ are the sample standard deviations for these performance measures. The number of candidates in the election, and the percentage of data that is missing are also shown for reference.

after which it falls slightly to a correlation of 0.85 for the full tree, which has depth equal to the number of candidates (this final fluctuation maybe random). This is suggests that the sequences in the Debian Logo set cannot be embedded in a smaller dimensional space very readily: to predict the next candidate in a sequence, all preceding candidates are required. A probable reason for this is because the candidates in this contest are potential *logos* for the Debian project, and could be ordered on any number of different axes depending on the subjective tastes of the electorate. For example, while it is easy to believe that in political contests the electorate might agree on a 1 or 2 dimensional embedding of the candidates, it is not at all clear what it would mean for voters to agree that 'Vase' is to the left of 'Spiral' (two example candidates, shown in Figure 7.17. Thus, it is to be expected that the Markov Tree would struggle with learning the order of candidates on this set. In contrast, political contests like the other Debian sets and the Irish sets, tend to have relatively low-order embedding, as candidates differentiate themselves on a small number of issues, in a fairly clear fashion. This explains the high performance of our model on those sets. This affirms the idea that the dimensionality of voters' collective preferences should guide practitioner's parameterization of the Markov Tree approach. For example, if the preferences of the Martian mining companies are thought to be related to the position

	First Error	Single Winner	τ	$\lvert C\rvert$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	13.6
Debian 2005	6.12 ± 1.90	0.00 ± 0.02	0.98 ± 0.04	7	15.5
Debian 2006	6.82 ± 2.34	0.00 ± 0.00	0.98 ± 0.03	8	14.8
Debian 2007	5.34 ± 3.87	0.45 ± 0.56	0.97 ± 0.04	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	2.77 ± 1.30	0.00 ± 0.06	0.79 ± 0.10	8	40.0
Dublin North	4.20 ± 0.48	0.00 ± 0.00	0.88 ± 0.04	12	58.5
Dublin West	2.35 ± 1.51	0.00 ± 0.00	0.93 ± 0.03	9	50.8
Meath	2.05 ± 0.38	0.01 ± 0.10	0.84 ± 0.04	14	66.8

Table 7.5: Table showing the mean First Error Location, Single Winner Error, and Kendall correlation for a Markov Tree of order 3, averaged across many runs on each of ten different datasets under the **Borda** voting rule. The ranges shown with $\pm$ are the sample standard deviations for these performance measures. The number of candidates in the election, and the percentage of data that is missing are also shown for reference.

of each mining site (a 2-dimensional property) and the quality of the site (a 1 dimensional property, say), it would be reasonable to consider a model that uses at least $k = 3$.

7.6.1 Bias

An interesting property of the Markov Tree approach is that it has a built-in way to mitigate the problems of bias against less popular candidates, via the choice of prior. In the experiments used in this chapter, the prior was always uniform, and used just one sample per candidate. However, if bias is a concern, then a stronger prior, with more observations per candidate could be used to reduce the impact of observing lots of data.

7.7 Summary

This chapter proposed a new machine learning algorithm for use imputing the preferences of voters. The algorithm is based on the idea of learning restricted Markov Trees from

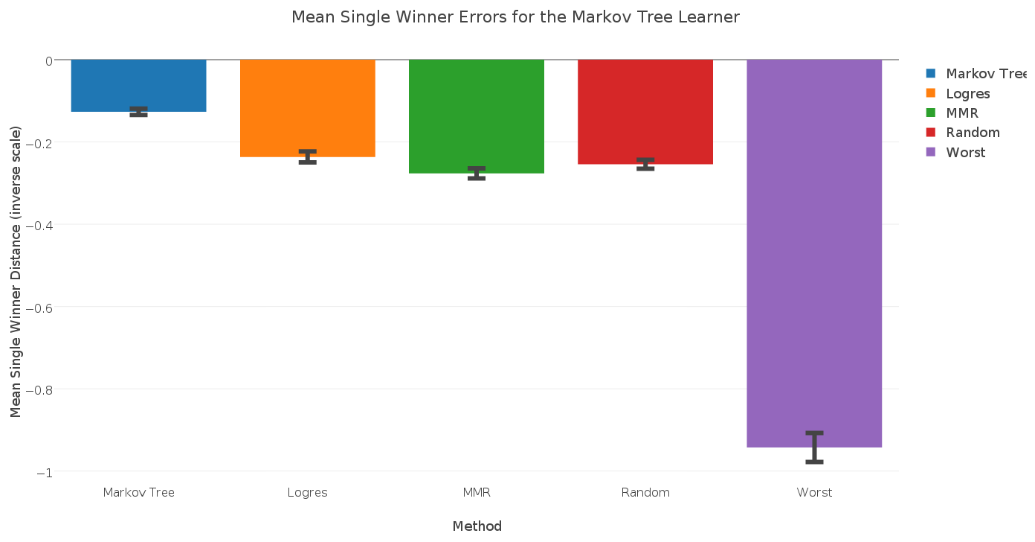
	First Error	Single Winner	τ	$\|C\|$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	4.85 ± 0.75	0.04 ± 0.19	0.98 ± 0.11	5	13.6
Debian 2005	6.97 ± 0.39	0.00 ± 0.07	1.00 ± 0.01	7	15.5
Debian 2006	6.92 ± 1.00	0.00 ± 0.00	0.96 ± 0.04	8	14.8
Debian 2007	6.49 ± 2.00	0.00 ± 0.00	0.95 ± 0.05	9	19.1
Debian 2010	5.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	4.90 ± 0.80	0.00 ± 0.00	0.68 ± 0.09	8	40.0
Dublin North	5.70 ± 3.01	0.02 ± 0.13	0.91 ± 0.06	12	58.5
Dublin West	4.25 ± 2.06	0.52 ± 0.49	0.84 ± 0.10	9	50.8
Meath	2.29 ± 1.39	0.29 ± 0.45	0.84 ± 0.04	14	66.8

Table 7.6: Table showing the mean First Error Location, Single Winner Error, and Kendall correlation for a Markov Tree of order 3, averaged across many runs on each of ten different datasets under the **Copeland** voting rule. The ranges shown with $\pm$ are the sample standard deviations for these performance measures. The number of candidates in the election, and the percentage of data that is missing are also shown for reference.

data, and is able to efficiently learn the preferences of voters from data, provided that voters collectively view the candidates as embedded in a low dimensional space. The proposed implementation of the model uses lazy expansion of the tree and hash maps to minimize memory usage and maximize speed. The algorithm is also straightforward to run in parallel. The proposed model has a theoretically bounded convergence rate, and was shown to converge to the central rankings of Mallows and RUM distributions when provided with enough preferences sampled from such a distribution. The model was empirically shown to perform better in terms of deciding the outcome of the election (i.e. Single Winner Error) than any competing approach, but was less good at recovering the entire ordering of candidates. The reason for this was demonstrated to be the result of one of the datasets involved (Debian Logo) not having a low-dimensional embedding of the candidates. It is therefore reasonable to conclude that the Markov Tree learner should be preferred to competing approaches *provided the candidates can be embedded in a low dimensional space.*

	First Error	Single Winner	τ	$|C|$	% Missing
Debian 2002	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	11.9
Debian 2003	3.55 ± 1.92	0.30 ± 0.42	0.93 ± 0.10	5	13.6
Debian 2005	1.77 ± 1.19	0.83 ± 1.10	0.75 ± 0.13	7	15.5
Debian 2006	2.01 ± 1.67	0.74 ± 0.80	0.82 ± 0.10	8	14.8
Debian 2007	2.08 ± 1.53	0.78 ± 1.05	0.79 ± 0.10	9	19.1
Debian 2010	4.34 ± 1.49	0.13 ± 0.31	0.97 ± 0.07	5	11.0
Debian 2012	4.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00	4	13.2
Debian Logo	1.15 ± 0.50	2.23 ± 1.65	0.43 ± 0.18	8	40.0
Dublin North	2.82 ± 1.09	0.18 ± 0.49	0.49 ± 0.09	12	58.5
Dublin West	1.98 ± 0.46	0.12 ± 0.36	0.49 ± 0.11	9	50.8
Meath	1.00 ± 0.00	5.03 ± 1.57	0.41 ± 0.09	14	66.8

Table 7.7: Table showing the mean First Error Location, Single Winner Error, and Kendall correlation for a Markov Tree of order 3, averaged across many runs on each of ten different datasets under the **Veto** voting rule. The ranges shown with $\pm$ are the sample standard deviations for these performance measures. The number of candidates in the election, and the percentage of data that is missing are also shown for reference.



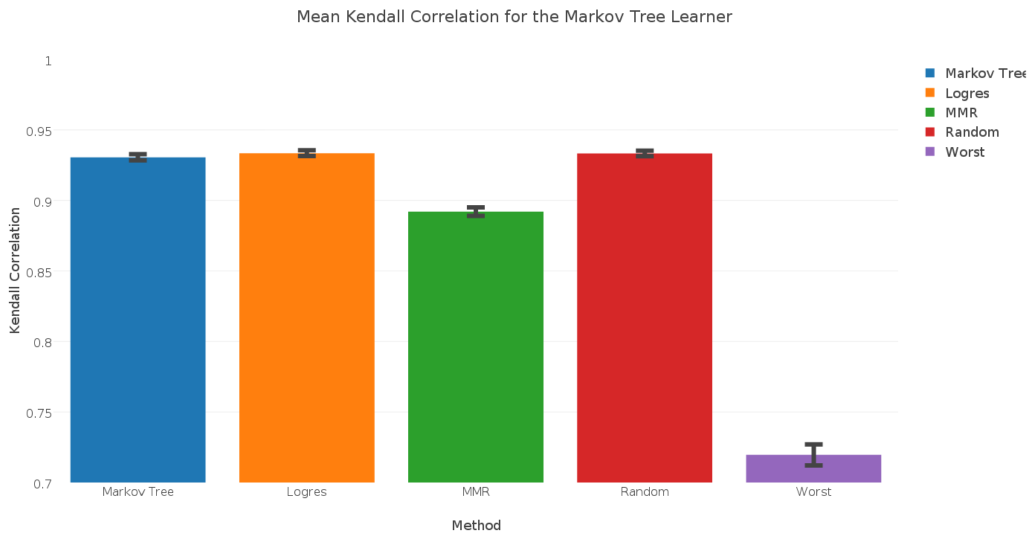


Figure 7.16: Summary of performance for the Markov Tree Learner compared with three competitors and the worst case model. (Top): Single Winner Error (inverse scale). Higher values indicate better performance, 0 is the best possible. (Bottom): Kendall Correlation. Higher values indicate better performance, 1 is the best possible.

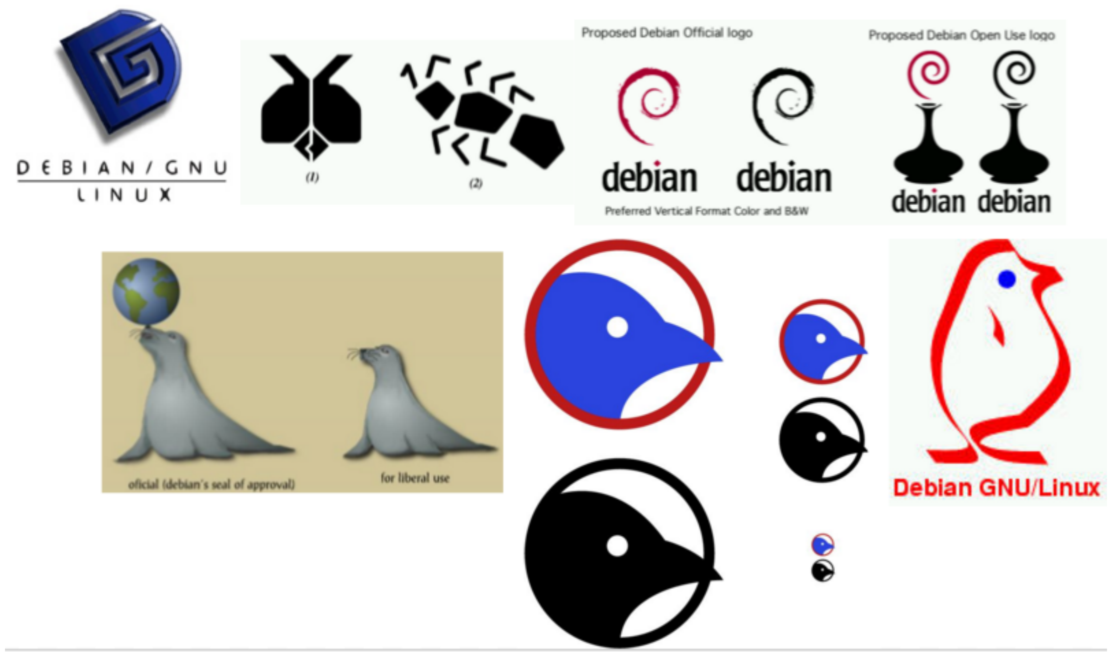

DEBIAN / GNU
LINUX
(1)
(2)
Proposed Debian Official logo
debian
debian
Preferred Vertical Format Color and B&W
Proposed Debian Open Use logo
debian
debian
oficial (debian's seal of approval)
for liberal use
Debian GNU/Linux

Figure 7.17: The candidate logos from the Debian Logo election. The logos are quite different from each other, and it is not at all clear how candidates might collectively view them within the same space. Logos were recovered from the Debian project's website: https://www.debian.org/vote/1999/vote_0004.

Chapter 8

Axiomatic Analysis

If we exclude the possibility of interpersonal comparisons of utility, then the only methods of passing from individual tastes to social preferences which will be satisfactory and which will be defined for a wide range of sets of individual orderings are either imposed or dictatorial.

Kenneth Arrow [2012]

As established in previous chapters, one can view imputation-based social choice methods, which are compositions of imputation algorithms and conventional social choice functions, as "black-box" social choice functions for use in deciding elections with partial ballots, that is, as social choice functions that map from a set of partial ballots to an outcome. Knowing that such techniques are social choice functions gives rise to the question of whether they are *fair* social choice functions. That is, whether the decisions they make are typically (or consistently) representative of the will of the electorate.

Chapters 2 discussed prior work characterizing the "fairness" of different electoral systems using an axiomatic approach. Axiomatic approaches begin by specifying a set of mathematically rigorous properties that would seem to be required to make an electoral system fair. The fairness of a particular system can then be expressed as the extent to which it is capable of upholding the axioms in question, especially as compared to other systems of interest. There are three significant sets of axioms to consider. First, Arrow's axioms constitute a *minimal* set of criteria which any sensible electoral system should satisfy. Namely (and informally):

1. Non-Dictatorship: The outcome must depend on at least two voters' preferences

2. Non-Imposition: For any number of voters, there exists *some* set of ballots that voters could submit to produce any particular outcome[1].

3. Monotonicity: If a voter moves a candidate X higher in their (complete) ballot, then X's overall position cannot worsen in the outcome.

4. Universality: Any (complete) ordering of the candidates must be an acceptable ballot for any voter to cast.

5. Independence of Irrelevant Alternatives: The position of candidate Z on a ballot must not affect the relative ordering of candidates X and Y in the outcome.

The closely related Gibbard-Satterthwaite Axioms are similarly meant to constitute a minimal set of requirements for any voting system *when voters behave strategically.* They are Axioms 1 and 2 of Arrow's set above, along with

6. Strategy-Proofness: No voter can bring about a more-preferred outcome by reporting a (complete) ordering of the candidates that differs from their true preferences.

The proofs of both the Gibbard-Satterthwaite and Arrow's theorems, which show that the sets of axioms proposed by the authors in question are mutually incompatible, assume that voters have all specified complete preference information. That is, the voting systems considered by these theories assume that a ballot is a complete linear ordering of the set of alternatives. It is not immediately obvious that a system operating on *incomplete* preferences should be subject to these constraints, and indeed, the question was not considered in full detail until rather recently. In a generalization of Arrow's theorem, Pini et al. [2009] show that allowing both the votes and the outcomes to be selected from the space of partial orders over the candidates does not provide a circumvention of Arrow's theorem. A similar result by Reffgen [2011] shows that the Gibbard-Satterthwaite theorem holds under even very weak orders over the candidates, including a system where ballots consist of just a Top-2 ordering.

[1]Arrow's theorem can be expressed using two different sets of axioms. The set in Chapter 2 included the Unanimity axiom. Here, Non-Imposition and Monotonicity are used. A method that satisfies both of these and I.I.A. will always also satisfy Unanimity. The larger set of axioms is used here because it has a more natural mapping to the axioms defined for imputation rules later in this section.

In addition to considering fairness criteria for complete ballot systems in a partial ballot context, one might reasonably suppose that there exist properties important *only* (or at least, primarily) to the fairness of electoral systems that operate over partial ballots, and not to the fairness of systems that utilize complete ballots. Woodall's axioms [Woodall, 1987] [Woodall, 1994] serve this purpose by considering a number of specialized monotonicity properties focused on top ordered preferences. These criteria are important particularly in the context of voting systems like the popular **Single Transferable Vote**, which do not satisfy Arrow's montonicity in general, but which *do* satisfy many important subtypes of monotonicity. Many of the imputation algorithms considered in this thesis can respond to preferences in a non-monotonic fashion, so extensions of Woodall's axioms may be important when comparing them.

Finally, an important question, explored later in this chapter, follows from past work combining either several voting rules together or several classifiers together. Both efforts have led to the surprising result that combinations can be more (or less) than the sum of their parts. In the voting context Nina Narodytska et al. [2012] and Walsh and Xia [2012] show that combinations of voting rules that are not computationally strategyproof (i.e. in which computing a strategic ballot requires a polynomial number of operations) can yield voting rules that *are* difficult to manipulate. In the learning context, meta-classifiers like Boosting [Freund and Schapire, 1997] and Bootstrap Aggregation [Breiman, 1996] techniques rely on the insight that averaging the outputs of many simple classifiers trained with the same algorithm, but with different subsamples of the input data, can provide better performance than training a single (complex) classifier on the input data as given. Such meta-classification approaches can even be used to combine disparate classifiers that have been trained on entirely separate views of the same data [Kittler et al., 1998].

In all of these works, combinations of voting systems *or* combinations of classifiers allowed for systems which exhibited properties beyond those of the individual components. It is therefore natural to suppose that combinations of voting systems *and* classifiers can produce the same result. For instance, combining a certain imputation system with any voting rule might yield a joint system that will not satisfy Arrow's monotonicity requirements. Alternatively, perhaps systems that are computationally vulnerable to manipulation could be protected by combining them with a more complex or stochastic machine learning algorithm. General questions of this nature are considered near the end of the chapter.

8.1 Formal Problem Statement

This chapter is concerned with electoral systems that decide the outcome of partial ballot elections. It does not assume any particular underlying model for the production of these ballots.

A **partial ballot election** consists of a set of ballots (votes) V belonging to a set of voters A and a set of candidates C. Each voter casts a single ballot, and the ballot of voter A_i is denoted by $b_i \in B$. Importantly, the way in which ties are broken can be pivotal in axiomatic analysis. In this chapter, it is assumed that ties are broken using a unique, lexicographic ordering over C, such that candidates appearing earlier in the order win ties over those who appear later in the order.

A ballot takes the form of a **top order** over C, rather than a total order or a partial order. This is done both to simplify analysis, and because the data used in the empirical validations of earlier chapters came in the form of top orders. A top order is defined as a tuple (L, G, C), where C is the set of candidates that *could* appear in the order, $G \subset C$ is the *ground set*, the set of candidates that *do* appear in the order, and L is a linear order over the ground set G. L is treated as both a binary relation over G (e.g. xLy means x precedes y in L), and as a list to which items may be appended (an appended candidate c comes after every candidate that was defined in the original ordering). The set of permissible ballots is defined recursively, where $\mathcal{T}(C)_1$ is the set of all ballots that rank just one candidate, and $\mathcal{T}(C)_i$ is the set of all ballots that rank exactly i candidates:

$$\mathcal{T}(C)_1 = \{(c, \{c\}, C) | c \in C\}$$

$$\mathcal{T}(C)_i = \{(L \cdot c, G \cup c, C) | (L, G, C) \in \mathcal{T}(C)_{i-1} \wedge c \notin G \wedge c \in C\}$$

where $L \cdot c$ implies concatenation of c onto L. The set of permissible ballots is given by $\bigcup_{1 \leq i \leq |C|} \mathcal{T}(C)_i$. It is said that a candidate a precedes a candidate b on a top-ordered ballot if $a \in G \wedge b \notin G$, or $a \in G \wedge b \in G \wedge aLb$. For conciseness, a ballot b_i is often treated as having an associated binary relation $\succ_i$, such that if aL_ib or $a \in G_i \wedge b \notin G_i$, then $a \succ_i b$.

Less formally, if a candidate is in the ground set of the ordering, then the voter has indicated that they prefer the candidate to all candidates not in the ground set, and also to any of the other candidates in the ground set that appear after it in the linear order L. This limits voters to specifying one or more top choices, which must be ordered relative to each other, while leaving the remaining candidates unordered (tied) at the bottom of the ballot.

A **partial ballot election** is decided (in this chapter) by a **social choice function** $S : \mathcal{T}(C)^{|A|}_{|C|} \rightarrow O$, mapping the set of possible ballots the voters could collectively produce onto the set of outcomes O. O could be either C, if the function will only pick a winner, or the set of all possible linear orders over C, $\mathcal{L}(C)$, if the function will rank all the candidates.

The family of social choice functions considered in this chapter is comprised of pairs of **imputation methods** and social choice functions for **complete ballot elections**. An imputation method is a mapping $I : \mathcal{T}(C)^{|A|}_{|C|} \rightarrow \mathcal{L}(C)^{|A|}$, with the property that its output set is a consistent extension of its input. That is, for every $x, y \in C$, if k ballots $b_i \in B$ have $x \succ_i y$, then at least k ballots $b_j \in I(B)$ also rank $a \succ_j b$. A social choice function for complete ballot elections is given by $S : \mathcal{L}(C)^{|A|} \rightarrow O$. It should be apparent that for any combination of imputation method and social choice function for complete ballots, $S(I(B))$ is a social choice function for partial ballot elections.

It will occasionally be useful to write about pairs of imputation methods and social choice functions in a way that unambiguously shows we refer to their combination, but without reference to a set of ballots. In this case, given social choice function S and imputation method I, $S \oplus I$ is used to indicate the name of the combined function that produces $S(I(B))$ on a ballot set B.

8.2 Methods Under Consideration

The chapter will consider the properties of several simple imputation methods and several common social choice functions for partial ballot elections. Some further notation is required to describe the imputation functions under consideration. First, a top-order ballot $b_i = \{L_i, G_i, C\}$ is said to be a **prefix** of another top-order ballot $b_j = \{L_j, G_j, C\}$, if and only if $G_i \subset G_j$ and $\forall a, b \in G_i$, $aL_ib \rightarrow aL_jb$. To indicate that b_i is a prefix of b_j, the notation $b_j \models b_i$ is used, which may be read as "b_j models b_i", or "b_j extends b_i" . Finally, $\text{Pos}(b_j, x)$ denotes the candidate a such that for exactly x other candidates b in G_j, bL_ja, that is, the candidate that is outranked by exactly x others on ballot b_j.

The imputation methods that will be considered (which indicate how values will be imputed if voters have not indicated preferences) are:

- **Hot Deck Classifier**: This method picks a single ballot, which is denoted b_1 without loss of generality, and imputes every other ballot according to the ordering specified by b_1 (effectively behaving like a dictatorial voting rule). b_1 is imputed according to the lexicographic tie-breaking order if necessary. Then every other ballot b_i is imputed as follows:

$$\text{HD}(b_i = \{L_i, G_i, C\}) = \begin{cases} b_i & \text{if } \forall a, b \in C, a \succ_i b \text{ or } b \succ_i a \\ \text{HD}(\{L_i x, G_i \cup x, C\}) & \text{otherwise} \end{cases}$$

where

$$x = \operatorname*{argmin}_{0 \leq j < |C|} \text{Pos}(b_1, j) \notin G_i$$

- **Plurality Classifier**: This method extends ballots using the most common extensions found in the input set, or via the lexicographic ordering of C if more than one candidate is tied for the most common extension. Formally, for every ballot $b_i \in B$, the Plurality Classifier's output set will contain a ballot given by the recurrence:

  $$\text{PC}(b_i = \{L_i, G_i, C\}) = \begin{cases} b_i & \text{if } \forall a, b \in C, a \succ_i b \text{ or } b \succ_i a \\ \text{PC}(\{L_i x, G_i \cup x, C\}) & \text{otherwise} \end{cases}$$

  where
  $$x = \operatorname*{argmax}_{c \in C \setminus G_i} |\{b_j | (b_j \in B) \wedge (b_j \models b_i) \wedge \text{Pos}(b_j, |G_i|) = c\}|$$

  and, in the event that multiple elements of G_i are maximal, the one appearing first in the lexicographic tie-breaking ordering over C is selected.

- **Proportionate Classifier**: This method extends ballots in a fashion proportionate to the distributions of extensions found in the input set, rather than assigning all ballots to the most common extension. Formally, let B be the multi-set of original ballots. B can be partitioned into subsets of ballots with identical length, $X_i = \{b_j = \{L_j, G_j, C\} | b_j \in B \wedge |G_j| = i\}$. Portions of these subsets X_i can then be imputed using particular subsets Y, as described below.

  For every pair of candidates (a, b), let $X_{1,a} = \{b_j = \{L_j, G_j, C\} | b_j \in X_1 \wedge \text{Pos}(b_j, 0) = a\}$, $Y_{2,a} = \{b_j = \{L_j, G_j, C\} | b_j \in X_2 \wedge \text{Pos}(b_j, 0) = a\}$ and $Y_{2,a,b} = \{b_j = \{L_j, G_j, C\} | b_j \in Y_{2,a} \wedge \text{Pos}(b_j, 1) = b\}$.

The proportionate classifier imputes the second candidate on each ballot in X_1 in proportion to the fraction of extensions observed that rank each of the different candidates in second place. This is slightly complicated by the discrete number of ballots that can be imputed, an issue addressed by ordering the candidates according to the lexicographic tie-breaking order, such that $c_i \in C$ precedes $i-1$ candidates in the order, and preferentially imputing completions that come higher in the order. More formally, the number of ballots in X_1 that rate that aLc_i following the imputation of a second candidate is given by:

$$n_{|C|-1} = \left\lceil |X_{1,a}| \frac{|Y_{2,a,c_{|C|-1}}|}{|Y_{2,a}|} \right\rceil$$

for the top candidate in the tie-breaking order, and for the others:

$$n_i = \max(\lceil (|X_{1,a}| - \sum_{|C|>j>i} n_j) \frac{|Y_{2,a,c_i}|}{|Y_{2,a}| - \sum_{|C|>j>i} |Y_{2,a,c_j}|} \rceil, 0)$$

After X_1 has been imputed, the Proportionate classifier proceeds to impute all third preferences, then all fourth preferences, and so on. More formally, for imputations of subsequent candidates, the Proportionate Classifier imputes the sets $Z_i = X_i \cup I(Z_{i-1})$, where $Z_1 = X_1$ in exactly the same fashion. For every ordering over i candidates L^* and every candidate $b \in C \setminus L^*$, we let $Z_{i,L^*} = \{b_j = \{L_j, G_j, C\} | b_j \in Z_i \wedge L_j = L^*\}$, $Y_{i+1,L^*} = \{b_j = \{L_j, G_j, C\} | b_j \in X_{i+1} \wedge L_j = L^*\}$ and $Y_{i+1,L^*,b} = \{b_j = \{L_j, G_j, C\} | b_j \in Y_{i+1,L^*} \wedge \text{Pos}(b_j, i) = b\}$.

The Proportionate Classifier imputes Z_i by ordering the candidates according to the lexicographic tie-breaking order, such that $c_i \in C$ precedes i candidates in the order. The number of ballots in Z_{i,L^*} that rate that $\text{Pos}(b_j, i) = b$ following imputation is given by:

$$n_{|C|-1} = \lceil |Z_{1,L^*}| \frac{|Y_{2,L^*,c_{|C|-1}}|}{|Y_{2,L^*}|} \rceil$$

for the top candidate in the tie-breaking order, and for the others:

$$n_i = \max(\lceil (|Z_{1,L^*}| - \sum_{|C|>j>i} n_j) \frac{|Y_{2,L^*,c_i}|}{|Y_{2,L^*}| - \sum_{|C|>j>i} |Y_{2,L^*,c_j}|} \rceil, 0)$$

- 1st **Order Markov Tree**: This is the model described in Chapter 7, with depth parameter $k = 1$. It imputes ballots according to the recurrence:

$$\text{MT}(b_i = \{L_i, G_i, C\}) = \begin{cases} b_i & \text{if } \forall a, b \in C, a \succ_i b \text{ or } b \succ_i a \\ \text{MT}(\{L_i x, G_i \cup x, C\}) & \text{otherwise} \end{cases}$$

  where

$$x = \operatorname*{argmax}_{c \in C \setminus G_i} |\{b_j | b_j \in B \wedge \exists 0 \leq h < |C| - 1 \; s.t. \; \text{Pos}(b_j, h) = \text{Pos}(b_i, |G_i| - 1) \wedge \text{Pos}(b_j, h+1) = c\}|$$

Having described the four imputation methods addressed in this chapter, it remains to describe the set social choice functions for complete ballot elections that will be considered. However, no concrete voting rules are analyzed, only families of rules that satisfy different axiomatic properties. It may assist the reader's understanding of this chapter's notation to see some example voting rules expressed using it however, so three are presented briefly below:

- **Plurality**: The plurality winner is the candidate that appears in the highest position on the greatest number of ballots. Formally, the plurality winner is:

$$\text{Plurality}(B) = \operatorname*{argmax}_{c \in C} \sum_{b_j \in B} 1 \text{ iff } \text{Pos}(b_j, 0) = c$$

- **Borda**: A candidate c receives x points from each ballot b_j where $\text{Pos}(b_j, x) = c$. The candidate with the **lowest** point total across all ballots is output as the winner. More formally, the Borda winner of an election with complete ballots B and candidate set C is:

$$\text{Borda}(B) = \operatorname*{argmax}_{c \in C} \sum_{b_j \in B} x \text{ s.t. } \text{Pos}(b_j, x) = c$$

- **Veto**: A candidate c receives 1 point from each ballot b_j where $\text{Pos}(b_j, |C| - 1) \neq c$. The candidate with the **highest** point total across all ballots is output as the winner.

8.3 Axioms for Imputation Methods

It is possible to specify a series of axioms for imputation methods that correspond in some respect to the axioms 1–6 for social choice functions that were given above. These axioms will be useful for characterizing the behaviour of combinations of imputation methods and social choice functions in general terms. Further, these axiomatic properties are more relevant for imputation algorithms than the original set of axioms, since they concern broader properties of the fairness of the imputation, rather than the selection of a specific imputation (i.e. a specific outcome).

Strong Imputation Non-Dictatorship is possessed by an imputation method I if when $|B| \geq 3$ and $|C| \geq 3$, there exists no single ballot $b_j \in B$ such that $\forall a, b \in C$, if b_j ranks $a \succ_j b$, and there exists subsets $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee aL_ib)\}$ such that in $I(B)$, **at least** $\lceil \frac{|X||Y|}{|B|} \rceil + |Y|$ ballots rank $a \succ b$, regardless of the constitution of the remainder of B.

Weak Imputation Non-Dictatorship is possessed by an imputation method I if when $|B| \geq 3$ and $|C| \geq 3$, there exists no single ballot $b_j \in B$ such that $\forall a, b \in C$, if b_j ranks $a \succ_j b$, and there exist subsets $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \in G \wedge (b \notin G_i \vee aL_ib)\}$ then in $I(B)$, **exactly** $|X| + |Y|$ ballots rank $a \succ b$, regardless of the constitution of the remainder of B.

Weak Imputation Non-Imposition is possessed by an imputation method I if for every pair $a, b \in C$, there exists a set of ballots B in which at least one of a and b are in ground set of at least one ballot in B, such that after partitioning $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee aL_ib)\}$, in $I(B)$ **at least** $\lceil \frac{|X||Y|}{|B|} \rceil + |Y|$ ballots rank $a \succ b$.

Strong Imputation Non-Imposition is possessed by an imputation method I if for every pair $a, b \in C$, there exists a set of ballots B in which at least one of a and b are in ground set of at least one ballot in B, such that after partitioning $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee aL_ib)\}$, in $I(B)$ **exactly** $|X| + |Y|$ ballots rank $a \succ b$.

Imputation Monotonicity is possessed by an imputation method I if for every set of ballots B, every pair of candidates $a, b \in C$, and for every ballot $b_j \in B$, if $a \succ_j b$, then if B is partitioned into $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee bL_ia)\}$, let the ballot set $B^* = (B \setminus b_j) \cup b_j^*$ where b_j^* has $b \in G_j^*$ and bL_j^*a or $a \notin G_j^*$, and suppose $I(B)$ ranks $b \succ a$ on $|Y| + z$ ballots for some $0 \leq z \leq |X|$, then $I(B^*)$ ranks $b \succ a$ on **at least** $|Y| + z + 1$ ballots.

Imputation Neutrality[2] is possessed by any imputation method for which permuting the names of the candidates (and thus effectively swapping their positions on ballots) must produce a completion of the ballots that is the corresponding permutation of the imputation method's output. Formally, suppose that for any permutation (i.e. bijective mapping) $\rho : C \to C$, the corresponding permutation of the partial ballots is $\rho(B) = \{\rho(b_j) | b_j \in B\}$, and the permutation of an individual ballot is $\rho(b_j) = \{\rho(L_j), \rho(G_j), C\}$, where $\rho(G_j) = \{\rho(g) | g \in G_j\}$, and $\rho(a)\rho(L_j)\rho(b) \leftrightarrow aL_jb \ \forall a, b \in G_j$. A neutral imputation method I satisfies the property that, for every permutation ρ and set of ballots B, $I(\rho(B)) = \rho(I(B))$.

Imputation I.I.A. is possessed by an imputation method I if the following conditions are satisfied. For all pairs of pairs of candidates $((a, b), (c, d))$ such that $a, b, c \in C$, $a \neq b$, and $d \in C \setminus \{a, b, c\}$, and every set of ballots B, for every ballot $b_j \in B$ that has $a \in G_j$ and $b \notin G_j$ or aL_jb, given ballot set $B^* = (B \setminus b_j) \cup b_j^*$ where b_j^* has $b \in G_j$ and bL_ja, if in $I(B)$ **exactly** z ballots rank $c \succ d$, then in $I(B^*)$ **exactly** z ballots rank $c \succ d$.

Strong Imputation L -Strategy-Proofness is possessed by an imputation method if there exists no set of ballots that contains an L -manipulation with respect to I. A set of ballots B contains an L -manipulation with respect to an imputation method I if there exists some ballot $b_j = \{L_j, G_j, C\} \in B$ and some pair of candidates $a, b \in G_j$ so that aL_jb, and a ballot $b_j^* = \{L_j^*, G_j, C\}$, such that if $I(B)$ contains z ballots ranking $b \succ a$, then $I((B \setminus b_j) \cup b_j^*)$ contains at least $z + 1$ ballots ranking $b \succ a$.

Weak Imputation L -Strategy-Proofness is possessed by an imputation method if there exists no set of ballots that contains an L -manipulation such that for every pair of candidates $(c, d) \in C \times C \setminus (a, b)$, if cL_jd **and** cL_ja, if $I(B)$ contains z ballots ranking $d \succ c$, then $I((B \setminus b_j) \cup b_j^*)$ contains at least z ballots ranking $d \succ c$ also. More informally, a weak L -manipulation is one where a voter can increase the number of ballots that impute b over a, without harming the chances of candidates that the voter likes more than a.

Strong Imputation G -Strategy-Proofness is possessed by an imputation method if there exists no set of ballots which contains a G -manipulation with respect to I. A set of ballots B contains a G -manipulation with respect to an imputation method I if there exists a ballot $b_j = \{L_j, G_j, C\} \in B$ such that for some pair of candidates $a, b \in G_j$ ranked aL_jb and a ballot $b_j^* = \{L_j^*, G_j^*, C\}$, such that if $I(B)$ contains z ballots ranking $b \succ a$, then $I((B \setminus b_j) \cup b_j^*)$ contains at least $z + 1$ ballots ranking $b \succ a$.

Weak Imputation G -Strategy-Proofness is possessed by an imputation method if there exists no set of ballots which contains a G -manipulation such that for every pair

[2]Neutrality [May, 1952] is used instead of universality because voters have already been restricted to casting top-ordered ballots.

of candidates $(c,d) \in C \times C \setminus (a,b)$, if cL_jd **and** cL_ja, then if $I(B)$ contains z ballots ranking $d \succ c$, then $I((B \setminus b_j) \cup b_j^*)$ contains at least z ballots ranking $d \succ c$ also. More informally, a weak G -manipulation is one where a voter can increase the number of ballots that impute b over a, without harming the chances of candidates that the voter likes more than a.

8.3.1 Discussion

Although the axioms defined above are logical analogs of those used by Arrow's theorem and the Gibbard-Satterwaithe theorem, it is worth considering the design question of whether they are also *axioms* (i.e. essential properties) in the context of imputation methods. The motivation behind the design of the original axioms was to specify a minimal set of properties that any sane social choice system should possess. However, imputation models, even in a voting context, need not satisfy the same requirements. The motivation behind the new axioms defined in this chapter are now discussed, along with the implications of an imputation method violating them.

An imputation method that fails to satisfy Weak Non-Dictatorship is likely inappropriate for use in a social choice context, insofar as it can assign massively disproportionate power to the dictator (i.e. the pivotal voter). A method that satisfies Weak Non-Dictatorship, but not Strong Non-Dictatorship is still problematic, but perhaps less so in the event that the assignment is close to the lower bound implied, which is simply proportionate to the rate at which those who express a preference with respect to a and b preferred b to a.

The strong and weak Imputation Non-Imposition also seem necessary. If there is *no* input under which the classifier will adopt the policy of imputing ballots so that a given candidate is at least not *disfavoured*, then the imputation method is clearly unusably biased against that candidate, and cannot be deployed to a social choice setting. Strong imposition at first appears vital, but an imputation policy that adopts a smoothing approach, and always assigns *some* (perhaps infinitely small) set of ballots to a disfavoured outcome is actually not an unreasonable construction, at least for social choice over very large ballot sets.

Imputation Monotonicity may not actually be an essential axiom for imputation methods, but was included for completeness. Reasonable imputation methods might impute ballots based on the similarity of two voters' preferences. If a voter raises a candidate on its ballot, it may become dissimilar to voters with many unspecified preferences, and so have less of an impact on the election. Such methods should not necessarily be ruled out.

However, a sensible similarity-based imputation method that rates incomplete ballot b_i as similar to more complete ballot b_j ought to also impute b_i in a manner consistent with the preferences of b_j. If b_j ranks aL_jb, or $b \notin G_j$ but $a \in G_j$, then it is the case that a similarity-based imputation method ought to impute aL_ib to the extent that it uses information from b_j to determine the imputation of b_i. However, after the positions of a and b are reversed on b_j, either b_j is now dissimilar to b_i, and so no longer affects the imputation (which ought only to improve the chance that bL_ia), or b_j still affects the imputation of b_i, but will now contribute information that should favour bL_ia. This suggests that monotonicity may be satisfied by most imputation methods after all.

Imputation Neutrality is a clearly necessary axiom, insofar as a non-neutral imputation method is, by definition, favouring some candidate based only on its *name*.

Imputation I.I.A. is interestingly, not essential, and perhaps even detrimental, for an imputation method to possess. I.I.A. requires that methods not use information provided by the relative ordering of candidates on voters ballots, a property which is perhaps undesirable. For instance, imagine that voters' preferences are drawn from a domain like the left-right political spectrum. A ballot that lists only two choices, first Conservatives and then Socialists, should probably be imputed with right-leaning parties first, and then left-leaning ones. If the order of the two expressed preferences is swapped however, then the order of the imputed parties ought to switch as well.

Imputation Strategy-Proofness is defined in several parts. The definition used here is necessarily different from that adopted by Reffgen's work [Reffgen, 2011] on strategy-proofness for top-ordered ballots. This is because Reffgen's proof holds for sets of ballots where all voters express the same number of preferences, but the results of this chapter must hold for sets where voters may express any number of preferences up to $|C|$, and may express different numbers of preferences. Two types of strategic behaviours are considered. In the first case, it is assumed that each voter ranks all the alternatives for which they *know* the ordering (i.e. G_j is fixed, but L_j may be misreported). In the second, voters are allowed to misreport both G_j and L_j, provided that the reported L_j is a total ordering of the reported G_j, and no other elements of C. These forms are respectively referred to as L-strategy-proofness and G-strategy-proofness. Both appear to be necessary, but it is possible to imagine imputation algorithms that satisfy one, but not the other.

8.4 Axiomatic Assessment of Imputation Algorithms

This section provides proofs showing which of the Imputation Axioms are satisfied by each of the imputation methods listed in the previous section. This both provides support for

	Non-Dict.	Non-Imp.	Monotonic	Neutral	I.I.A.	L-SP	G-SP
Hot Deck	✗	✓	✓	✓	✓	✓	✓
Plurality	✓	✓	✓	✓	✗	✗	✗
Proportionate	✓	✓	✓	✓	✗	✗	✗
Markov Tree	✓	✓	✗	✓	✗	✗	✗

Table 8.1: A summary of the axiomatic properties that are and are not satisfied by each of several different imputation methods.

the sensibility of the axioms, by showing the circumstances in which they do and do not hold, as well as offering circumstantial evidence for the existence of an impossibility result similar in form to Arrow's Theorem or the Gibbard-Satterthwaite Theorem. The results are summarized in Table 8.1.

8.4.1 Hot Deck Classifier

Theorem 7. *The Hot Deck Classifier does not satisfy Strong or Weak Imputation Non-Dictatorship.*

Proof. By definition of the Hot Deck Classifier, if ballot b_1 ranks any pair $(a, b) \in C \times C$ such that $a \succ b$, then exactly $|X| + |Y|$ ballots will rank $a \succ b$ in the output. Therefore b_1 is a dictator. $\square$

Theorem 8. *The Hot Deck Classifier satisfies both Strong and Weak Imputation Non-Imposition.*

Proof. The proof is by construction. Given a pair of candidates $(a, b) \in C \times C$, and that $|B| \geq 2$, a set of ballots will be constructed where $a \succ_1 b$. The remaining ballots are given by $b_{2,\ldots,|B|} = \{L, C \setminus \{a, b\}, C\}$, where L is any top-order over $C \setminus \{a, b\}$. The Hot Deck Classifier will impute all ballots that do not assign an ordering to a and b to have $a \succ b$, which is sufficient to satisfy both forms of non-imposition. $\square$

Theorem 9. *The Hot Deck Classifier satisfies Imputation Monotonicity*

Proof. Given a set of ballots B, let $b_j \in B$ be a ballot, and $(a, b) \in C \times C$ be any pair of candidates such that $a \succ_j b$. Let $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, $Y = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee aL_ib)\}$ and $Z = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge b \in G_i \wedge (a \notin G_i \vee bL_ia)\}$.

If $b_i = b_1$, then by definition of the Hot Deck Classifier, the total number of ballots ranking $b \succ a$ in $I(B)$ is $|Z|$. If b_i is swapped for $b_i^* = \{L^*, G^*, C\}$ such that $b \in G^*$ and either bL^*a or $a \notin G^*$, then the total number of ballots ranking $b \succ a$ is now $|Z| + |X| + 1$.

If $b_j \neq b_1$, then by definition of the Hot Deck Classifier, the total number of ballots ranking $b \succ a$ in $I(B)$ is either $|Z|$ or $|Z| + |X|$. Note that $b_j \in Y$ by assumption. If b_j is swapped for $b_j^* = \{L^*, G^*, C\}$ such that bL^*a and $a, b \in G^*$ then no change is made in the number of ballots that are imputed with $a \succ b$, by definition of the Hot Deck Classifier, but $b_j^* \in Z$ not Y, so the total number of ballots ranking $b \succ a$ in $I(B)$ is now either $|Z| + |X| + 1$ or $|Z| + 1$, and in either case, 1 greater than before. □

Theorem 10. *The Hot Deck Classifier satisfies Imputation Neutrality.*

Proof. Suppose for contradiction that there exists a set of ballots B, over candidate set C, and a permutation ρ over C, such that $I(\rho(B)) \neq \rho(I(B))$. Denote the imputation of a particular ballot b_i that is part of a set of ballots B by $I(b_i, B)$.

There must therefore exist some ballot b_i such that $I(\rho(b_i), \rho(B)) \neq \rho(I(b_i, B))$, which implies there exists some pair of candidates $(a, b) \in C \times C$ such that $a \succ_i b$ on $I(\rho(b_i), \rho(B))$, but $b \succ_i a$ on $\rho(I(b_i, B))$. Assume without loss of generality that a beats b in ties, and thus also that $\rho(a)$ beats $\rho(b)$ in ties.

If $a \in G_i$ and $b \notin G_i$ or aL_ib, then $I(b_i, b)$ must have aL_ib. Therefore $\rho(I(b_i, B))$ must also rank $\rho(a)$ before $\rho(b)$ by definition. However, $\rho(b_i)$ must either not rank $\rho(b)$, or must rank $\rho(a)L_i\rho(b)$, so $I(\rho(b_i), \rho(B))$ must do so as well. A symmetric argument can be made for $b \in G_i$, so it follows that $a \notin G_i$ and $b \notin G_i$.

Now, consider the behaviour of the Hot Deck Classifier in imputing b_i. If $b_i = b_1$, then since $a \notin G_i$ and $b \notin G_i$, it follows that b_i is imputed with the tie-breaking ordering of a and b, so that $a \succ_i b$. However, under the permutation ρ, b_i does not rank $\rho(a)$ or $\rho(b)$, and is imputed with the tie breaking order, giving that $\rho(a) \succ_i \rho(b)$. Therefore, $b_i \neq b_1$.

If $b_i \neq b_1$, and $a \notin G_i$ and $b \notin G_i$, b_i then if b_1 ranks either a or b, b_i is imputed with $a \succ_i b \leftrightarrow a \succ_1 b$. However, in this case, $\rho(b_1)$ ranks $\rho(a)$ and $\rho(b)$, and $\rho(b_i)$ ranks neither, causing the imputation $\rho(a) \succ_i \rho(b) \leftrightarrow \rho(a) \succ_1 \rho(b)$. So b_1 must not rank a or b. However, in this case, b_1 is imputed with the tie-breaking ordering of a and b prior to the imputation of b_i, and an identical argument can be used to the one presented for the case where b_1 did rank a or b. The assumption that b_i exists is contradicted, so the Hot Deck Classifier must satisfy Imputation Neutrality.

□

Theorem 11. *The Hot Deck Classifier satisfies Imputation I.I.A.*

Proof. Suppose for contradiction that the Hot Deck Classifier does not satisfy Imputation I.I.A. Then there must exist a pair of candidates $(a, b) \in C \times C \setminus a$ and a set of ballots B under which for some ballot $b_j \in I(B)$ and pair of candidates $(c, d) \in C \times (C \setminus \{a, b, c\})$, such that if some ballot $b_i \in B$ ranks $a \succ_i b$, then replacing b_i with a ballot b^* that is identical to b_i, but swaps the order of a and b ($b \succ^* a$), causes $d \succ_j c$ in $I(B \setminus b_i \cup b^*)$, even though $c \succ_j d$ in $I(B)$.

Note if non-adjacent candidates in a top-order are swapped, the relative order of other candidates must necessarily change (e.g. if $a \succ b \succ c$, then swapping to $c \succ b \succ a$ necessarily changes the relative orders of a and b, and of c and b. Therefore b^* represents the top order of b_i with a single adjacent pair swapped. Further, note that b_j must rank neither c nor d.

Under the Hot Deck Classifier, the imputation of b_1 is deterministic, and cannot be affected by changing the order of candidates on any other ballot (it is just the tie-breaking order). Therefore, b_1 cannot be b_j. Additionally, for any ballot other than b_1, the imputation of the ballot is affected only by the ordering present on b_1. Therefore $b_1 = b_i$.

Given any $b_j \neq b_1$, b_j is imputed with $d \succ_j c$ only if $d \succ_1 c$. However, $d \neq b$. Since only the relative ordering of a and b has changed in b_1, therefore $c \succ_i d$ if and only if $c \succ^* d$. It follows that b_j cannot be imputed with $d \succ_j c$.

□

Theorem 12. *The Hot Deck Classifier satisfies both Strong and Weak L-Strategy-Proofness*

Proof. It is necessary to consider only whether b_1 has the opportunity for an L-manipulation, because no other voter's ballot influences the imputation. Suppose for contradiction that there exists a pair of candidates $(a, b) \in C \times C$ such that $a \in G_1$ and $b \in G_1$, and (without loss of generality) aL_1b, but that there exists some ballot $b^* = \{L_1^*, G_1, C\}$ such that in $I(B \setminus b_1 \cup b^*)$ more voters rank $a \succ b$ than in $I(B)$. Since the Hot Deck Classifier satisfies Imputation I.I.A., only changing the relative order of a and b on b^* can alter the fraction of ballots imputed with $a \succ b$. Since the Hot Deck Classifier satisfies Imputation Monotonicity, lowering the position of b relative to a can only decrease the fraction of votes imputed with $a \succ b$. But $a \succ_1 b$ by assumption. Therefore, no Strong L-manipulation exists which implies no Weak L-manipulation exists either. □

Theorem 13. *The Hot Deck Classifier satisfies both Strong and Weak G-Strategy-Proofness*

Proof. It is necessary to consider only whether b_1 has the opportunity for an G-manipulation, because no other voter's ballot influences the imputation. Suppose for contradiction that there exists a pair of candidates $(a, b) \in C \times C$ such that $a \in G_1$ and $b \in G_1$, and (without loss of generality) aL_1b, but that there exists some ballot $b^* = \{L^*, G^*, C\}$ such that in $I(B \setminus b_1 \cup b^*)$ more voters rank $a \succ b$ than in $I(B)$. Since the Hot Deck Classifier satisfies Imputation I.I.A., only changing the relative order of a and b on b^* can alter the fraction of ballots imputed with $a \succ b$. Therefore adding candidates to G_1 cannot change the fraction of ballots imputed with $a \succ b$, nor can removing candidates that are neither a nor b. Removing b does not change its relative position to a. Removing a causes b^* to rank $b \succ^* a$, and since the Hot Deck Classifier satisfies Imputation Monotonicity, lowering the position of b relative to a can only decrease the fraction of voters imputed with $a \succ b$. Therefore, no such G^* exists and no Strong G-manipulation exists which implies no Weak G-manipulation exists either. □

8.4.2 Plurality Classifier

Theorem 14. *The Plurality Classifier satisfies both Strong and Weak Imputation Non-Dictatorship.*

Proof. Suppose for contradiction that $|B| \geq 3$, and that ballot b_1 is the dictator for every pair of candidates (a, b). Suppose that aL_1b, and that a loses ties to b. The set of possible ballot sets must include at least one set where every ballot b_i for $2 \leq i \leq |B| - 1$ consists of identical preferences such that $b_i = \{L^*, C, C\}$, where L^* is any ordering over C such that $\text{Pos}(b_i, |C| - 2) = b$, and $\text{Pos}(b_i, |C| - 1) = a$. The remaining ballot is given by $b_{|B|} = \{L', C \setminus \{a, b\}, C\}$, where L' is L^* with a and b truncated (i.e. $b_{|B|} \models b_i$). Since $|B| > 3$, at least one complete ballot other than b_1 is a prefix of $b_{|B|}$, and ranks bL^*a. As a loses ties to b, the Plurality Classifier will impute $b_{|B|}$ with $bL'a$. Therefore, b_1 is not a weak dictator for a over b. Since 100% of imputed ballots were imputed counter to the expressed preferences of b_1, b_1 is not a strong dictator either. An identical argument holds for every other ballot, and for every other pair of candidates. □

Theorem 15. *The Plurality Classifier satisfies both Strong and Weak non-imposition.*

Proof. The proof is via construction. Given a pair of candidates $(a, b) \in C \times C$, and that $|B| \geq 2$, a set of ballots is constructed where $b_1 ... b_{|B|-1}$ are identical, complete, ballots of the form $b_i = \{L^*, C, C\}$, $\text{Pos}(b_i, |C|-2) = b$, and $\text{Pos}(b_i, |C|-1) = a$. The remaining ballot is given by $b_{|B|} = \{L', C \setminus \{a, b\}, C\}$, where L' is L^* truncated in the last two positions. The plurality classifier will impute all ballots that do not assign an ordering to a and b to have $a \succ b$, which is sufficient to satisfy both forms of non-imposition. □

Theorem 16. *The Plurality Classifier satisfies Imputation Monotonicity.*

Proof. Let B be any set of ballots over a set of candidates C, imputed to the ballot set $I(B)$. Suppose for contradiction that there exists a ballot $b_j \in B$ and a pair of candidates $(a, b) \in C \times C$ such that aL_jb, but that swapping the positions of a and b on b_j to yield the ballot set B^* reduces the total number of ballots that rank a after b in $I(B^*)$.

If for every ballot b_i, at least one of $b \in G_i$ or $a \in G_i$ is true, then no imputation occurs for the relationship between a and b. Adjusting one ballot so that b is now above a must therefore increase the total number that rank b above a. So there must be at least one ballot in the set X of ballots that rank neither a nor b, and this ballot must be imputed such that $a \succ b$ when it is imputed alongside B, but $b \succ a$ when imputed alongside B^*. Call this ballot b_{pivot}.

Suppose that b_{pivot} ranks no elements. Then the plurality classifier imputes it with the most common first preference among the other votes. The most common first preference in B cannot be a, or b_{pivot}'s imputation would rank a above b. If it is b under B, then it must change to a under B^*. However, since b_j has increased the position of b relative to a, b cannot be a less common first preference among voters in B^* than it is among voters in B. Therefore b_{pivot} must not rank a or b as its first preference.

Suppose that b_{pivot} has $|G_{\text{pivot}}| = k$ elements with defined ranks, either specified by the voter or imputed by the plurality classifier thus far, and that neither a nor b has yet been ranked.

When deciding which candidate to assign to the next rank, the plurality classifier will impute the most common $k + 1^{\text{th}}$ preference among ballots that are consistent extensions of b_{pivot}. If b_j is not consistent extension of b_{pivot}, then swapping the positions of a and b on b_j cannot make it a consistent extension either, and so the plurality classifier must make the same decision for the imputation of b_{pivot} under both B and B^* here. Therefore, b_j is a consistent extension of b_{pivot}.

Since b_j is a consistent extension of b_{pivot}, and b_{pivot} ranks neither a nor b, b_j does not rank either a or b before position k.

If the $k + 1^{th}$ position of b_j is neither a nor b, then swapping a and b to produce b_j^* cannot alter the imputation decision for the $k + 1^{th}$ preference of b_{pivot}. Therefore the $k + 1^{th}$ position of b_j must be either a or b.

The $k+1^{th}$ position of b_j cannot be b by assumption that a precedes b on b_j. Therefore it is a.

If the $k+1^{th}$ position of b_j is a, then the same position is b in b_j^*. However, this means that more ballots which are consistent extensions of b_{pivot} rank b in the $k+1^{th}$ position under B^* than under B, and fewer rank a in that position, so the plurality classifier cannot impute with b in B and a in B^*. This is a contradiction. □

Theorem 17. *The Plurality Classifier satisfies Imputation Neutrality.*

Proof. Suppose for contradiction that there exists a set of ballots B, over candidate set C, and a permutation of C ρ, such that $I(\rho(B)) \neq \rho(I(B))$. The imputation of a particular ballot b_i that is part of a set of ballots B is denoted by $I(b_i, B)$.

There must therefore exist some ballot b_i such $I(\rho(b_i), \rho(B)) \neq \rho(I(b_i, B))$, which implies there exists some pair of candidates $(a, b) \in C \times C$ such that $a \succ_i b$ on $I(\rho(b_i), \rho(B))$, but $b \succ_i a$ on $\rho(I(b_i, B))$.

If $a \in G_i$ and $b \notin G_i$ or aL_ib, then $I(b_i, b)$ must have aL_ib. Therefore $\rho(I(b_i, B))$ must also rank $\rho(a)$ before $\rho(b)$ by definition. However, $\rho(b_i)$ must either not rank $\rho(b)$, or must rank $\rho(a)L_i\rho(b)$, so $I(\rho(b_i), \rho(B))$ must do so as well. A symmetric argument can be made for $b \in G_i$, so it follows that $a \notin G_i$ and $b \notin G_i$.

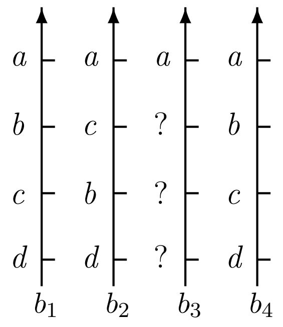
Now, consider the behaviour of the plurality classifier in imputing b_i. Suppose that b_i ranks no candidates at all. The plurality classifier will impute the most common first preference in B, which will be labelled $c_{\max}$. However, when imputing $\rho(b_i)$ in conjunction with $\rho(B)$, the plurality classifier will impute the most common first preference in $\rho(B)$, which by definition must be $\rho(c_{\max})$. Therefore $\text{Pos}(I(\rho(b_i), \rho(B)), 1) = \rho(\text{Pos}(I(b_i, B), 1))$

Inductively, suppose that for all $0 < j < k$, $\text{Pos}(I(\rho(b_i), \rho(B)), j) = \rho(\text{Pos}(I(b_i, B), j))$. The plurality classifier will impute $\text{Pos}(I(\rho(b_i), \rho(B)), k)$ with the most common k^{th} preference among the set of ballots X for which the current partial imputation of b_i is a prefix. Call this candidate $c_{\max}$. Via the inductive hypothesis, $\rho(X)$ is the set of ballots for which the partial imputation of $\rho(b_i)$ is a prefix. Therefore, the most common k^{th} preference among $\rho(X)$ is $\rho(c_{\max})$. So $\text{Pos}(I(\rho(b_i), \rho(B)), k) = \rho(\text{Pos}(I(b_i, B), k))$. This contradicts the initial assumption. □

Theorem 18. *The Plurality Classifier does not satisfy Imputation I.I.A.*

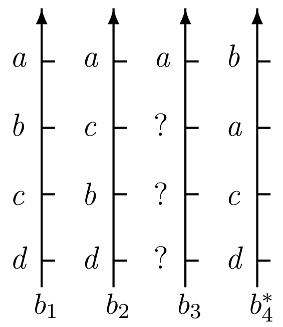
Proof. The proof is via counter-example.

Suppose an election is held over 4 candidates $C = \{a, b, c, d\}$, and (without loss of generality) that c precedes b in the tie-breaking ordering, and that there are 4 voters, with preferences:



Ballot b_3 is a prefix of all other ballots. The most common second preference is b, so b_3's second preference will be b. It is then a prefix of only ballots b_1 and b_4, so its final imputation is as a copy of b_1.

Suppose that ballot b_4 swaps the locations of a and b to yield the ballots:



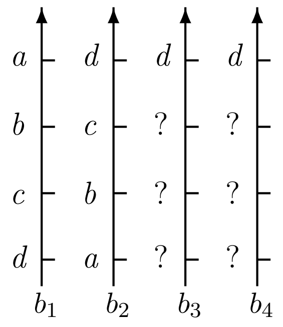
Now b_3 is a prefix of only b_1 and b_2. Both b and c are tied for most common second preference in this set, but c wins ties, and the second preference c is imputed. After this step, b_3 will only be a prefix of b_2, and so the final result is that a copy of b_2 is imputed. Therefore, swapping the positions of candidates a and b on b_4 has caused the imputed ordering of b and c to be reversed on b_3. Since b_3 is the only ballot that is imputed, a different number of ballots rank $b \succ c$ in the imputation with b_4 than in the imputation with b_4^*, and the plurality classifier does not satisfy imputation I.I.A.

□

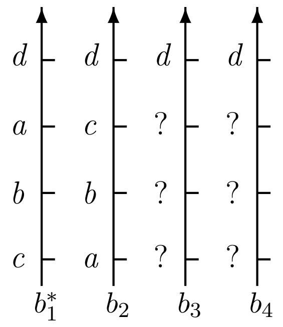
Theorem 19. *The Plurality Classifier does not satisfy Strong or Weak L-Strategy-Proofness.*

Proof. The proof is via counter example.

Suppose that there exists an election over 4 candidates $C = \{a, b, c, d\}$, and (without loss of generality) that a precedes c in the tie-breaking ordering, and that there are 4 voters, with preferences:



If b_1 reports their preferences truthfully, then b_3 and b_4 will both be imputed with b_2's preferences, which notably rank a after b. However, if b_1 misreports with the ballot b_1^*:



then, as a beats c in the tie-breaking ordering, both b_3 and b_4 will be imputed with b_1^*'s ordering, which ranks b after a. Therefore, $a \succ b$ once in the profile with b_1, and thrice in the profile with b_1^*. Since b_1 ranks no candidates above a, Weak-Imputation strategy-proofness is not satisfied, which implies the Strong version is not satisfied either[3].

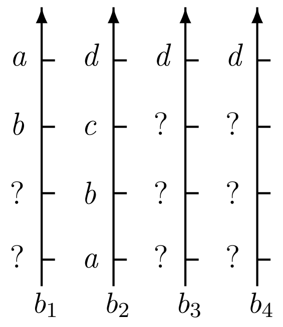
□

Theorem 20. *The Plurality Classifier does not satisfy Strong or Weak Imputation G-Strategy-Proofness.*

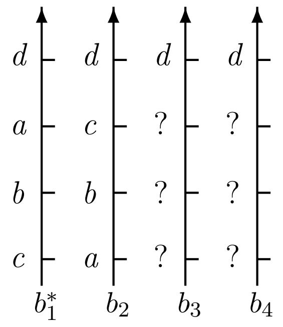
Proof. The proof is via counter example.

Assume an election over 4 candidates $C = \{a, b, c, d\}$, and (without loss of generality) that a precedes c in the tie-breaking ordering, and that there are 4 voters, with preferences:

[3]Although this example may not seem especially useful, consider the case where these ballots are a subset of a close election between b and a. b_1 helps their most preferred candidate a most in this contest by ranking a second, as shown.



If b_1 reports their preferences truthfully, then b_3 and b_4 will both be imputed with b_2's preferences, which notably rank a after b. However, it b_1 misreports with the ballot b_1^*:



then, as a beats c in the tie-breaking ordering, both b_3 and b_4 will be imputed with b_1^*'s ordering, which ranks b after a. Therefore, $a \succ b$ once in the profile with b_1, and thrice in the profile with b_1^*. Since b_1 ranks no candidates above a, no candidate that b_1 prefers to a is harmed, and Weak-Imputation G-strategy-proofness is not satisfied, which implies the Strong version is not satisfied either[4]. □

8.4.3 The Proportionate Classifier

Theorem 21. *The Proportionate Classifier satisfies Strong and Weak Imputation Non-Dictatorship*

Proof. Suppose for contradiction and without loss of generality that there exists a finite set of ballots B, $|B| \geq 3$, and $b_1 \in B$ is a dictator such that for every pair $(a, b) \in C \times C$, if $a \succ_1 b$, then, if $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i =$

[4]Although this example may not seem especially useful, consider the case where these ballots are a subset of a close election between b and a. b_1 helps their most preferred candidate a most in this contest by ranking a second, as shown.

$\{L_i, G_i, C\} | b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee aL_ib)\}$ at least a fraction $\lceil |X| \frac{|Y|}{|B|} \rceil + |Y|$ of ballots rank $a \succ b$ in $I(B)$.

Define the remainder of B such that for all $2 \leq i \leq |B| - 1$, $b_i = \{L^*, C, C\}$, and $b_{|B|} = \{L', C \setminus \{a, b\}, C\}$ where L' is L^* with a and b removed, and L^* is any ranking ending in $b \succ a$, such that $b_1 \not\models b_{|B|}$. Since $b_2 ... b_{|B|-1} \models b_{|B|}$, the proportionate classifier will impute b_j such that $b \succ a$. Therefore, only $|Y|$ ballots in $I(B)$ rank $a \succ b$, and both Weak and Strong Imputation Non-Dictatorship are satisfied. □

Theorem 22. *The Proportionate Classifier satisfies both Strong and Weak Imputation Non-Imposition.*

Proof. The proof is by construction. For any two candidates $(a, b) \in C \times C$, let B be a set of ballots over C such that in b_1, aL_1b, but for every other ballot $b_i \neq b_1$, $b_i = \{L_i, \emptyset, C\}$. By definition $b_1 \models b_i$ for all i, and b_i is not a prefix of any other ballot in B. Therefore, the proportionate classifier will impute aL_ib in every b_i. □

Theorem 23. *The Proportionate Classifier satisfies Imputation Monontonicity.*

Proof. Let B be a set of ballots, and $(a, b) \in C \times C$ be a pair of candidates. Let $b_j = \{L_j, G_j, C\}$ such that aL_jb or $b \notin G_j$, and let $b^* = \{L^*, G^*, C\}$, such that for any pair of candidates $(c, d) \in (C \times C) \setminus (a, b)$, $cL_jd \leftrightarrow cL^*d$, and for any candidate $c \in C \setminus a$, $c \in G_j \leftrightarrow c \in G^*$, and $b \in G^*$, and either $bL^*a \wedge a \in G^*$ or $a \notin G^*$ (that is, let b^* be b_j with the relationship between a and b reversed). Assume for contradiction that $I(B)$ contains fewer ballots that rank $a \succ b$ than $I(B \setminus b_j \cup b^*)$ does.

Note that b_j only has an impact on the imputation of ballots b_i such that $b_j \models b_i$, by definition of the proportionate classifier. However, if b_i ranks either a or b, then by definition of a top order, b_j cannot influence the order of a and b on b_i. Therefore, there must exist some ballot b_i such that $b_j \models b_i$ and $a \notin G_i \wedge b \notin G_i$. Note that since b_i does not rank a or b, switching the relative order of a and b cannot change whether or not b_j has an impact on the imputation of b_i. Further, not that b_i will be imputed identically under B or $B \setminus b_j \cup b^*$ up to position $k = \min_k \text{Pos}(b_j, k) = a \wedge \text{Pos}(b^*, k) = b$.

When the imputation decision is made at position k, since there is one fewer ballot ranking a at position k, the proportion of ballots the Proportionate Classifier assigns to a cannot increase, it can only decrease or remain unchanged, which contradicts the assumption that the Proportionate Classifier satisfies Imputation Monotonicity. □

Theorem 24. *The Proportionate Classifier satisfies Imputation Neutrality.*

Proof. Suppose for contradiction that there exists a set of ballots B, over candidate set C, and a permutation over C denoted ρ, such that $I(\rho(B)) \neq \rho(I(B))$. Denote the imputation of a particular ballot b_i that is part of a set of ballots B by $I(b_i, B)$.

There must therefore exist some ballot b_i such $I(\rho(b_i), \rho(B)) \neq \rho(I(b_i, B))$, which implies there exits some pair of candidates $(a, b) \in C \times C$ such that $a \succ_i b$ on $I(\rho(b_i), \rho(B))$, but $b \succ_i a$ on $\rho(I(b_i, B))$.

If $a \in G_i$ and $b \notin G_i$ or aL_ib, then $I(b_i, b)$ must have aL_ib. Therefore $\rho(I(b_i, B))$ must also rank $\rho(a)$ before $\rho(b)$ by definition. However, $\rho(b_i)$ must either not rank $\rho(b)$, or must rank $\rho(a)L_i\rho(b)$, so $I(\rho(b_i), \rho(B))$ must do so as well. A symmetric argument can be made for $b \in G_i$, so it follows that $a \notin G_i$ and $b \notin G_i$.

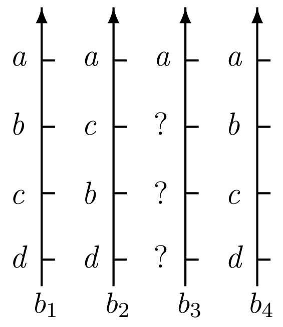
Now, consider the behaviour of the Proportionate Classifier in imputing b_i. Suppose that under $I(b_i, B)$, a is the first candidate that was imputed into b_i. There exists a set of ballots $B^* \subset B$, such that for every $b_j \in B^*$, $b_j \models b_i$. There may be z copies of b_i in B. Of these, by definition of the Proportionate Classifier, exactly n_l are assigned to a for some $0 \leq l \leq |C| - 1$. n_l is determined by the proportion of B^* that rank the candidate that is l^{th} in the tie-breaking order at position $|G_i| + 1$, as well as the value of l. Note that if n_l votes are assigned to a under $I(b_i, B)$, then under $I(\rho(b_i, B))$, n_l votes are assigned to $\rho(a)$. This is because the candidate at location l in the tie-breaking order under ρ is by definition $\rho(a)$, $\rho(B^*)$ is precisely the set of ballots that will be used to impute $\rho(b_i)$, and the same number of members of $\rho(B^*)$ rank $\rho(a)$ at position $|G_i| + 1$ as members of B^* ranked a there. Therefore $\rho(a)$ must be imputed ahead of $\rho(b)$. It follows that a cannot have been the first candidate imputed on b_i. Since $a \succ_i b$ under $I(b_i, B)$ by assumption, b has not yet been imputed either.

However, suppose that neither a nor b have been imputed on b_i after $1 \leq k$ positions have been imputed. Suppose that a is to be imputed at position $k + 1$ on b_i. By precisely the same argument used above, neither a nor b can be imputed at position $k+1$ (the set B^* will differ, but the form of the argument is identical). This means a can never be imputed on b_i, which is a contradiction. $\square$

Theorem 25. *The Proportionate Classifier does not satisfy imputation I.I.A.*

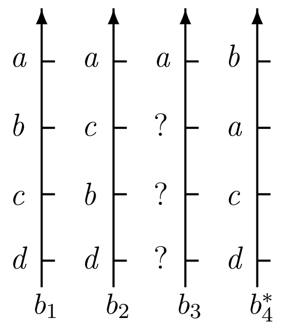
Proof. The proof is via counter-example, and very similar to the form used for the Plurality Classifier, modulo some details concerning the behaviour of the Proportionate Classifier in the event of a tie.

Consider an election over 4 candidates $C = \{a, b, c, d\}$, suppose (without loss of generality) that c precedes b in the tie-breaking ordering, and that there are 4 voters, with preferences:



Ballot b_3 is a prefix of all other ballots. The most common second preference is b, so b_3's second preference will be b. It is then a prefix of only ballots b_1 and b_4, so it's final imputation is as a copy of b_1.

Suppose that ballot b_4 swaps the locations of a and b to yield the ballots:



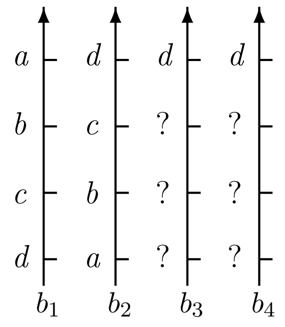
Now b_3 is a prefix of only b_1 and b_2. Both b and c are tied for most common second preference in this set, but c wins ties. This means that n_l for c is computed before n_l for b. Since there is only one ballot to impute, n_l will assign the ceiling of the ratio of the number of ballots that impute c here to all ballots that $\models b_3$. This ratio is 0.5, so the single ballot is assigned second preference c. After this step, b_3 will only be a prefix of b_2, and so the final result is that a copy of b_2 is imputed. Therefore, swapping the positions of candidates a and b on b_4 has caused the imputed ordering b and c to be reversed on b_3. Since b_3 is the only ballot that is imputed, a different number of ballots rank $b \succ c$ in the imputation with b_4 than in the imputation with b_4^*, and the Proportionate Classifier does not satisfy imputation I.I.A.

□

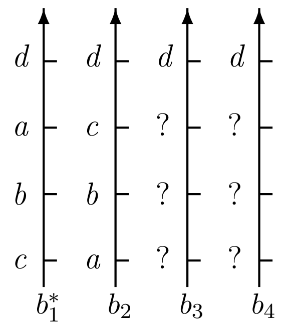
Theorem 26. *The Proportionate Classifier does not satisfy Strong or Weak L-Strategy-Proofness.*

Proof. The proof is via counter example, and again, is very similar in form to the corresponding proof for the Plurality Classifier.

Suppose that there exists an election over 4 candidates $C = \{a, b, c, d\}$, and that there are 4 voters, with preferences:



If b_1 reports their preferences truthfully, then b_3 and b_4 will both be imputed with b_2's preferences, which notably rank a after b, because only $b_2 \models \{b_3, b_4\}$. However, if b_1 misreports with the ballot b_1^*:



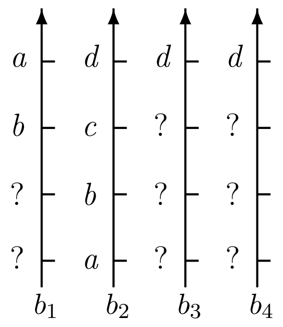
then one of b_3 and b_4 will be imputed will be imputed with b_1^*'s ordering, and the other with b_2's ordering. Since b_1^*'s ordering ranks b after a, $a \succ b$ once in the profile with b_1, and twice in the profile with b_1^*. Since b_1 ranks no candidates above a, Weak-Imputation Strategy-Proofness is not satisfied, which implies the Strong version is also not satisfied.

□

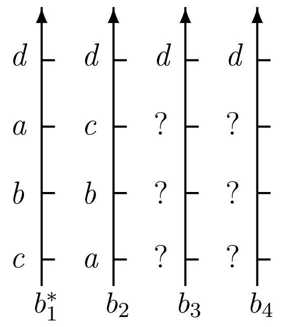
Theorem 27. *The Proportionate Classifier does not satisfy Strong or Weak Imputation G-Strategy-Proofness.*

Proof. The proof is via counter example, again just a slight variant on the proof for the Plurality Classifier.

Assume an election over 4 candidates $C = \{a, b, c, d\}$, and that there are 4 voters, with preferences:



If b_1 reports their preferences truthfully, then b_3 and b_4 will both be imputed with b_2's preferences, which notably rank a after b. However, if b_1 misreports with the ballot b_1^*:



then, as in the previous proof, one of b_3 or b_4 will be imputed with the preferences of b_1^*, which ranks b after a. Therefore, $a \succ b$ once in the profile with b_1, and twice in the profile with b_1^*. Since b_1 ranks no candidates above a, no candidate that b_1 prefers to a is harmed, and Weak-Imputation G-strategy-proofness is not satisfied, which implies the Strong version is not satisfied either. □

8.4.4 First Order Markov Tree

Theorem 28. *The First Order Markov Tree Classifier satisfies Strong and Weak Imputation Non-Dictatorship.*

Proof. Suppose for contradiction and without loss of generality that there exists a finite set of ballots B, $|B| \geq 3$, and that $b_1 \in B$ is a dictator such that for every pair $(a, b) \in C \times C$, if $a \succ_1 b$, then, if $X = \{b_i = \{L_i, G_i, C\} | b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, and $Y = \{b_i =$

$\{L_i, G_i, C\}|b_i \in B \wedge a \in G_i \wedge (b \notin G_i \vee aL_ib)\}$ at least a fraction $\lceil |X|\frac{|Y|}{|B|}\rceil + |Y|$ of ballots rank $a \succ b$ in $I(B)$. Assume without loss of generality that a beats b in the tie-breaking order.

Define $B \setminus b_1$ such that for all $2 \leq i \leq |B| - 1$, $b_i = \{L^*, C, C\}$, and $b_{|B|} = \{L', C \setminus \{a, b\}, C\}$ where L' is L^* with a and b removed, and L^* is structured so that $a \succ b$. For all $b_2 ... b_{|B|-1}$, $\text{Pos}(b_i, |G_{|B|}| - 1) = \text{Pos}(b_{|B|}, |G_{|B|} - 1|$ and $\text{Pos}(b_i, |G_{|B|}|) = a$, so there is at least one ballot in B that counts toward imputing $b_{|B|}$ with $a \succ b$. Since there is a most one ballot that counts toward imputing $b_{|B|}$ with $b \succ a$ (i.e. b_1), and a wins ties over b, the first order Markov Tree will impute $a \succ b$ on $|X|$ ballots. Therefore b_1 is not a Strong or Weak dictator for a over b. □

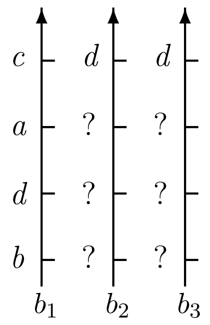
Theorem 29. *The First Order Markov Tree Classifier satisfies both Strong and Weak Imputation Non-Imposition.*

Proof. The proof is via construction. Given an arbitrary pair of candidates $(a, b) \in C \times C$ such that $a \neq b$, the First Order Markov Tree Classifier is non-imposing if there exists at least one set of ballots B such that $X = \{b_i = \{L_i, G_i, C\}|b_i \in B \wedge a \notin G_i \wedge b \notin G_i\}$, $|X| > 1$ and $\exists b_j \in B$ such that $a \in G_j \vee b \in G_j$. Such a set consists of a ballot b_j ranking only $a \succ_j b$, and some arbitrary number of empty ballots. □

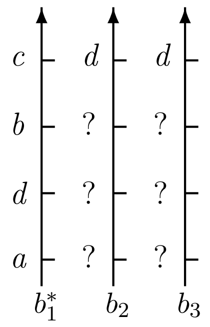
Theorem 30. *The First Order Markov Tree Classifier does not satisfy Imputation Monotonicity.*

Proof. The proof is by counterexample.

Assume an election over 4 candidates $C = \{a, b, c, d\}$, and that there are 3 voters, with preferences:



Under these preferences, b_2 and b_3 will both be imputed with $d \succ b \succ a \succ c$, or $d \succ b \succ c \succ a$, depending on the tie-breaking ordering for a and c. Note that both orders rank b ahead of a. However, if b_1 raises the position of b on its ballot resulting in b_1^*:



then b_2 and b_3 will be imputed with either $d \succ a \succ c \succ b$ or $d \succ a \succ b \succ c$, depending on the tie-breaking ordering for b and c. Note that both orderings rank a ahead of b. Therefore, swapping the positions of candidates a and b on b_1 caused 2 fewer ballots in the resulting imputed preferences to rank b ahead of a, and the First Order Markov Tree Classifier does not satisfy Imputation Monotonicity.

□

Theorem 31. *The First Order Markov Tree Classifier satisfies Imputation Neutrality.*

Proof. Suppose for contradiction that there exists a set of ballots B, over candidate set C, and a permutation of C called ρ, such that $I(\rho(B)) \neq \rho(I(B))$. Denote the imputation of a particular ballot b_i that is part of a set of ballots B by $I(b_i, B)$.

There must therefore exist some ballot b_i such $I(\rho(b_i), \rho(B)) \neq \rho(I(b_i, B))$, which implies there exits some pair of candidates $(a, b) \in C \times C$ such that $a \succ_i b$ on $I(\rho(b_i), \rho(B))$, but $b \succ_i a$ on $\rho(I(b_i, B))$.

If $a \in G_i$ and $b \notin G_i$ or aL_ib, then $I(b_i, b)$ must have aL_ib. Therefore $\rho(I(b_i, B))$ must also rank $\rho(a)$ before $\rho(b)$ by definition. However, $\rho(b_i)$ must either not rank $\rho(b)$, or must rank $\rho(a)L_i\rho(b)$, so $I(\rho(b_i), \rho(B))$ must do so as well. A symmetric argument can be made for $b \in G_i$, so it follows that $a \notin G_i$ and $b \notin G_i$.

Now, consider the behaviour of the First Order Markov Tree Classifier in imputing b_i. Suppose that under $I(b_i, B)$, a is the first candidate that was imputed into b_i, and that b_i ranks no candidates at all. Then by definition of the First Order Markov Tree Classifier, it must be the case that a is the most common first preference among ballots in B. If a is the most common first preference in B (or wins a tie for most common), then $\rho(a)$ is by definition the most common first preference in $\rho(B)$ (or wins a tie for the most common), meaning that $\rho(a)$ will be imputed ahead of $\rho(b)$. Therefore a cannot be imputed first on an empty ballot.

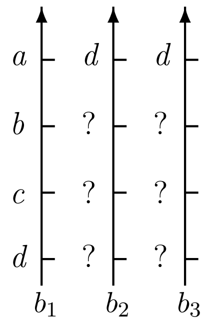
Suppose that a is imputed first on a ballot b_i that is not empty, and, without loss of generality, that $\text{Pos}(b_i, |G_i| - 1) = c$. It follows that in B, a must be the most common (or win a tie among the most common) candidates that immediately follow c and are not in G_i. By definition then, in $\rho(B)$, $\rho(a)$ must be the most common (or win a tie among the most common) candidates that immediately follow $\rho(c)$ and are not in $\rho(G_i)$. Therefore $\rho(a)$ is imputed ahead of $\rho(b)$.

Suppose that a is imputed after k other candidates have been imputed on b_i. This is treated identically to if a were imputed first on a ballot that ranked k additional candidates in the imputed order. Therefore, the argument above applies equally well, and a cannot be imputed at any location on b_i, which is a contradiction. Therefore, the First Order Markov Tree satisfies Imputation Neutrality. □

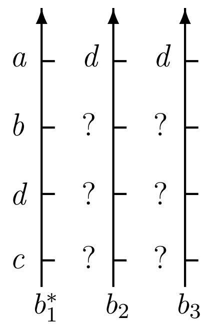
Theorem 32. *The First Order Markov Tree Classifier does not satisfy Imputation I.I.A.*

Proof. The proof is via counterexample.

Assume an election over 4 candidates $C = \{a, b, c, d\}$, and that there are 3 voters. Without loss of generality, assume a tie-breaking order such that $a \succ b \succ c \succ d$. Voters have initial preferences:



Under these preferences, b_2 and b_3 will both be imputed $d \succ a \succ b \succ c$. Note that both imputed ballots therefore rank c after a. However, if b_1 swaps the positions of d and c on its ballot resulting in b_1^*:

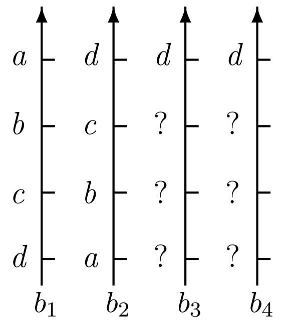


then b_2 and b_3 will be imputed with either $d \succ c \succ a \succ b$ instead, which ranks c above a. Therefore, swapping the order of candidates c and d on b_1 caused 2 fewer ballots in the resulting imputed preferences to rank a ahead of c, and the First Order Markov Tree Classifier does not satisfy Imputation I.I.A. $\square$

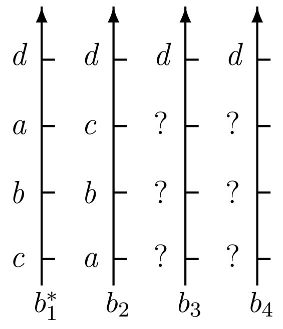
Theorem 33. *The First Order Markov Tree Classifier does not satisfy Strong or Weak Imputation L-Strategy-Proofness.*

Proof. The proof is via counter example, and again, is very similar in form to the corresponding proof for the Plurality Classifier.

Suppose that there exists an election over 4 candidates $C = \{a, b, c, d\}$, with tie-breaking order $a \succ b \succ c \succ d$, and that there are 4 voters, with preferences:



If b_1 reports their preferences truthfully, then b_3 and b_4 will both be imputed with b_2's preferences, which notably rank a after b. This is because the only candidate to follow d on any ballot is c. There is a tie for candidates that follow c between b and d, but since d is already in the ground set of b_3 and b_4, b is imputed. However, if b_1 misreports with the ballot b_1^*:



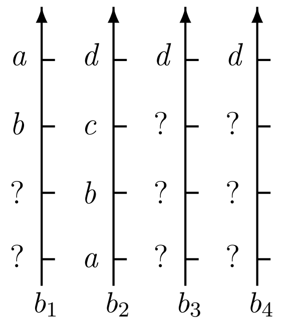
then both b_3 and b_4 will be imputed will be imputed with b_1^*'s ordering. Since b_1^*'s ordering ranks b after a, $a \succ b$ once in the profile with b_1, and thrice in the profile with

b_1^*. Since b_1 ranks no candidates above a, Weak Imputation L-Strategy-Proofness is not satisfied, which implies the Strong version is also not satisfied. □

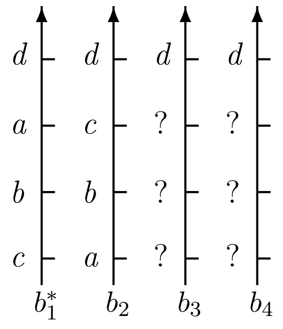
Theorem 34. *The First Order Markov Tree Classifier does not satisfy Strong or Weak G-Strategy-Proofness.*

Proof. The proof is via counter example, and again, is very similar in form to the corresponding proof for the Plurality Classifier.

Suppose that there exists an election over 4 candidates $C = \{a, b, c, d\}$, with tie-breaking order $a \succ b \succ c \succ d$, and that there are 4 voters, with preferences:



If b_1 reports their preferences truthfully, then b_3 and b_4 will both be imputed with b_2's preferences, which notably rank a after b. This is because the only candidate to follow d on any ballot is c, and the only candidate to follow c is b. However, if b_1 misreports with the ballot b_1^*:



then both b_3 and b_4 will be imputed will be imputed with b_1^*'s ordering. Since b_1^*'s ordering ranks b after a, $a \succ b$ once in the profile with b_1, and thrice in the profile with b_1^*. Since b_1 ranks no candidates above a, Weak Imputation G-Strategy-Proofness is not satisfied, which implies the Strong version is also not satisfied. □

8.4.5 Discussion

The proofs in the preceding section show the versatility of proposed set of axioms, and demonstrate how they could be used to describe and compare the properties of different algorithms used as part of the imputation-based approach to social choice. It is interesting to note that of the systems considered, only the dictatorial Hot Deck Classifier is strategy-proof. It is also the only method to satisfy the I.I.A. criterion. Similarly, it is perhaps surprising that the First Order Markov Tree Classifier does not satisfy Imputation Monotonicity. Although not shown in detail here, it appears that the monotonicity of the technique is tightly related to the order of the tree. For instance, it is possible to show that the swapping of single pair of adjacent candidates in a voter's ballot will monotonically improve the fraction of votes imputed with the new ordering of the candidates, even though swaps over larger distances (or multiple swaps) may decrease it instead. The Markov Tree also provides a reasonable argument against the current Imputation Monotonicity axiom, or perhaps against monotonicity entirely in the imputation context. The Markov Tree Classifier is not monotonic because it tries to impute ballots with the completions used by *similar* voters. Raising a candidate in one's ranking might make one very dissimilar to other voters, and so an imputation method should perhaps avoid using the altered ranking to impute votes.

It seems probable that variants of Arrow's Theorem and the Gibbard-Satterthwaite Theorem could be proven for these sets of axioms, but such proofs remain open questions at this time.

8.5 Combinations

The final results presented in this section concern the combination of voting rules and imputation systems, and the extent to which reasonable axiomatic properties are preserved or harmed following their combinations. A preliminary study of the problem was conducted, which shows that there is much interesting work to be done in the future.

The interesting property of combining voting rules [Walsh and Xia, 2012; Nina Narodytska et al., 2012] is that combined rules can be both better (i.e. fairer, more strategy-proof) or worse (i.e. less fair, more vulnerable to manipulation). The first result below demonstrates that combining a non-dictatorial voting rule with a dictatorial imputation method does not produce a dictatorial voting rule overall.

Theorem 35. *If S is a non-dictatorial voting rule and I is an imputation algorithm,* *than $S \oplus I$ is a non-dictatorial voting rule, even if I does not satisfy Imputation Non-Dictatorship.*

Proof. The proof is by contradiction. Suppose without loss of generality that $b_1 \in B$ is a dictator for every pair of candidates $(a, b) \in C \times C$ such that $a \neq b$, under voting rule $S \oplus I$. Then by definition it must be the case that under $S(I(B))$, $a \succ b$ if aL_ib, regardless of the composition of the remainder of B.

By assumption, S is non-dictatorial. Therefore, there exists some profile B^* under which b_1 is not pivotal for $a \succ b$ under $S(B^*)$. Since S is defined to operate only over complete ballots, B^* must be complete. However, then $I(B^*) = B^*$, and therefore $S(I(B^*)) = S(B^*)$, so since b_1 is not a dictator for (a, b) under $S(B^*)$, b_1 is not a dictator for (a, b) under $S \oplus I$ either. □

In contrast, the second result shows that if the imputation method is manipulable, then the combined rule is manipulable, even if the voting rule is strategy-proof. This proof relies on the concept of Anonymity for an imputation method. Anonymity is analogous to Neutrality, but for the names of the voters rather than the names of the candidates. It says that if the names of the voters are permuted, but the set of ballots is otherwise unchanged, the behaviour of the imputation algorithm should not change.

Theorem 36. *If I does not satisfy Imputation G-Strategy-Proofness, but satisfies Imputation Neutrality and is Anonymous then $S \oplus I$ is not a strategy-proof voting rule if S is Non-Imposing.*

Proof. Suppose for contradiction that $S \oplus I$ is strategy-proof.

If S is not strategy-proof, then there must exist some complete set of ballots B such that $S(B)$ returns outcome o_1, but under $B^* = B \setminus b_j \cup b_j^*$, $S(B^*)$ returns some other outcome o_2 such that $o_2 L_j o_1$. Since B^* is complete, $S(I(B)) \neq S(I(B^*))$ as well, and $S \oplus I$ is not strategy-proof, regardless of the behaviour of I. Therefore, S must be strategy-proof.

If S is strategy-proof, it must violate one of Non-Dictatorship or Non-Imposition, by the Gibbard-Saterthwaite Theorem (since S is resolute and deterministic by the definition of a voting rule used here). S is Non-Imposing by assumption. Suppose that S is dictatorial, and that b_1 is the dictator (without loss of generality). By definition of Imputation L-Strategy-Proofness, there exists a profile B where for some pair of candidates $(a, b) \in C \times C$, $a \neq b$, $I(B)$ ranks $a \succ b$ on z ballots, but for some $B^* = B \setminus b_j \cup b_j^*$ such that aL_jb, $I(B^*)$ ranks $a \succ b$ on at least $z + 1$ ballots. Therefore, at least two ballots were imputed with $a \succ b$ under B^* that were imputed with $b \succ a$ under B. Without loss of generality, let one of these be b_1 (The rankings can be permuted because I is anonymous). Assume without loss of generality that b is ranked first on b_1 after imputation under B (the candidates can be permuted because I is neutral). Then, the first ranked candidate under $S(I(B))$ is b, but b is ranked after a under $S(I(B^*))$. Therefore S does not satisfy Non-Dictatorship. This is a contradiction, so $S \oplus I$ must not be Strategy-Proof. □

8.6 Summary

This chapter provided a reasonable set of axioms for use in analyzing and comparing the performance of competing imputation methods, as part of the imputation-based approach to social choice with incomplete information. The axioms are logical extensions of the canonical axioms used in the analysis of voting systems, but are tailored to the imputation domain. An analysis was performed over four simple imputation methods. It was also demonstrated that combinations of voting rules and imputation methods may exhibit properties of either method.

Overall, the proposed axioms and corresponding analysis offer a compelling avenue of future study integrating machine learning and social choice. This area shows great promise in the further development of the imputation-based approach, providing strong theoretical characterizations of competing algorithms, and providing insights into which learning algorithms might work most effectively with which voting rules.

Chapter 9

Manipulation Analysis

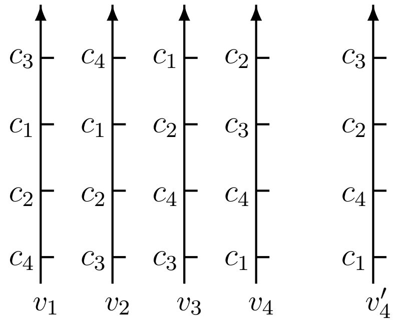
DON'T MAKE A STATEMENT, MAKE A DIFFERENCE!

strategicvoting.ca, site motto

Chapter 4 showed that in general, imputation-based voting systems are subject to the Gibbard-Satterwaith theorem [Gibbard, 1973; Satterthwaite, 1975], while Chapter 8 explored this in greater detail, with an expanded set of axioms that captured the spirit of the Gibbard-Satterwaith theorem for imputation algorithms. These results suggest that the imputation-based approach to social choice incentivizes users to misreport their preferences in order to achieve more desirable outcomes. Since the imputation algorithms themselves can be directly influenced by manipulations of this kind, the problem is perhaps even more concerning than with conventional social choice algorithms, since a manipulation can cause other voters' ballots to be imputed incorrectly as well. This chapter considers different strategies for mitigating the impact of manipulation on imputation-based approaches to social choice, with a focus on approaches to reduce the expected benefits of misreporting one's preferences in the first place. An existing approach from the literature is integrated into the Prefmine testbed, and evaluated. The central finding is that there is little to no impact on the system's performance when using the approach, even though the incentive to manipulate is greatly reduced.

9.1 Motivation

As described briefly in Chapter 2, manipulation problems arise in social choice when voters can change the outcome of the election by casting a ballot which they know to be an inaccurate reflection of their preferences. As a simple illustrative example, consider the election in Figure 9.1, which involves four voters (v_1,...,v_4) casting ballots over four candidates (c_1,...,c_4), and using the **Plurality** system to decide the election. Ties are to be broken lexicographically (i.e. $c_1 \succ c_2 \succ c_3 \succ c_4$). When voters express their true preferences, there is a multiway tie between all four candidates in the election, which causes c_1 to win via the tie-breaking rule. However, if v_4 instead casts the ballot denoted v_4' in Figure 9.1, swapping the order of candidates c_3 and c_2, then c_3 will now win the election outright. Since v_4 prefers c_3 to c_1, there is a good reason for v_4 to misreport its preferences.



c_3 c_1 c_2 c_4 v_1
c_4 c_1 c_2 c_3 v_2
c_1 c_2 c_4 c_3 v_3
c_2 c_3 c_4 c_1 v_4
c_3 c_2 c_4 c_1 v_4'

Figure 9.1: An example of an election with a profitable manipulation for voter v_4 under the plurality system. Initially all candidates are tied, but tie-breaking is lexicographic. c_1 is the winner. If v_4 changes their reported preferences to v_4' then c_3 wins instead.

It may not be immediately obvious why this behaviour is undesirable. If ballots are supposed to be a reflection of a voter's desires, and v_4 would rather have c_3 win the election, then why should v_4 cast a ballot that does not cause c_3? A careful inspection of the ballots shown in Figure 9.1 will reveal that c_1 is actually a Condorcet winner, if it wins ties in the pairwise contests. Even if c_1 does not win ties, c_2 is the only other candidate to win any pairwise contests outright, and c_1 is the first or second most preferred candidate for three fourths of the electorate. The manipulation performed by v_4 has moved the winner from a more popular candidate to a less popular one.

Note that the example above assumes both that v_4 has enough information to vote strategically (i.e. they know that a tie will occur between c_1 and c_3), and that the other candidates are not going to cast a strategic ballot (for instance, if v_2 moves c_1 to the top of their ballot in anticipation of v_4's vote). This is the canonical setting for the study of manipulation, and there are certainly scenarios to which it is applicable. For example, on a small ad hoc committee, an especially astute member may have determined the vote counts in advance, while other members have not. It is also a reasonable worst-case position, because if no manipulations are possible even when a voter has complete information about its peers, then clearly no worthwhile manipulations exist at all. There do however exist other approaches to modelling manipulation which may be more realistic for the behaviours of larger groups of voters. For example, much work has been done on iterative voting dynamics, where voters repeatedly observe the ballots of their peers and update their votes (e.g. [Tsang and Larson, 2016; Grandi et al., 2013; Lev and Rosenschein, 2012; Rabinovich et al., 2015]. Findings in this domain often more closely mirror empirical results showing that strategic voting is rare [Cain, 1978; Tsang and Larson, 2016] or has minimal impact [Lu et al., 2012; Brânzei et al., 2013] in real world elections, or even show some positive benefits to strategic voting [Grandi et al., 2013]. Despite this, reducing the ability of voters to manipulate elections seems a worthy goal. If the voters would not have engaged in strategic behaviours anyway then (provided any decrease in system performance is small) there is no drawback. On the other hand, if voters would have engaged in strategic behaviours, they will no longer have the opportunity to do so, allowing the system to select winners that are truly representative of the collective preferences of the voters.

An early approach to circumventing the Gibbard-Satterthwaite theorem's results focuses on imposing a "quasi-linear" preference structure on voters. In this structure, voter utility functions must take the form of a function with exactly one local maxima over a common ordering of the outcome space (i.e. they must be single-peaked) [Black et al., 1958]. The canonical example of quasi-linear preferences is a vote on how to set the temperature on a thermostat for a room with several people in it. Each person has a preferred temperature, and their utility declines monotonically at some rate to either side of it. If the centre knows the ranking dimension that voters are collectively using for the candidates (e.g. if the centre knows that voters all agree 20 degrees is hotter than 19 degrees), then the *median rule* consists of asking each voter for their most preferred alternative (as with **Plurality**), sorting the reported values according to the known ordering along the dimension voters are using, and selecting the median value after sorting. Moulin showed that this is a strategy-proof social choice function when all voters have quasi-linear preferences [Moulin, 1980; Arrow et al., 2010]. However, this applies only when voters collectively rank candidates on a single dimension. In many applications this is not the case. For example, in the Martian swarm problem of Chapter 3, the quality of different mining sites depends on at least two factors (a two-dimensional location, and a risk/reward tradeoff).

There is a long history within computational social choice of attempts to circumvent the

Gibbard-Satterwaithe theorem by designing voting rules that are protected from manipulation via computational complexity. Gibbard-Satterwaithe says, in essence, that any voting rule under which every candidate is able to win, and that is sensitive to the preferences of at least two voters, must provide opportunities for manipulation. However, the proof is not constructive, and does not specify exactly what ballot a particular voter should cast to manipulate the outcome. One possible circumvention of the theorem is then to design voting rules for which the problem of determining what ballot a given voter should cast to cause a particular candidate to win is computationally hard. This approach began in the early 90's with Batholdi et al.'s work [Bartholdi III et al., 1989; Bartholdi III and Orlin, 1991], which demonstrated that the problem of manipulations in **Single Transferable Vote** was NP-Hard. A large body of work has since emerged showing the computational hardness of manipulation and other related problems for a variety of different voting rules and preference models [Menon and Larson, 2016; Fitzsimmons and Hemaspaandra, 2015; Narodytska and Walsh, 2014; Faliszewski et al., 2009, 2008, 2010; Faliszewski and Procaccia, 2010; Walsh, 2007, 2009; Davies et al., 2011, 2012].

However, protecting elections from manipulation via computational complexity requires that finding manipulations should be a difficult task most (or ideally all) of the time. Since complexity results typically consider only the worse-case analysis, it is unclear whether the offered protections are significant. Recent work by Isaksson et al. casts doubt on the idea that this approach to circumventing the constraints of the Gibbard-Satterthwaite theorem is reasonable [Isaksson et al., 2012]. The authors provide a *quantitative* proof of Gibbard-Satterthwaite, and conclude that a random vote submitted by a manipulator has a probability that is lower bounded by a polynomial in the size of the election (i.e. the number of voters and the number of candidates) of successfully manipulating the outcome of a given election. A voter interested in manipulation can simply use a Las Vegas algorithm to manipulate the vote, trying preferences at random, computing the outcome of the resulting vote profile, and repeating until his desired outcome is returned. The expected work done is polynomial. Isakson et al.'s result builds on extensive earlier work showing similar results [Friedgut et al., 2008; Conitzer and Sandholm, 2006; Dobzinski and Procaccia, 2008; Procaccia and Rosenschein, 2007]. Walsh also finds that truly difficult manipulation problems are uncommon [Walsh, 2011, 2009], questioning the resistance offered by computational hardness. On this basis, further examination of the computational hardness of imputation-based approaches is avoided, but might be an interesting avenue of future work. In particular, obtaining an empirical estimate of the true probability of manipulation should be possible for specific imputation methods, and might allow for robust strategy-proofness through computational hardness.

A final approach to circumventing the Gibbard-Satterwaithe theorem is to use a randomized voting system, rather than a deterministic one. A randomized or stochastic voting system selects an outcome from some probability distribution over the candidates that is a function of the voters' preferences. Gibbard [1977] showed that it was possible to create randomized mechanisms that were strategy-proof, but that the only such mechanisms were those constructed entirely of "unilateral" (i.e. dictatorial) mechanisms and "duple" mechanisms (i.e. votes over just a single pair of the candidates). An example of such a mechanism is the class "Random Dictator", which picks an outcome by first sampling a ballot at random, and then selecting the outcome that is specified by that ballot. Voters have no incentive to misreport their preferences because if their ballot is not drawn, there is no effect, and if their ballot is drawn they get exactly what they want by being truthful. An example of a mechanism based on duple rules would be one that sampled two candidates uniformly at random, and then held a pairwise runoff between them using the **Majority** rule, selecting the winner of this contest as the outcome of the election. Voters should submit an accurate ranking of their preferences because any information about candidates that are not selected for the runoff is unused, and voters cannot help the candidate they prefer in the runoff by ranking them lower than the candidate they dislike.

Although Gibbard [1977] showed that the only strategy-proof voting rules were trivial, there has been considerable recent interest in creating voting systems that leverage stochastic approaches to mitigate the impact of manipulation, and it is this trend that will be drawn on to reduce manipulation's impact on the imputation-based approach. An approach that is particularly appealing is based on the idea of *approximations* of voting rules, original proposed by Procaccia [2010], and extended by Birrell and Pass [2011]. The core idea underlying such schemes is to construct rules that are strategy-proof, but that produce outputs that are "close" to more conventional rules that are not strategy-proof. Procaccia's approach to approximating a scoring rule, for example, is to sample a ballot uniformly at random, but then to sample a candidate from the ballot with probability proportionate to the score that each candidate receives from the ballot. For example, under **Borda**, the candidate at the top of the sampled ballot would be selected with probability

$$\frac{|C|}{\sum_{i=1}^{|C|} i}$$

Procaccia shows that the Borda score of the candidate that is selected as the winner by this scheme is within a small bound of the score of the true winner, in expectation. Birrill and Pass adopt a slightly different framework, which is better suited to the experiments that are performed in this thesis. However, Procaccia's approach could also be adopted. There have also been a number of papers extending the work of Birrill and Pass [Lee, 2015; Filos-Ratsikas and Miltersen, 2014]

9.1.1 Why Worry About Manipulation?

As discussed earlier, a consequence of every classifier being a social choice function is that every classifier is subject to the Gibbard-Satterthwaite theorem [Gibbard, 1973][Satterthwaite, 1975]. This theorem states that every social choice function over a C such that $|C| > 2$ is either *manipulable*, *dictatorial*, *non-resolute* or *imposing*. Formally, these properties are defined for a social choice function S as follows:

1. S is *manipulable* if *ex post facto*, a single agent i could switch their ballot from b_i to b'_i, and $S(B_{-i}, b'_i) \succ_i S(B)$. That is, if an agent with knowledge of all other votes can change the outcome in truthful equilibrium to an outcome they prefer more by changing only their own vote.

2. S is *dictatorial* if there exists some i such that for every valid set of ballots B, $S(B) = S(B_i)$. That is, the outcome is always selected by exactly one vote, always from the same voter.

3. S is *non-resolute* if for some vote profile B, $|S(B)| > 1$. That is, there exists a vote profile for which S cannot break a tie.

4. S is *imposing* if there exists some outcome $o \in O$ such that for every valid set of ballots B, $S(B) \neq o$. That is, there is some outcome that no profile of votes can produce.

Although the imputation-based approach to social choice is subject to the Gibbard-Satterthwaite Theorem, it is not immediately apparent that this should be concerning. For instance, if imputation-based approaches generally did not need to avoid imposition, then perhaps they could be readily made strategy-proof without needing to circumvent the theorem in a more creative fashion. It is also unclear whether all of these properties are even desirable in the context of the imputation-based approach. However, if all of them are important, then other methods must be adopted to circumvent the theorem. For this reason, the importance of each property in the context of imputation is now briefly considered.

Clearly manipulation is something which must be avoided by an imputation-based voting policy. If the policy is manipulable then a shrewd voter can change the imputed votes of other players. Since these votes are then used to compute the election results this, at minimum, this yields a manipulable election. This is especially disconcerting because a single voter is potentially changing many votes (not just his own). Therefore, a direct compromise on manipulability is not possible.

Another possibility is a dictatorial imputation method. In the context of imputation, if a classifier were dictatorial then the selection of imputation policy would have to depend on the preference profiles of a single specific agent alone. While this would rule out most complex imputation methods, this constraint could be relaxed if a variant of 'hot-deck' imputation were used, where the imputed values for B are simply all set to the value of e.g. b_1 (i.e. the first completed ballot). This approach seems quite undesirable however, since it makes no effort to impute the actual values agents would have specified for the missing components - the imputed values will be essentially meaningless. This gives the dictator exceptional power for many partial vote profiles, and thus will often transform the election as a whole into a choice by the dictator. In such a case there is no reason to use imputation - one can simply ask the dictator for her preferences directly and pick a winner on that basis. More complex dictatorial methods are also possible, but all suffer from the same basic weakness. Therefore, dictatorial methods are not suitable for selecting an imputation policy.

Resoluteness also seems a natural requirement of any imputation system. If an imputation method is not resolute then it returns more than one imputation of the partial preference profiles in B. Since these profiles are used to select a winner in the election, this is equivalent to holding two elections, with possibly different outcomes, which either results in two different winning outcomes or requires some method of choosing between them.

Imposition is the most promising avenue for avoiding concerns about the Gibbard-Satterthwaite Theorem. Formally, a classifier is non-imposing if, for any set of partial orders B, and a valid completion of those orders (i.e. one which produces only total orders that are consistent with B), $\hat{B}$, there exists a set of completed orderings $\hat{B}$ such that $I(B) = \hat{B}$. Essentially every classifier is impositional in the sense that it consists of picking a *policy* - a deterministic mapping from partial preferences to full preferences. This means that any two identical partial ballots in B must be completed in exactly the same way. A voter who prefers outcomes in which the imputed ballots are distributed in some other way cannot realize their preference. This is certainly not an implausible preference. For instance, if two preferences are nearly tied in the completed ballots, but the imputation algorithm fills all incomplete ballots with the more popular of the two, then those voters disadvantaged by this arrangement may prefer a distributional system instead. However, lifting this restriction would require voters with identical partial preferences to be treated differently, which may also be undesirable.

If avoiding imposition is desirable, then deterministic mappings that specify a distribution over the outcome space of the entire election (O) for each possible input, and impute accordingly, could be allowed. This might be preferable in settings where the imputed votes have low variability relative to their total number, but would be non-imposing and thus vulnerable to manipulation. It would be easily implemented with something like the Markov Tree approach presented earlier in Chapter 7. However, it might still be preferable to avoid manipulation entirely designing a strategy-proof version of the imputation-based approach. This is the topic of the remainder of the chapter.

9.2 Addressing Manipulation with Differential Privacy

In their 2011 work "Approximately Strategy-Proof Voting", Birrell and Pass [2011] show that a voter's ability to manipulate an election can be reduced by introducing a randomized component into the ballots, rather than into the voting rule itself. This is an elegant construction from the perspective of the imputation-based approach, because one of the features of the approach was the ability for users to make use of existing and popular rules intended for total orders. As an example to demonstrate the intuition behind this approach, suppose that the votes are to be counted by hand, and that hand counting is unreliable. Even if a voter knows exactly what ballots all other voters will cast, and knows how to cast a ballot to manipulate the outcome, these small errors in counting may reduce or eliminate the advantage of casting a manipulative ballot. As a more concrete example, return to the ballots in Figure 9.1. Suppose that hand counting will introduce an error in the ballot counts with probability 0.5. The error will favour one of the candidates, increasing their tally by a single point, and reducing the score of some candidate by the same amount. Effectively the voting system will first of all select c_1 outright with probability $\frac{4}{8}$ (since there is a 0.5 chance that no corruption will occur and c_1 is the current winner). v_1 is randomly selected as the vote to be corrupted with probability $\frac{1}{8}$. Suppose this is the selection; then c_3 will lose a point, since v_1 ranks c_3 first. One of the 4 candidates is selected to gain a point. If this turns out to be c_1 or c_3, then c_1 will still be the winner. If this turns out to be c_2, then it will be the winner (and likewise for c_4). Thus, once v_1 has been selected for corruption, c_1 wins with probability 0.5, and each of c_2 and c_4 with probability 0.25. Similar logic applies to each of the other ballots that could be corrupted. Therefore, the overall probability distribution is:

$$P(c_1) = \frac{1}{2} + \frac{1}{8}(3\frac{2}{4} + \frac{1}{4}) = \frac{23}{32}$$

$$P(c_2) = P(c_3) = P(c_4) = \frac{1}{8}(3\frac{1}{4}) = \frac{3}{32}$$

Now if v_4 casts ballot v_4', the probability distribution over the winner changes so that c_3 wins outright with probability $\frac{1}{2}$. If either of the votes for c_3 are randomized than there is a $\frac{2}{4}$ chance of c_1 being declared the winner, and a $\frac{2}{4}$ chance of c_3 being declared the winner. If the vote for c_1 is randomized, c_3 wins with certainty. If the vote for c_4 is corrupted there is a $\frac{1}{4}$ chance of c_1 winning, and a $\frac{3}{4}$ chance of c_3 winning. Therefore the new distribution over the winners will be given by

$$P(c_1) = \frac{1}{8}(2\frac{2}{4} + \frac{1}{4}) = \frac{5}{32}$$

$$P(c_3) = \frac{1}{2} + \frac{1}{8}(2\frac{2}{4} + \frac{4}{4} + \frac{3}{4}) = \frac{27}{32}$$

Introducing the potential for errors in counting has thus reduced the potential benefits of casting a manipulative ballot, provided that v_4 is risk-neutral or risk-averse. This is because in the original election, v_4 was already more likely to get someone they preferred to c_1 elected, and in the election with the manipulating ballot, v_4 is less likely then before to avoid having c_1 elected. The resulting rule could be implemented by selecting a *ballot* uniformly at random before the voting rule is applied, and replacing that ballot with one generated uniformly at random[1].

On its own, the observation that the randomization of some of the ballots in an election reduces the potential gain for a manipulator is not especially useful. If the gain is still positive, the manipulator will be undeterred. The other observation made by Birrell and Pass is that strategic behaviour often has an associated cost. For example, finding a manipulation (or even determining whether one exists) might be computationally expensive (even if it is not intractable). Social pressures in a public ballot might make casting a strategic ballot damaging to one's reputation[2]. In domains where partial preferences are permitted, a voter might even have to expend considerable effort just to determine whether a strategic vote is a good idea. If the expected benefits of casting a manipulative ballot are low, and the costs are sufficiently high, then a rational voter would choose not to invest

[1]If a manipulator has risk-seeking preferences, they might derive extra utility from taking big gambles, and so might prefer to manipulate even if their expected payoffs are reduced.

[2]Observe for example, the response from members of Canada's New Democratic Party to the strategic voting in a recent election [Petrick, 2015; Westlake, 2015; News, 2015]. The author has also observed much more radical statements made by friends and acquaintances on this issue.

the effort in casting a strategic vote. Similarly, if a coalition of voters wants to sway the outcome of an election, the coordination costs may exceed the expected gains.

The main results demonstrated by Birrill and Pass show that it is possible to build reasonable approximations of voting rules that prevent any manipulations that have higher expected value than some constant ϵ. Voters are permitted to have any utility functions satisfying the modest constraints that the utility for having any given candidate win must be a real-value between 0 and 1, and that if a voter prefers a candidate c_i to some other candidate c_j, then the voter receives at least as much utility from c_i winning as from c_j winning. Under this utility model, ϵ can be understood as the largest amount any voter can hope to gain in expectation by submitting a manipulating ballot. A voting rule that satisfies this property is called ϵ-strategy-proof.

Rephrasing the results from Birrell and Pass [2011] into the notation of this thesis, the *corruption distance* between a ballot profile B and an outcome o is the minimum number of votes that would have to be *corrupted* (i.e. changed to an arbitrary preference ordering) for voting rule S to return o. Formally, this is defined as:

$$\ell(B, o, S) = \min_{B' \ s.t. \ S(B')=o} \Delta(B, B')$$

where Δ is a component-wise delta function over the two sets of ballots (i.e. a count of the number of ballots that differ between the two sets). The corruption distance between two distinct social choice functions S and S' is analogously given by

$$d_v((B, S(B)), (B, S'(B))) = |\ell(B, S(B), S) - \ell(B, S'(B), S')|$$

where the first term will always reduce to zero (since $min_{B' s.t. S(B')=S(B)} \Delta(B, B') = 0$). The second term thus measures the number of votes that would have to be corrupted for $S(B)$ to return the same result as $S'(B)$ on a complete set of ballots B. Finally, S is said to be a δ-approximation of S' if their corruption distance is less than δ for *every* preference profile input B. Note that if S' is stochastic, this must hold for every *possible* outcome S' could produce on a given profile.

The primary result of Birrell and Pass [2011] can now be stated formally as follows: Given a social choice function S, it is possible to create a δ-approximate voting rule (where $\delta = \frac{|C|(|C|+1+\epsilon)}{\epsilon} - 1$) that is ϵ-strategy-proof for any $\epsilon > 0$, where δ is the corruption distance between the approximation and S, and $\epsilon > 0$. In practice, this mechanism functions by sampling an outcome c with probability proportionate to

$$max(1 - \frac{\epsilon}{|C|(|C|+1+\epsilon)} d_v((B, S(B)), (B, c))), 0)$$

One problem with the proposed mechanism is that it is not a very good approximation when the number of ballots cast is small relative to the number of candidates. In particular, unless $|B| \gg \frac{|C|^2}{\epsilon}$, the true winner will need to win by a landslide to be picked with high probability. In practice, one might expect a value of ϵ near 0.01 to be reasonable, since a vote who can gain at most one part in a hundred of their maximum utility from a manipulation is unlikely to consider it worth the effort. In an election with 14 candidates like the Meath set however, this would require a candidate to lead by an amount requiring the corruption approximately 18,000 ballots before they would be certain of victory, and even a lead requiring the corruption of 9,000 ballots would only make them about twice as likely to win as other candidates. Clearly this is a low-fidelity approximation.

Fortunately, Birrell and Pass address this possibility as well, via the incorporation of techniques from differential privacy - the field of anonymizing data in a way that preserves the aggregates with high probability [McSherry and Talwar, 2007]. This should seem a natural extension, since a social choice function can be viewed as an aggregate computed over votes. Before discussing this secondary result, more notation from Birrell and Pass [2011] must be provided. A voting rule S' is said to be a (δ, μ)-approximation of S if it returns an answer more than a corruption distance of δ from S, *with probability no greater than* μ. For instance, a $(2, 0.25)$ approximation returns an answer equivalent to one produced by the corruption of at most 2 votes at least $\frac{3}{4}$ of the time. The secondary result of Birrell and Pass [2011] is then as follows (with adapted notation):

Theorem 37. *For any deterministic voting rule S and any $\epsilon > 0$, $\delta > 0$, S has an ϵ-strategy-proof (δ, μ)-approximation S', where* $\mu = \frac{|C|-1}{(\epsilon+1)^{(\frac{\delta+1}{2})}+|C|-1}$.

The precise details of this mechanism involve selecting an outcome c with probability proportionate to:

$$\exp(\frac{\ln(\epsilon+1)}{2}(|B| - d_v((B, S(B)), (B, c))))$$

To demonstrate the precise value of the mechanism, suppose it is desired that $\epsilon = \mu = 0.01$. Solving for d produces

$$\delta = \frac{2\log(\frac{(1-\mu)(|C|-1)}{\mu})}{\log((\epsilon+1))} - 1$$

So substituting the values from the Meath example,

$$\delta = 2 * \log((1 - 0.01)(14 - 1)/0.01)/\log((1.01)) - 1 \sim 1,428$$

which implies that the outcome selected will be within a corruption distance of $1,428$ with probability 0.99, and the mechanism does not allow manipulations that result in a gain of more than 0.01 points of subjective utility for any agent. These quantities appear workable. Additionally, the mechanism exhibits a very rapid decrease in the probability of picking other candidates. For example, a candidate that wins by a margin of say, 100 ballots, under a mechanism with $\epsilon = 0.01$ will be nearly three times as likely to be chosen as the winner. If $\epsilon = 0.02$ is used, then this increases to a factor of more than 7. At $\epsilon = 0.03$, the ratio is nearly twenty to one. The exact choice of ϵ will naturally depend on the application domain under consideration. If manipulations are of great value or are very easy to compute and execute, then lower values of ϵ must be used. If the election is expected to be tight, (i.e. a small margin of victory), then instead a higher value of ϵ should be used, to ensure recovery of the correct winner with high probability.

Birrell and Pass do not discuss how the corruption distance ℓ should actually be computed, and for certain voting rules this appears to be a computationally difficult problem, since it amounts to asking whether there exists a subset of the electorate who could manipulate the outcome. However, even ignoring the results discussed earlier in the chapter, which suggest such manipulation problems are often not very challenging, there are many voting rules for which the problem is immediately tractable. For example, under **Borda**, suppose that c^* has won the election with a Borda score x points higher than some other candidate c. For a given ballot, it is easy to compute contribution of the ballot to the Borda score advantage of c^* over c, which is just the number of positions c^* appears ahead of c by (it will be negative if c is ranked higher on a given ballot). If the ballots are sorted according to this measurement, then the ballots giving the highest advantage to c^* over c can be removed, and replaced with a ballot that ranks c^* last and c first. This process can be repeated until c^* no longer has a higher Borda score than c, and the number of ballots that were replaced will clearly yield the correct corruption distance. A similar process can be performed for any monotonic scoring rule.

In the case of the imputation-based approach however, this problem is greatly compounded because ℓ is the minimum number of (partial) ballots in the input that must be corrupted to cause a given classification algorithm to output a completion that is in turn interpreted by a voting rule as providing a different outcome. Computing how a given learning algorithm will respond to arbitrary changes to subsets of its inputs is not straightforward, and while robust classification algorithms exist (e.g. [Feng et al., 2014]), they are robust only in the sense that *adding* at most some number of adversarial ballots will not change the estimates of the model too much. However, since computing ℓ involves corrupting arbitrary subsets of the data, this approach would not provide an effective guarantee. Earlier approaches to adversarial classification [Dalvi et al., 2004] provide protection even against adversarial replacement of exemplars, but are tailored to specific classifiers like Naive Bayes.

9.3 Robustness of Imputations

Although it is not straightforward to compute the distance measure ℓ, it is still possible to provide some empirical support for the idea of ϵ-strategy-proof versions of the imputation-based approach to social choice, by assessing the robustness of the imputation-based approach when many ballots are corrupted. To accomplish this, the proportion of problem instances where changing the output of the learning algorithm on a subset of the ballots changes the outcome of the election is assessed empirically. In reality, ℓ requires the computation of the effect of *any* such subset, so these results provide only an exploratory and somewhat informal assessment of the values of ℓ that might be present in real world datasets. It may be useful to future researchers when considering avenues of research into which combinations of imputation algorithms and social choice functions are easy to manipulate.

In the experiment, a modified version of the Prefmine testbed system 6 was created. Following the imputation step, a subset of the completed ballots was sampled uniformly at random and removed entirely. The aggregate scores of each candidate were then computed under each of several scoring rules, and the margin of victory between the candidate with the highest score and the runner-up was measured, and then compared to the maximum change in the margin that could be achieved by replacing the removed ballots adversarially. The Markov Tree learner from Chapter 7 with a model depth of $k = 3$ was used. Experiments were performed on 100 problem instances generated from each of the three Irish datasets (Dublin North, Dublin West, and Meath). The number of ballots that were replaced adversarially was determined by the formula

$$2 * \log((1 - 0.05)(|C| - 1)/0.05)/\log(1 + \frac{1}{|C|}) - 1$$

because an outcome requiring the corruption of fewer than this many ballots would be selected 95% of the time in a $\frac{1}{|C|}$ approximation of the imputation-based voting rule, as discussed in the previous section (i.e. this is the value of δ when $\mu = 0.05$ and $\epsilon = \frac{1}{|C|}$).

The results of these experiments are summarized in Figure 9.2. Each figure shows three violin plots, one for each of the Dublin North, Dublin West, and Meath datasets[3]. Each violin plot shows the distribution of the margin of victory that was observed after removing a subset of the ballots. Note that these margins are actually much smaller than what might be expected in elections of this size, because only about 10% of the original ballots are complete in each election. A dashed red line through each plot shows the maximum change in the margin of victory that could be achieved by replacing the removed ballots adversarially. Overall the results suggest that both **Borda** and **K-Approval** are usually difficult for a randomly selected subset of voters to manipulate, and consequently that they would be fairly easy to protect using an approximately strategy-proof mechanism. The majority of runs under both Dublin North and Dublin West were not manipulable even by an adversary with the power to replace more than 100 ballots adversarially. On both rules, Meath was more difficult to protect completely. Note however that even if the outcome can be changed, an approximately strategy-proof system will prefer (by exponential factors) outcomes that are closer to the winner.

9.4 Discussion

The most promising route toward making a strategy-proof version of the imputation-based approach to social choice appears to be via the ϵ-strategy-proof approach. Birrell and Pass [2011] suggests that for *any* combination of voting rule and imputation algorithm, an approximately strategy-proof rule exists, but it is defined in terms of ℓ, a distance measure that is not straightforward to compute for the imputation-based approach. Interestingly however, if an efficient algorithm (or approximation algorithm) *does* exist for finding manipulations for a given combination of voting rule and imputation method, then such an algorithm can be used straightforwardly to compute ℓ for that combination, and thus create an ϵ-strategy-proof version of the method. The construction and characterization of such algorithms is left to future work however.

Despite the difficulty of computing ℓ, a small experiment was performed to assess the potential of the approximately strategy-proof voting to work with the imputation-based approach. It was found that under **Borda** and **K-Approval**, the margins of victory in real world elections were often large enough that the implementation of the system would have

[3]A violin plot is comprised of a box plot (black, middle of each violin), with a smoothed histogram to either side. They show a more complete shape of the distribution than a box-plot alone [Hintze and Nelson, 1998]

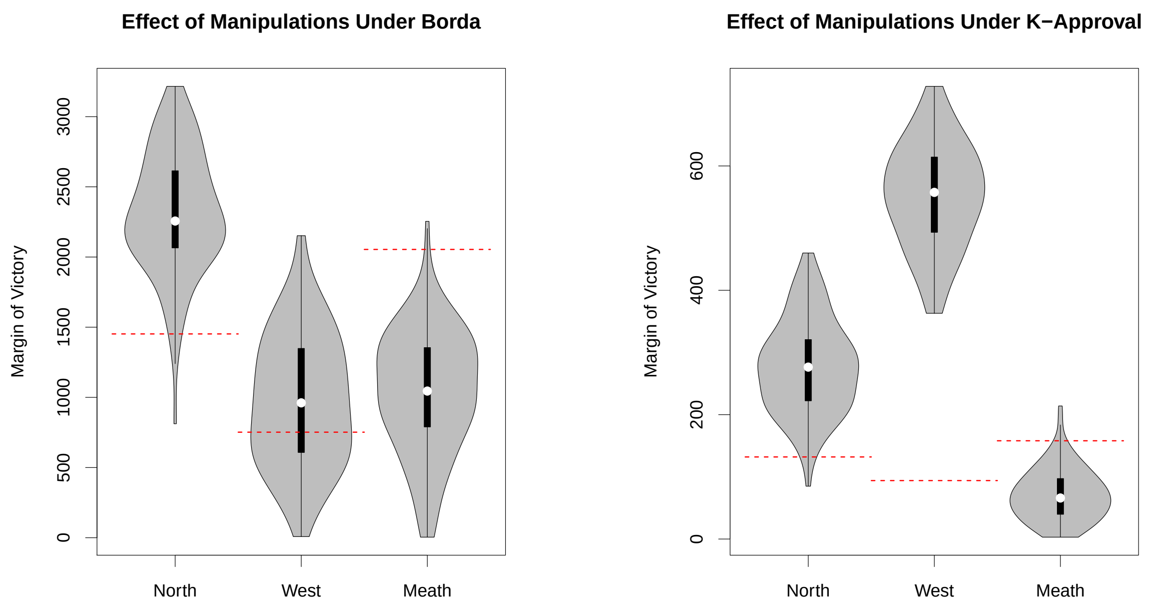


Figure 9.2: Violin plots showing the distribution of the margin of victory under the **Borda** voting rule (left) and **K-Approval** voting rule (right). Dashed red lines show the maximum decrease in the margin that could be achieved by replacing the removed subset of the ballots adversarially.

very little impact on the choice of outcome. Assessing the performance of a strategy-proof version of the imputation-based approach remains an open problem.

Chapter 10

Conclusion and Future Work

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

Winston Churchill, [1942]

In the first chapter of this thesis, the overarching goal of the research contained herein was described as answering the question: **Can existing techniques from machine learning be used in a novel way, in order to provide improved computational social choice solutions and hence, to make better group decisions in multiagent systems?** The question was broken down into three sub-parts, which respectively concerned how exactly machine learning techniques ought to be used; whether the techniques were effective or not in the chosen application; and whether there was a principled way to compare the "fairness" of different machine learning algorithms that might be applied. Over the past seven chapters, these questions have been answered in detail, with the presentation of a new approach to the problem of social choice with incomplete information, called the imputation-based approach. The proposed system was validated using a novel experiment design on real world electoral data, and found to offer substantial advantages over modern competitors. As well, a framework for representing and analyzing machine learning algorithms as social choice functions, or as part of composite social choice functions, provided an interesting theoretical connection between computational social choice and machine learning, and provided a new set of tools to help to answer the question of which algorithms were best suited to applications in social choice. The proposal of a new learning algorithm that was intended specifically for use with the imputation-based approach, and the creation of the carefully designed Prefmine testbed system, are other significant contributions.

This chapter serves to summarize the contributions of the thesis, to provide direct contrast with related work, including some that has not yet been discussed, and to propose future research directions that build upon the work contained in the thesis. The chapter begins with a brief example of an artificial application domain in multiagent coordination, wherein the advantages of the proposed approach are elegantly highlighted. The core contributions of the thesis are then summarized, and contrasted with some additional related work. The final section provides ideas for future work.

10.1 The Coordination Example

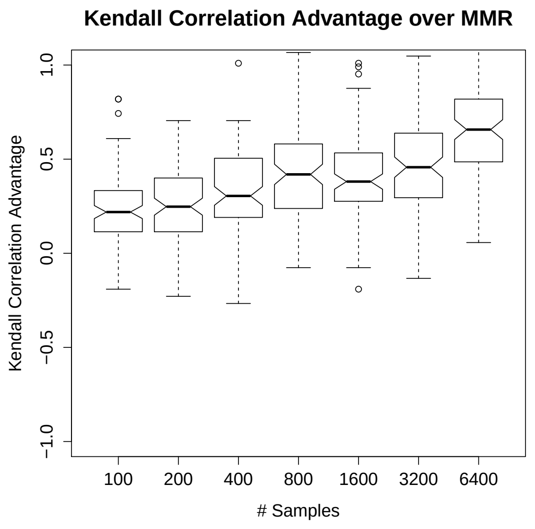
As emphasized near the end of Chapter 5, the experimental design adopted in this thesis, although based on real-world data, required the use of assumptions about how missing information was to be distributed. In the interest of fairness to competing methods, a neutral distribution was picked. There was no direct correlation between the candidates a voter ranked highly and the probability that a vote ranked very few candidates overall. Although the imputation-based approach offered performance advantages even under these quite restrictive assumptions, the advantages were not as large as those that would be present when the data are better aligned with the assumptions of the imputation-based approach. In effect, this experiment design matched the assumptions of the random competitor.

In many situations, the candidates a voter prefers may be intimately linked to the voters' general knowledge about the candidates, especially in applications that are further removed from political contests, like multiagent coordination. As a concrete example, let us return to and expand the Martian swarm scenario described in Chapter 3 and 4. In this scenario, mining sites and the headquarters of various mining companies were distributed on a two-dimensional grid. The companies wanted to mine resources from the sites, but are required to coordinate, and to operate their robots as a team or swarm. It is riskier to operate one's robots further from one's headquarters, but each company values the risk differently. Companies have different information about the quality of different sites. Suppose that viable mining sites are modelled as points distributed on the grid according to a two-dimensional Gaussian, centered at the origin, but that mining companies' headquarters are distributed uniformly at random throughout the space. Each site has an expected profit that will be received from mining it. The probability that a company knows enough about the site to estimate its expected profit is proportionate to the true value of the site, divided by distance between the company's headquarters and the site[1]. This creates a scenario where companies with headquarters near the origin will have lots of information, but companies further away will have much less.

In this scenario, there is a strong correlation between the candidates voters prefer and the number of candidates voters will be able to rank. Those with headquarters near the origin will have information about many different candidates. Those located further away will have information about only a few. However, the ordering of those few points will tend to provide a great deal of information about how the company would value the various unranked points near the centre, because both valuations are dependent on the position of the company relative to the origin.

An implementation of the above problem domain was made within the Prefmine testbed system. In this implementation, a space consisting of real-valued numbers between $(-5, 5) \times (-5, 5)$ was used for the grid. Mining sites were sampled from a Gaussian distribution with mean $(0, 0)$, and standard deviation of 1.5 along each major axis, with no covariance. If two sampled sites were within a Euclidean distance of 0.1 units of each other, one of the sites was re-sampled. The locations of company headquarters were sampled uniformly at random, with the constraint that no company could have headquarters within a Euclidean distance of 0.1 units of a mining site, or of any other company's headquarters. Each company had a utility function that placed the expected value of mining a site at $\frac{1}{r\delta}$, where r was the company's risk parameter, sampled from an exponential distribution with parameter $\lambda = 0.5$, and δ was the Euclidean distance between the company's headquarters and the mining site. A company knows the (correct) quality of the closest mining site with certainty. Given that a company knows the quality of the k closest sites already, it knows the quality of the next nearest site with probability $\frac{2}{1+e^{\delta}}$. For simplicity, no company knows the quality of more distant sites without also knowing the quality of less distant ones. Each mining company operates 1 robot for simplicity. A company's ballot ranks a mining site only if its quality is known. Candidates are ranked according to the company's utility function. A third order Markov Tree was used to impute the ballots for the imputation-based approach. The Markov Tree was selected because this is clearly a domain where the candidates can be embedded in a low dimensional space, so the model is expected to perform well, as discussed in Chapter 5. The **Borda** and **K-Approval** voting rules were used to decide the election. All experiments used 15 mining sites, and the number of companies varied between 100 and 6400, with 100 problem instances being generated for each number. Results were also collected for MMR and the random approach. Figure 10.1 shows the advantage in Kendall Correlation for the imputation-based approach over each competitor under **Borda**, while Figure 10.2 shows the same measure-

[1]This corresponds to the notion that information about more valuable sites will spread further than information about less valuable ones, all else being equal.



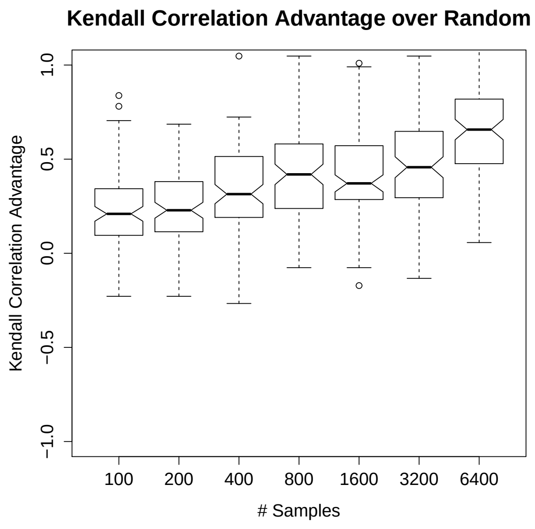
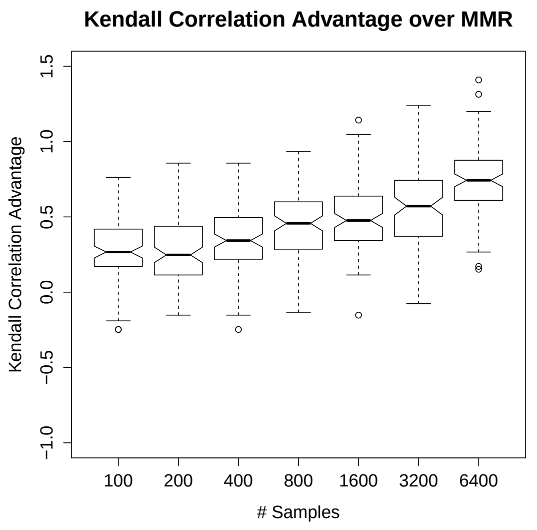


Figure 10.1: Improvement in Kendall Correlation from using the imputation-based approach in the coordination example problem under the **Borda** voting rule. Box plots show the distribution of advantage over 100 problem instances for different numbers of companies. Notches indicate a 95% confidence interval for the median [Chambers, 1983].

ment under **K-Approval**. The advantage is the Kendall Correlation between the outcome using the imputation approach and the ground truth, less the Kendall Correlation between the outcome using each competitor and the ground truth, on each of the 100 problems for each parameter setting. Values greater than 0 indicate advantage for the imputation-based approach.

The advantage of the imputation-based approach is very pronounced in this example application, and for the 6,400 company case, begins to approach the distance between the correct outcome and an outcome sampled uniformly at random. Indeed, examining the raw results, the competing methods are often selecting outcomes that are worse than randomly selected ones. The reason for this is that the problem domain produces (by design) ballots with very extreme and asymmetric missingness. Most ballots will rank only one or two of the 15 candidates. The exception to this is for voters located near the centre of the grid, who will be close to many candidates, and so will rank many candidates. In the case of MMR, the extreme missingness gives wide latitude to make nearly any candidate win the election. This problem is not helped by having a larger number of ballots to work with, since these ballots are also highly incomplete, and while any candidates they order provide



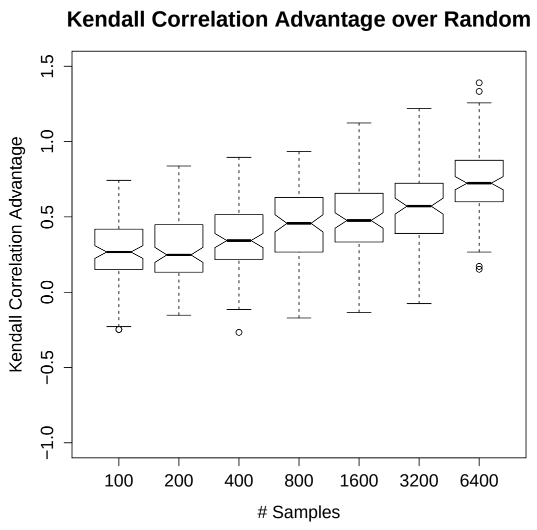


Figure 10.2: Improvement in Kendall Correlation from using the imputation-based approach in the coordination example problem under the **K-Approval** voting rule. Box plots show the distribution of advantage over 100 problem instances for different numbers of companies. Notches indicate a 95% confidence interval for the median [Chambers, 1983]. Note the change in the limits of the vertical axis.

MMR with some information to constrain its selection of the winner, this is outweighed by the unconstrained choices for the ordering of the remaining candidates these extra ballots do not order. In essence, additional ballots add more degrees of freedom than they subtract, and MMR is picking a winner by solving a nearly unconstrained optimization problem. The random approach (the author's casting of the MLE competitor [Xia and Conitzer, 2011]) also fares rather poorly for this reason. There is a lot of noise in the relative positions of the candidates using only the given preferences. In contrast, the imputation-based approach utilizes features in the given ballots to determine how voters would have voted very efficiently. If two voters rank the same candidate highly, they must be close to one another in the grid, and so will have correlated opinions about more distant candidates. Effectively, each ballot improves the quality of the machine learning model, without increasing the complexity of the imputation task significantly. Given enough data, the Markov Tree will be able to predict a voter's preferences with a great deal of accuracy, given only their first few preferences. The example illustrates the great strength of the imputation-based approach in domains where voters' preferences truly do exhibit structure of this kind.

10.2 Conclusion and Discussion

As outlined in the introduction of this chapter, the thesis has made a number of distinct contributions that were presented over the previous seven chapters. The precise nature of each contribution is expanded upon here, providing both the details of the contribution's scope, and a suggestion of which communities might find the results most useful.

The core contribution of the thesis is the design and validation of a new approach to the problems of group decision making with incomplete information, one that is predicated on leveraging machine learning and imputation to address missingness in a manner that aligns with users' true preferences. These problems arise in many contexts, ranging from political contests to coordination problems in multiagent systems, and the demonstrated performance of the new imputation-based approach ensures that practitioners in these areas will have a powerful new tool for making better group decisions. As shown by the Martian swarm example in the previous section, and as discussed in Chapter 4, if the problem domain is more consistent with the assumptions made by the imputation-based approach then the performance gains can be pronounced, while the results of Chapter 5 demonstrate that the approach performs well even when the problem domain is more neutral.

In addition to providing improved performance, the new system offers a tangible connection between the fields of machine learning and computational social choice. The demonstration in Chapter 4 that the imputation process itself is a form of social choice is the key avenue for this conclusion. In addition, the axiomatic analysis of Chapter 8 that provided a means to assess the axiomatic fairness of machine learning algorithms as well as compositions of learning algorithms and social choice functions, also serves to provide a new bridge connecting the fields, and fertile ground for theorists from either domain to obtain more refined answers to the question of which learning algorithms are best suited to social choice problems. The analysis of the best way to create strategy-proof approximations of imputation-based voting systems introduced in Chapter 9 further contributes a compelling problem area for future theoretical study.

The Prefmine testbed system, described in Chapter 6 offers researchers in computational social choice a straightforward way to validate new techniques for *social choice with incomplete information*, using the experimental methodology described in Chapter 5, namely learning a distribution of missingness from real world data, and then using that model to ablate ballots for which the ground truth was known to create realistic problems. This methodology allows experimenters to make more realistic datasets than might otherwise be available, and so may facilitate the design of more practical systems in the future. The careful construction of the system also provides assurances against experimenter error, and enables replication of existing results with ease. As a complement to the work of others in the computational social choice community which provide collections of real world data [Mattei and Walsh, 2013], the testbed system will allow this data to be used in a consistent fashion. Further, the testbed system provides a direct means for practitioners to analyze data, or to implement the techniques developed in this thesis for use in any social choice domain. Researchers in areas like recommender systems might also be interested in the Prefmine testbed, as it provides a simple way to evaluate their algorithms in a somewhat different context (group decision, instead of individual recommendation).

Finally, the development of a learning algorithm tailored specifically to the imputation-based approach provided somewhat better decision making performance than conventional machine learning models (offering further gains to potential practitioners), as described in Chapter 7. The Markov Tree Learner also provided more robust theoretical guarantees about the behaviour of the imputation-based approach, both with respect to the amount of data needed for convergence to a reasonable imputation model, and with respect to the ability to learn the parameters of artificial data distributions. The learner was also efficient and the models it produced had an intuitive interpretation (modelling a trajectory through the space in which candidates are embedded), which could help guide practitioners in the parameterization of the learner.

Taken together, these contributions substantially advance the state of the art in the problem of social choice with incomplete information, as well as helping to connect machine learning and computational social choice closer together, and offering powerful new tools for practitioners and experimentalists. It is therefore concluded that existing techniques from machine learning can indeed be used in a novel way, in order to provide improved computational social choice solutions, and hence to make better group decisions in multiagent systems.

10.3 Related Work

There are a number of recent works that are tangentially related to topics covered in the thesis, or that study the same general area without studying the specific problems considered by the thesis. This section will briefly summarize these works, and contrast them with the results of the thesis to provide more contextualization.

The approach adopted in the thesis toward the creation and analysis of social choice mechanisms is conceptually similar to Xia's recently proposed Generalized Decision Scoring Rule (GDSR) framework [Xia, 2015], in which mechanisms are "designed by social choice, evaluated by statistics and computer science". Like in the GDSR framework, the imputation-based approach attempts to minimize the distance between an objectively correct outcome and the outcomes produced by different mechanisms. A key difference between the two approaches is that the GDSR framework has thus far been used to characterize asymptotic performance of broad classes of mechanisms, while the imputation-based approach have been evaluated primarily through empirical tests against real world data. Additionally, GDSRs are axiomatically characterized in terms of a property called "local consistency", which requires that for any two sets of ballots B_1, B_2 over the same set of alternatives, if $I(B_1) = I(B_2)$, then $I(B_1 \cup B_2) = I(B_1) = I(B_2)$, where I is a voting procedure. That is, if two sets of ballots produce the same outcome, their union must also produce that outcome. The imputation-based approach does not have this requirement, and it is actually rather easy to imagine learning algorithms such that this property would not hold. For example, suppose that B_1 contains many incomplete ballots that are similar to a small set of complete ballots in B_2, but dissimilar to every complete ballot in B_1. and vice versa. A sensible imputation algorithm might impute the ballots in each subset very differently from the ballots in the whole, and this behaviour does not appear to be undesirable in the context of the imputation-based approach. Consequently, a different set of axioms, and different corresponding statistical analysis, is required for a full characterization of the imputation-based approach in the abstract.

Lang et al. [2012] considers the problem of determining the winner (i.e. top ranked candidate) in elections containing only partial ballots under both Schulze's rule [Schulze, 2011] and a Tree-based approach wherein a number of pairwise contests are held. Like the imputation-based approach to determining the winner, the raw ballots are used, and no further elicitation occurs (in contrast to say, the approach suggested by Lu and Boutilier [2011b]). However, unlike the imputation-based approach, it is not assumed that voters with similar preferences regarding the ranking of some candidates will also have similar preferences regarding the ranking of others, and the problem studied concerns primarily the computational efficiency of determining the winner under these systems, rather than the quality of the decisions under some objective measure. More generally, two problems are considered at length: determining whether a unique winner exists that *must* win (regardless of how preferences are completed), or determining the composition of the set of *possible* winners (i.e. candidates that could win, under some completion) when performing social choice with incomplete preferences. Both problems have received a great deal of study [Konczak and Lang, 2005; Xia and Conitzer, 2008; Pini et al., 2011, 2007; Baumeister and Rothe, 2012]. The imputation-based approach will always (by definition) select a possible winner of the election, since it imputes ballots in a fashion that is consistent with the expressed preferences of voters. However, it also (for better or for worse) will select a unique winner, corresponding to the assumptions embedded in whatever imputation algorithm was used, even if there are several possible winners. Results in Chapter 5 demonstrate that this approach is quite effective on real world data, and intuitively, it will be effective whenever there are clear patterns in voters' preferences, and there is enough data to learn them.

Procaccia and Shah [2015] develop a framework for eliciting probabilistic statements from voters regarding their preferences (i.e. rather than asking whether a is preferred to b, the algorithm can ask voters for the probability that they will prefer a to b), and efficiently aggregates these sorts of probabilistic preferences to determine an ordering that is close to a central ranking. The techniques were evaluated by starting with data from Preflib [Mattei and Walsh, 2013], and generating a probability distribution for each vote via a Mallows model with a central ranking equal to the vote itself. The approach appears to work well in practice, and is supported by strong theoretical bounds. The main differentiating factors between the model described in the paper and the imputation-based approach are that the imputation-based approach can work with whatever precise relationships voters are willing to provide, whereas the paper's approach works by eliciting probabilistic information. Depending on the problem domain, either option might be preferable. For example, when working with human voters, eliciting (accurate) probabilistic information might not be straightforward, while when working with software agents it might be quite easy. Lu and Boutilier [2013] propose the use of minimax regret for selecting multiple winners as well as single winner [Lu and Boutilier, 2011b], described at some length in

Chapter 5. This approach addresses the same problem as the imputation based approach, and is likely preferable when working in domains where voters can be queried for more information, because it provides powerful heuristics for the order in which further preference information should be elicited. If little information is given and no more can be elicited, the approach can make conservative decisions even in domains with strongly structured preferences, however.

A number of different authors study the problem of inferring an objective central ranking from "noisy" ballots [Caragiannis et al., 2013; Conitzer et al., 2009], that is, ballots which order candidates correctly with higher probability than they are ordered incorrectly. These include algorithms for efficiently learning the central ranking of Mallows distributions [Lu and Boutilier, 2011a] or mixtures of several Mallows distributions [Melia and Chen, 2010], and effectively learning random utility models [Azari et al., 2012; Azari Soufiani et al., 2013], as well as a much longer literature going back to Kemeny [1959] (with notable recent developments in this vein including the empirical studies of Ali and Meilă [2012]). First-order Markov Chains have been considered for the purpose of aggregating partial orders of candidates recently [**?**], but the model proposed in Chapter 7 can learn higher order models, and can be used to impute preferences as well as to learn a central ranking, as demonstrated in Sections 7.5 and 7.6. Interesting recent work also includes finding voting rules that are "consistent in the limit" with many different statistical preference models simultaneously. For example, Caragiannis et al. [2014] show that picking the *modal* ranking from an infinitely large set of ballots will yield the correct ordering in expectation for an enormous family of distributions. Procaccia et al. [2016] shows a similar rule for distributions where noise might be inserted adversarially. The primary difference between these approaches and the imputation-based approach is the assumption of an objectively correct central ranking. The imputation-based approach does not assume that such a ranking exists, and instead assumes only that individual voters have underlying correct rankings, which might be generated from any number of different, unrelated processes. While the techniques described in this paragraph could be used as part of the imputation-based approach (e.g. as imputation algorithms themselves), they are best understood as examining a closely related but complementary problem domain.

10.4 Future Work

As the epigraph for this chapter asserts, this thesis does not constitute the end of the research project that was undertaken, but only the end of the beginning of that project. There remain many interesting and exciting avenues for future work at the intersection of machine learning and computational social choice, building upon the imputation-based approach proposed here. This section ends the thesis by listing a number of such avenues.

One area with great potential for future work is improving the realism of the experimental design used in the Prefmine testbed system. Although the current experimental design is based on real-world data, the ablation models used assume there is no relationship between which candidate a voter has expressed a preference for, and the amount of information the voters have. It also does not provide a straightforward means to generate partial preferences that are not top-orders over the candidates. The example presented earlier in this chapter demonstrates that asymmetric distributions of missingness favour the imputation-based approach, and there are application domains in which such distributions are eminently reasonable. One possible way to generate preferences with asymmetric missingness would be to learn (via the algorithms described in Chapter 7) a variant of a Markov Tree with order $k = |C|$. The tree would also learn the point at which a given sequence ends, and could then be used to generate artificial rankings with similar statistical properties to the original dataset, or to ablate ballots from the original set in a more asymmetric manner. A carefully constructed variant of the Markov Tree learner could also be used to generate arbitrary partial orders, rather than only top orders, provided that a set of partial orders was available to learn from.

Chapters 5 and 6 also included a discussion of the problem of bias in the imputation method, and how it relates to the problem of class-imbalance in machine learning more generally. When some candidates are not ranked at all, or are ranked by very few voters, classification models will tend to have larger errors in their estimates of the candidate's position on a ballot than for candidates about which the models have lots of data. There are a number of interesting potential avenues for addressing this issue. Standard techniques from machine learning for addressing the class-imbalance can be incorporated into the imputation-based approach, perhaps including approaches like dynamic subset selection [Gathercole and Ross, 1994; Doucette and Heywood, 2008], applied in Algorithm 2. Since such techniques can sometimes also improve training performance, the could be especially effective in this problem domain, given that the imputation-based approach sometimes trains a number of models that is quadratic in the number of candidates.

Chapters 5 and 6 considered a number of different learning approaches, but one approach that was not considered was the application of a meta-classification technique [Breiman, 1996; Freund and Schapire, 1997] to the imputation-based approach directly. Instead of training a single model to generate a single imputation of the ballots, several different models could be trained, each resulting in (via variation in the composition of their training sets) a different imputation and a different resulting decision. Some averaging over the different decisions output by the different models could provide more robust performance, and might also provide some increased resistance to manipulation (since the manipulator may be unable to influence every classifier effectively by using a single ballot).

The data used in the thesis for experimental validation in Chapter 5 and 7 is drawn from Preflib [Mattei and Walsh, 2013], but perhaps better data could be obtained, and then contributed back to the broader community. For example, the experiments with the **Veto** voting rule used data from elections that were conducted under rules in which the last preference on a ballot has minimal impact. Consequently, voters may not have given great thought to their final preferences. However, under **Veto**, the last preference has great importance, since it is the only one that influence the outcome. Running electoral experiments on a platform like Amazon's Mechanical Turk could perhaps allow for the collection of a large dataset under the **Veto** rule, with more representative values for voters' final preferences. Preflib now also contains a number of other electoral datasets for municipal elections that could be good sources of data to expand on the experiments conducted in the thesis.

Another advantage of gathering electoral data directly from controlled experiments is that it would be more effective to differentiate between a voter's ignorance and ambivalence using such data. The ranked ballot data contained in Preflib is reflective of most high stakes electoral systems, where voters are not permitted to indicate that they think two candidates are tied except perhaps by refusing to order them, which is indistinguishable from being unable to order them due to ignorance. In the context of the imputation-based approach, these two positions have dramatically different meaning. If two candidates are actually tied in the voter's view, then their ordering should not be imputed. In contrast, if the order of two candidates is unknown, the order *must* be imputed. Collecting data directly would allow voters to express whether they were unsure about the order of two candidates, or certain that they were equally good, and would reveal empirically how often voters are ignorant of the distinctions between two candidates. Interestingly, such experiments could be followed up in the manner proposed by Lau and Redlawsk [1997], by providing voters with much more targeted and useful information after their initial ballot, to produce datasets that had precise indications of which information was missing, and what the correct imputations would be (i.e. what voters' true preferences were).

Although the imputation-based approach to social choice is subject to Arrow's Theorem and the Gibbard-Satterthwaite Theorem, the original axiom sets were not quite ideal for the imputation-based approach, which is why new axioms were proposed specifically for the domain in Chapter 8. However, the results of the analysis in Chapter 8 suggest that there is a similar set of impossibility results lurking in this set of axioms, as all of the classifiers studied apart were either dictatorial, or violated several of the other axioms. Proving this would lend additional legitimacy to the new axioms, and would perhaps also expedite the analysis of other imputation algorithms.

The axiomatic analysis of additional imputation algorithms could provide further insights for practitioners into the appropriate algorithms to use in different situations. Logistic regression or another method based on features derived from voters' preferences would be a useful starting point, since the choice of features seems to exert significant influence over the behaviour of the algorithm. It would be interesting to show that certain features are required for a classification algorithm to satisfy a certain axiomatic property. It would also be interesting to do a more thorough analysis of combinations of voting systems and imputation algorithms, along the lines used by Nina Narodytska et al. [2012].

As discussed briefly in Chapter 9, there appears to be a close connection between the problem of designing strategy-proof imputation-based voting systems, and the problems studied in adversarial classification [Dalvi et al., 2004]. Exploring this topic more fully could provide an interesting avenue for making the imputation-based approach strategy-proof, that complements the approximation approach based on the work of Birrell and Pass [2011]. It would also be interesting to compare the two approaches empirically, and see which one offers a better approximation of the performance of the raw imputation-based approach.

On a more applied note, the Prefmine testbed described in Chapter 6 could be expanded to include more voting rules. In particular, adding Single Transferable Vote, quadratic scoring rules, and several other Condorcet extensions would improve the versatility of the system. The Prefmine user interface could also be improved, perhaps by integrating graph-generating software and other customized outputs, which would streamline the analysis of preliminary experiments. Providing integrated facilities for statistical analysis, and for viewing the data generated for different problems directly, would also improve the user experience. Further, Prefmine's parallelism features could be restored[2] to greatly boot performance. This would allow more ambitious and complex experiment designs to be constructed easily. The addition of a means to accept data in formats other than .soi would also broaden the appeal of the system.

The Prefmine system could be applied in the context of many practical applications. For example, it might be useful as a tool for medium-stakes voting, where many organizations still use poorly constructed or ad hoc systems. As an illustration, communities like Guelph, Ontario, have recently allocated funds for community development in part based on popular vote. Since candidate projects could be proposed by as few as three citizens, there were a very large number of alternatives, and partial ranked ballots were elicited from voters.

[2]Recall, they were disabled due to a change in the experimental programming language that was used for implementation.

However, the aggregation of the ballots was conducted using an ad-hoc variant of the **Borda** rule[3]. Although online systems exist to aid in deciding elections of this kind, such systems are not well suited to a medium-stakes application of this kind due to privacy concerns[4]. In contrast, Prefmine processes data locally, and has a pass-through option that would allow direct application of rules like Borda. Alternatively, with small modifications expanding on the `raw` option described in Chapter 6, Prefmine could accept partial ballots and output a decision directly using one or more of the techniques described in this thesis, rather than performing ablation experiments. This capability could be very useful to practitioners.

More speculatively, the imputation-based approach might be useful in multiagent coordination contests, like the recent contests to coordinate ad hoc teams in robotic soccer [Genter et al., 2015], as illustrated in the coordination example near the start of this chapter. A feature of the approach is that it allows members of a cooperative team to propose votes over arbitrary alternatives, and can assist the team in making decisions that are a reasonable approximation of the group's consensus even if not all group members have evaluated all alternatives.

Finally, a common question with reference to this work is whether the system could (or should) be deployed for higher stakes political contests, for example, the election of a national government. Although in principle the imputation-based approach could scale to a national level, political contests are a unique application domain, because the goal of a political contest is not necessarily to make the "best" or most representative decision. Rather, a good system for political contests must also convince voters (especially voters who do not like the outcome) that the outcome was selected in a fair and reasonable manner. An advantage of the **Plurality** system is that it is extremely simple and easy to explain to laypeople: the winner was the favourite candidate of the largest group. Voters who are displeased with the outcome have voted for a candidate that was favoured by a smaller group. As a proxy for other means of making the decision (e.g. force of arms), it is easy to understand and accept that the side with more supporters has defeated one's own side. Ranked ballot systems require more nuanced explanations. Voters must understand why a given candidate is the winner, and must be convinced that the system has made a reasonable choice. Clearly it is possible to convince voters of this, as even fairly complex systems like **Single Transferable Vote** have been adopted at the national scale in countries like Australia and Ireland. Iceland and the tiny island nation of Nauru

[3]The author learned this via private correspondence with one of the members of the committee that allocated funds.

[4]Voters are often uncomfortable with the idea that their private ballots might be given to a third-party website.

use variants of **Borda**. However, more complex systems like Schulze's method [Schulze, 2011] have not yet been adopted by any national assembly. Under this method, victory involves reasoning over paths between candidates in a weighted graph. Explaining such a system to laypeople, and convincing them that its decisions are fair and correct, is not a straightforward proposition, even if the system has been readily adopted by technically savvy societies like the Debian Project, an organization primarily comprised of software engineers. Even much simpler systems have failed to be explained adequately to the public. In Ontario, which uses district-level **Plurality** to elect its provincial legislature, a 2007 referendum to elect some of the members via an exceedingly simple provincial-level proportionate scheme (which did not even require the use of a ranked ballot), failed to be adopted, with one of the most common reasons given by voters being a lack of information about how the system would work [LeDuc et al., 2008]. Explaining the imputation-based approach, and its advantages, to a large group of laypeople seems unlikely to succeed, which precludes the use of the system in large scale, high stakes, applications, even if it were found to make high quality decisions there. It might be interesting to design variations on the imputation-based approach that were more easily explained to voters. For example, a multi-round protocol where voters cast initial ballots, and then are shown the imputation of their ballot that the system would utilize, may be effective. Voters could then revise the imputation of their own ballot before the system made a final decision. In this way, the imputation-based approach would support voters by providing them with more information, rather than deciding the outcome for them directly. Although such a system might be more palatable to voters, it has other practical barriers, like the need to re-identify voters between rounds. Further, some voters are inherently distrustful of having a computer recommend a completion of their ballots, and might refuse to accept any suggestions, even if these were in fact in their best interest.

10.5 Summary

At the onset, this thesis proposed to address three distinct subproblems, a goal which has now been achieved. A new vision for resolving social choice problems in the face of incomplete information was provided in Chapters 3 and 4, using techniques from machine learning to impute missing information in voters' ballots, and demonstrating that machine learning can indeed be applied to improve the quality of social choices.

A detailed validation of the proposed approach on real world data in Chapter 5 established that it was highly effective, meeting or exceeding the performance of state-of-the-art competitors, and the flexible testbed system developed in Chapter 6 helps practitioners, as well as researchers in computational social choice, to adopt the same methodology in the future. The purpose-built algorithms of Chapter 7 further refine performance, offering improved speed, decision-making quality, and interpretability for this problem domain, supported by theoretical guarantees and empirical results.

Finally, the theoretical results in Chapter 4 served to connect the fields of machine learning and computational social choice together, providing an exciting and fruitful area for future theoretical work from the perspective of both disciplines. Results in Chapter 8 showed that refined axioms could be developed specifically for the imputation-based approach, and provided an interesting initial look at the effects of combining different imputation algorithms and social choice functions together, in terms of axiomatic fairness. Chapter 9 considered practical ways to improve the resistance of the imputation-based approach to strategic behaviours, providing a general solution that works for any combination of voting rule and imputation method that is known to be vulnerable to manipulation in the first place.

In addition, the results in this chapter discussed applications for the system, and the powerful advantage it can offer in more structured preference domains. Future application areas await in a variety of medium-stakes areas, ranging from community decision making, through to robot coordination. The thesis also leaves open the question of how fairer imputation algorithms could be designed, and this provides an exciting avenue for future work, both through axiomatic analysis, and empirical validation.

In all, the thesis serves to advance the state of the art in the domain of group decisions with incomplete information, integrating machine learning, and providing insights into the nature of fair decision making for practitioners, voters, and researchers alike.

References

David J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 295–304. ACM, 2007. ISBN 1-59593-653-X.

Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine learning: ECML 2004*, pages 39–50. Springer, 2004. ISBN 3-540-23105-6.

Alnur Ali and Marina Meilă. Experiments with Kemeny ranking: What works when? *Mathematical Social Sciences*, 64(1):28–40, 2012.

Horst Alzer. On some inequalities for the gamma and psi functions. *Mathematics of Computation of the American Mathematical Society*, 66(217):373–389, 1997.

Kenneth J. Arrow. *Social choice and individual values*. Yale University Press, 12th edition, 2012. ISBN 0-300-18698-3.

Kenneth J. Arrow, Amartya Sen, and Kotaro Suzumura. *Handbook of Social Choice & Welfare*. Elsevier, 2nd edition, 2010. ISBN 0-08-092982-6.

Hossein Azari, David Parkes, and Lirong Xia. Random Utility Theory for Social Choice. In *Advances in Neural Information Processing Systems*, pages 126–134. NIPS Foundation, 2012.

Hossein Azari Soufiani, David C. Parkes, and Lirong Xia. Preference Elicitation For General Random Utility Models. In *Uncertainty in Artificial Intelligence: Proceedings of the 29th Conference*. AUAI Press, 2013. ISBN 0-9749039-9-X.

Alexander Balz and Robin Senge. WEKA-LR: A Label Ranking Extension for WEKA. URL http://www.uni-marburg.de/fb12/kebi/research/software/labelrankdoc.pdf.

John Bartholdi III, Craig A. Tovey, and Michael A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and welfare*, 6(2):157–165, 1989.

John J. Bartholdi III and James B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

Dorothea Baumeister and Jörg Rothe. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Information Processing Letters*, 112(5): 186–190, 2012.

Boris Bazhanov. *Memoirs of Stalin's former secretary*. Third Wave, 1992. Near the end of Chapter 5.

Bernard Berelson, Paul Felix Lazarsfeld, William N. McPhee, Paul Felix Lazarsfeld, and Paul Felix Lazarsfeld. *Voting: A Study of Opinion Formation in a Presidential Campaign*. University of Chicago Press, 1954.

Eleanor Birrell and Rafael Pass. Approximately Strategy-Proof Voting. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 67–72. AAAI Press, 2011.

Duncan Black. On Arrow's impossibility theorem. *The Journal of Law & Economics*, 12 (2):227–248, 1969.

Duncan Black, Robert Albert Newing, Iain McLean, Alistair McMillan, and Burt L. Monroe. *The theory of committees and elections*. Springer, 2nd edition, 1958. ISBN 0-521-04262-3.

Carlo E. Bonferroni. *Teoria statistica delle classi e calcolo delle probabilita*. Libreria internazionale Seeber, 1936.

S Bouveret. Whale3 - WHich ALternative is Elected. URL `http://whale3.noiraudes.net/whale3/index.do`.

Felix Brandt, Guillaume Chabin, and Christian Geist. Pnyx:: A Powerful and User-friendly Tool for Preference Aggregation. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1915–1916. International Foundation for Autonomous Agents and Multiagent Systems, 2015a. ISBN 1-4503-3413-X.

Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia. *Handbook of Computational Social Choice*. Cambridge University Press, 2015b.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Simina Brânzei, Ioannis Caragiannis, Jamie Morgenstern, and Ariel Procaccia. How Bad is Selfish Voting? In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 138–144. AAAI Press, 2013.

Dmitri Burago, Yuri Burago, and Sergei Ivanov. *A course in metric geometry*, volume 33. American Mathematical Society Providence, 2001. ISBN 0-8218-2129-6.

Bruce E. Cain. Strategic voting in Britain. *American Journal of Political Science*, 22(3): 639–655, 1978.

Angus Campbell, Philip Converse, Warren Miller, and Donald Stokes. *Elections and the political order.* Wiley New York, 1966.

Ioannis Caragiannis, Ariel D. Procaccia, and Nisarg Shah. When do noisy votes reveal the truth? In *Proceedings of the fourteenth ACM conference on electronic commerce*, pages 143–160. ACM Press, 2013. ISBN 1-4503-1962-9.

Ioannis Caragiannis, Ariel D. Procaccia, and Nisarg Shah. Modal ranking: A uniquely robust voting rule. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 616–622. AAAI Press, 2014.

Thomas Carlyle. *Sartor Resartus: The Life and Opinions of Herr Teufelsdröckh...* Chapman and Hall, 1893.

Francois Caron, Yee Whye Teh, and Thomas Brendan Murphy. Bayesian nonparametric Plackett–Luce models for the analysis of preferences for college degree programmes. *The Annals of Applied Statistics*, 8(2):1145–1181, 2014.

Michael X. Delli Carpini and Scott Keeter. *What Americans know about politics and why it matters.* Yale University Press, 1997. ISBN 0-300-07275-9.

John M. Chambers. *Graphical methods for data analysis.* Springer, 1983. ISBN 0-412-05271-7.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27–66, 2011.

Günther Charwat and Andreas Pfandler. Democratix: A Declarative Approach to Winner Determination. In *Algorithmic Decision Theory*, pages 253–269. Springer, 2015. ISBN 3-319-23113-8.

Weiwei Cheng, Eyke Hüllermeier, and Krzysztof J. Dembczynski. Label ranking methods based on the Plackett-Luce model. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 215–222, 2010.

Winston Churchill. The Bright Gleam of Victory, A Speech at the Lord Mayor's Day Luncheon, November 1942. URL http://www.winstonchurchill.org/resources/speeches/1941-1945-war-leader/987-the-end-of-the-beginning.

Vincent Conitzer. The maximum likelihood approach to voting on social networks. In *Proceedings of the Fifty-First Annual Allerton Conference on Communication, Control, and Computing*, pages 1482–1487. IEEE, 2013. ISBN 1-4799-3409-7.

Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 627–634. AAAI Press, 2006.

Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 109–115. AAAI Press, 2009.

Philip E. Converse. The Nature of Belief Systems in Mass Publics. In *Ideology and Discontent (ed. David Apter)*. New York Free Press, 1964.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

David R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, and Deepak Verma. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108, 2004.

Jessica Davies, George Katsirelos, Nina Narodytska, and Toby Walsh. Complexity of and algorithms for borda manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 657–662. AAAI Press, 2011.

Jessica Davies, Nina Narodytska, and Toby Walsh. Eliminating the Weakest Link: Making Manipulation Intractable? In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1333–1339. AAAI Press, 2012.

Jean-Charles de Borda. *Mémoire sur les élections au scrutin.* Baudouin, 1781.

Marie Jean Antoine Nicolas de Caritat. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix.* L'imprimerie royale, 1785.

Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Persi Diaconis and Ronald L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 262–268, 1977.

John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Price of fairness in kidney exchange. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1013–1020. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

Franz Dietrich and Kai Spiekermann. Independent opinions? On the causal foundations of belief formation and Jury Theorems. *Mind*, 122(487):655–685, 2013.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, pages 263–286, 1995.

Evgenia Dimitriadou, Kurt Hornik, Friedrich Leisch, David Meyer, and Andreas Weingessel. Misc functions of the Department of Statistics (e1071), TU Wien. *R package*, 1: 5–24, 2008.

Shahar Dobzinski and Ariel D. Procaccia. Frequent manipulability of elections: The case of two voters. In *Internet and Network Economics*, pages 653–664. Springer, 2008.

Paul Dolan, Rebecca Shaw, Aki Tsuchiya, and Alan Williams. QALY maximisation and people's preferences: a methodological review of the literature. *Health economics*, 14(2): 197–208, 2005.

John Doucette and Malcolm I. Heywood. GP classification under imbalanced data sets: active sub-sampling and AUC approximation. In *Genetic Programming*, pages 266–277. Springer, 2008. ISBN 3-540-78670-8.

John A. Doucette, Kate Larson, and Robin Cohen. Conventional machine learning for social choice. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, 2015.

Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

George H. Dunteman. *Principal components analysis*. Number 69 in Quantitative Applications in the Social Sciences. Sage, 1989. ISBN 0-8039-3104-2.

Piotr Faliszewski and Ariel D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

Piotr Faliszewski, Edith Hemaspaandra, and Henning Schnoor. Copeland voting: Ties matter. In *Proceedings of the 7th international joint conference on autonomous agents and multiagent systems-Volume 2*, pages 983–990. International Foundation for Autonomous Agents and Multiagent Systems, 2008. ISBN 0-9817381-1-7.

Piotr Faliszewski, Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 118–127. ACM Press, 2009. ISBN 1-60558-560-2.

Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Using complexity to protect elections. *Communications of the ACM*, 53(11):74–82, 2010.

Vincent E. Farrugia, Héctor P. Martínez, and Georgios N. Yannakakis. The preference learning toolbox. *arXiv preprint arXiv:1506.01709*, 2015.

Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. In *Advances in Neural Information Processing Systems*, pages 253–261. NIPS Foundation, 2014.

Aris Filos-Ratsikas and Peter Bro Miltersen. Truthful approximations to range voting. In *Web and Internet Economics*, pages 175–188. Springer, 2014. ISBN 3-319-13128-1.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

Zack Fitzsimmons and Edith Hemaspaandra. Complexity of manipulative actions when voting with ties. In *Algorithmic Decision Theory*, pages 103–119. Springer, 2015. ISBN 3-319-23113-8.

D. Foley. The strange history of the economic agent. *New School Economic Review*, 1(1): 82–94, 2004.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Ehud Friedgut, Gil Kalai, and Noam Nisan. Elections can be manipulated often. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 243–249. IEEE, 2008. ISBN 0-7695-3436-8.

Sean Gailmard. *Arrow's theorem on single-peaked domains*. PhD thesis, Department of Government & Institute for Quantitative Social Science, Harvard University, 2008.

Chris Gathercole and Peter Ross. Dynamic training subset selection for supervised learning in genetic programming. In *Parallel Problem Solving from Nature—PPSN III*, pages 312–321. Springer, 1994. ISBN 3-540-58484-6.

John Geanakoplos. Three brief proofs of Arrow's impossibility theorem. *Economic Theory*, 26(1):211–215, 2005.

Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The Potsdam answer set solving collection. *AI Communications*, 24(2):107–124, 2011.

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):365–385, 1991.

Katie Genter, Tim Laue, and Peter Stone. The RoboCup 2014 SPL Drop-in Player Competition: Encouraging Teamwork without Pre-coordination. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1745–1746. International Foundation for Autonomous Agents and Multiagent Systems, 2015. ISBN 1-4503-3413-X.

Karen Gerard and Gavin Mooney. QALY league tables: handle with care. *Health economics*, 2(1):59–64, 1993.

Allan Gibbard. Manipulation of voting schemes: a general result. *Econometrica: Journal of the Econometric Society*, 41(4):587–601, 1973.

Allan Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica: Journal of the Econometric Society*, 45(3):665–681, 1977.

Jonathan Goldman and Ariel D. Procaccia. Spliddit: Unleashing fair division algorithms. *ACM SIGecom Exchanges*, 13(2):41–46, 2015.

Gene H. Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

Umberto Grandi, Andrea Loreggia, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Restricted manipulation in iterative voting: Condorcet efficiency and Borda score. In *Algorithmic Decision Theory*, pages 181–192. Springer, 2013. ISBN 3-642-41574-1.

Scott D. Grosse. Assessing cost-effectiveness in healthcare: history of the $50,000 per QALY threshold. *Expert Review of Pharmacoeconomics & Outcomes Research*, 8(2): 165–178, 2008.

John Guiver and Edward Snelson. Bayesian inference for Plackett-Luce ranking models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 377–384. ACM, 2009. ISBN 1-60558-516-5.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

Les Hatton. The T-experiments: errors in scientific software. In *Quality of Numerical Software*, pages 12–31. Springer, 1997. ISBN 1-5041-2942-3.

Les Hatton and Andy Roberts. How accurate is scientific software? *IEEE Transactions on Software Engineering*, 20(10):785–797, 1994.

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

Magnus R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems1. *Journal of Research of the National Bureau of Standards*, 49(6), 1952.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Jerry L. Hintze and Ray D. Nelson. Violin plots: a box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

Thomas Hobbes. *Leviathan: Or the Matter, Forme and Power of a Commonwealth, Ecclesiasticall and Civil.* Yale University Press, 21st edition, 1900. ISBN 0-300-16318-5.

William Hosch. Machine Learning. *Encyclopedia Britannica, Online Edition*, 2016. URL http://www.britannica.com/technology/machine-learning.

David Hume. *A treatise of human nature*. Dover Publications, 2nd dover edition, 1739. ISBN 0-486-43250-5.

Gunter Hägele and Friedrich Pukelsheim. Llull's writings on electoral systems. *Studia Lulliana*, 41(97):3–38, 2001.

Eyke Hüllermeier and Johannes Fürnkranz. Learning from label preferences. In *Proceedings of the 14th International Conference on Discovery Science*, pages 2–17. Springer, 2011. ISBN 3-642-24476-9.

Marcus Isaksson, Guy Kindler, and Elchanan Mossel. The geometry of manipulation—a quantitative proof of the Gibbard-Satterthwaite theorem. *Combinatorica*, 32(2):221–250, 2012.

William James. *The will to believe and other essays in popular philosophy*. Harvard University Press, 6th edition, 1899. ISBN 0-674-95281-2.

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Albert Xin Jiang, Manish Jain, and Milind Tambe. Computational Game Theory for Security and Sustainability. *Journal of Information Processing*, 22(2):176–185, 2014.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM, 2002. ISBN 1-58113-567-X.

Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–226. ACM, 2006. ISBN 1-59593-339-5.

Toshihiro Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 583–588. ACM, 2003. ISBN 1-58113-737-0.

Diane Kelly and Rebecca Sanders. The challenge of testing scientific software. In *In Proceedings of the 2008 Conference for the Association for Software Testing*, pages 30–36. AST, 2008.

John G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

Maurice G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Josef Kittler, Mohamad Hatef, Robert PW Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

Stefan Knerr, Léon Personnaz, and Gérard Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990. ISBN 3-642-76155-0.

Donald E. Knuth. *Art of Computer Programming, Volume 4, Fascicle 4, The: Generating All Trees–History of Combinatorial Generation*. Addison-Wesley Professional, 2013. ISBN 0-13-270234-7.

Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Presentation at the IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, AAAI Press, 2005.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Jérôme Lang, Maria Silvia Pini, Francesca Rossi, Domenico Salvagnin, Kristen Brent Venable, and Toby Walsh. Winner determination in voting trees with incomplete preferences and weighted votes. *Autonomous Agents and Multi-Agent Systems*, 25(1):130–157, 2012.

Richard R. Lau and David P. Redlawsk. Voting correctly. *American Political Science Review*, 91(03):585–598, 1997.

Richard R. Lau, Parina Patel, Dalia F. Fahmy, and Robert R. Kaufman. Correct voting across thirty-three democracies: A preliminary analysis. *British Journal of Political Science*, 44(02):239–259, 2014.

Lawrence LeDuc, Heather Bastedo, and Catherine Baquero. The Quiet Referendum: Why Electoral Reform Failed in Ontario. In *Prepared for the annual meeting of the Canadian Political Science Association. University of British Columbia, June*, pages 4–6, 2008.

David T. Lee. Efficient, private, and epsilon-strategyproof elicitation of tournament voting rules. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2026–2032. AAAI Press, 2015. ISBN 1-57735-738-8.

Omer Lev and Jeffrey S. Rosenschein. Convergence of iterative voting. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 611–618. International Foundation for Autonomous Agents and Multiagent Systems, 2012. ISBN 0-9817381-2-5.

D. V. Lindley. Bayesian analysis in regression problems. *Bayesian statistics, DL Meyer and RO Collier, eds., Peacock publishers*, 1970.

Tyler Lu and Craig Boutilier. Learning Mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 145–152, 2011a.

Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 287–213. AAAI Press, 2011b. ISBN 1045-0823.

Tyler Lu and Craig Boutilier. Multi-Winner Social Choice with Incomplete Preferences. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 263–270. AAAI Press, 2013.

Tyler Lu, Pingzhong Tang, Ariel D. Procaccia, and Craig Boutilier. Bayesian vote manipulation: Optimal strategies and impact on welfare. *arXiv preprint arXiv:1210.4895*, 2012.

R. Duncan Luce. *Individual choice behavior: A theoretical analysis*. Wiley, 1959. ISBN 0-486-44136-9.

Colin L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1/2):114–130, 1957.

Nicholas Mattei and Toby Walsh. Preflib: A library for preferences http://www. preflib. org. In *Algorithmic Decision Theory*, pages 259–270. Springer, 2013. ISBN 3-642-41574-1.

Nicholas Matti. PrefLib-Tools: A small and lightweight set of Python tools for working with and generating data from www.PrefLib.org. URL `https://github.com/nmattei/PrefLib-Tools`.

Kenneth O. May. A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica: Journal of the Econometric Society*, pages 680–684, 1952.

Peter McCullagh and John A. Nelder. *Generalized linear models*. CRC press, 37th edition, 1989. ISBN 0-412-31760-5.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103. IEEE, 2007. ISBN 0-7695-3010-9.

Marina Melia and Harr Chen. Dirichlet process mixtures of generalized mallows models. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence (UAI2010)*, 2010.

Vijay Menon and Kate Larson. Reinstating Combinatorial Protections for Manipulation and Bribery in Single-Peaked and Nearly Single-Peaked Electorates. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 565–571. AAAI Press, 2016.

John Stuart Mill. *On liberty*. Broadview Press, broadview literary texts edition, 1999. ISBN 1-55111-199-3.

John M. Miyamoto and Stephen A. Eraker. Parameter estimates for a QALY utility model. *Medical Decision Making: An International Journal of the Society for Medical Decision Making*, 5(2):191–213, 1984.

Hervé Moulin. On strategy-proofness and single peakedness. *Public Choice*, 35(4):437–455, 1980.

Nina Narodytska and Toby Walsh. The computational impact of partial votes on strategic voting. In *The Proceedings of the 21st European Conference on Artificial Intelligence*. IOS Press, 2014.

CBC News. 'It hurts': NDP shut out of downtown Toronto in Liberal crush - Toronto - CBC News, October 2015. URL http://www.cbc.ca/news/canada/toronto/liberals-ndp-toronto-danforth-1.3279370.

Nina Narodytska, Toby Walsh, and Lirong Xia. Combining voting rules together. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI)*, 2012.

Erik Nord. The QALY—a measure of social value rather than individual utility? *Health Economics*, 3(2):89–93, 1994.

Liberal Party of Canada. Real Change — A New Plan for a Strong Middle Class (Liberal Party Platform), 2015. URL https://www.liberal.ca/files/2015/10/A-new-plan-for-a-strong-middle-class-BW-1.pdf.

Stephen Petrick. Strategic voting hurt NDP: Cassidy. *Inside Belleville*, October 2015. URL http://www.insidebelleville.com/news-story/5968523-strategic-voting-hurt-ndp-cassidy/.

Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Incompleteness and Incomparability in Preference Aggregation. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 1464–1469. AAAI Press, 2007.

Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Aggregating partially ordered preferences. *Journal of Logic and Computation*, 19(3):475–502, 2009.

Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Incompleteness and incomparability in preference aggregation: Complexity results. *Artificial Intelligence*, 175(7):1272–1289, 2011.

James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Robin L. Plackett. The analysis of permutations. *Applied Statistics*, 24(2):193–202, 1975.

John C. Platt, Nello Cristianini, and John Shawe-Taylor. Large Margin DAGs for Multiclass Classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553, 1999.

Karl Popper. *The logic of scientific discovery.* Routledge, 2nd edition, 2005. ISBN 1-134-47002-9.

W. H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical recipes 3rd edition: The art of scientific computing.* Cambridge University Press, 2007. ISBN 0-521-88068-8.

Ariel D. Procaccia. Can Approximation Circumvent Gibbard-Satterthwaite? In *In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 836–841, 2010.

Ariel D. Procaccia and Jeffrey S. Rosenschein. Average-case tractability of manipulation in voting via the fraction of manipulators. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 718–720. IFAAMAS, 2007.

Ariel D. Procaccia and Nisarg Shah. Optimal Aggregation of Uncertain Preferences. *Working Paper, Computer Science Department, Carnegie Mellon University*, 2015.

Ariel D. Procaccia, Nisarg Shah, and Yair Zick. Voting rules as error-correcting codes. *Artificial Intelligence*, 231:1–16, 2016.

Charles P. Quesenberry and D. C. Hurst. Large sample simultaneous confidence intervals for multinomial proportions. *Technometrics*, 6(2):191–195, 1964.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J. Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014. ISBN 0-08-050058-7.

Zinovi Rabinovich, Svetlana Obraztsova, Omer Lev, Evangelos Markakis, and Jeffrey S. Rosenschein. Analysis of Equilibria in Iterative Voting Schemes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1007–1013, 2015.

Alexander Reffgen. Generalizing the Gibbard–Satterthwaite theorem: partial preferences, the degree of manipulation, and multi-valuedness. *Social Choice and Welfare*, 37(1): 39–59, June 2011. ISSN 0176-1714.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

Brian Ripley and W. Venables. nnet: Feed-forward neural networks and multinomial log-linear models. *R package version*, 7(5), 2011.

P. Romanski. FSelector: Selecting attributes. *Vienna: R Foundation for Statistical Computing*, 2009.

Jean-Jacques Rousseau. *The Social Contract: & Discourses*. JM Dent & Sons, 1920.

Donald Rumsfeld. Defense.gov Transcript: DoD News Briefing - Secretary Rumsfeld and Gen. Myers, February 12, 11:30 AM EST. 2002. URL http://archive.defense.gov/Transcripts/Transcript.aspx?TranscriptID=2636.

Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Prentice Hall, Englewood Cliffs, NJ, USA, 1995.

Paul A. Samuelson. Reaffirming the existence of "reasonable" Bergson-Samuelson social welfare functions. *Economica*, 44(173):81–88, 1977.

Mark Allen Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

Joseph L. Schafer. Multiple imputation: a primer. *Statistical Methods in Medical Research*, 8(1):3–15, 1999.

Markus Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303, 2011.

Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2008. ISBN 1-139-47524-X.

Charles Spearman. 'Footrule'for measuring correlation. *British Journal of Psychology, 1904-1920*, 2(1):89–108, 1906.

John E. Stone, David Gohara, and Guochun Shi. OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in Science & Engineering*, 12 (1-3):66–73, 2010.

R. Core Team. *R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2013.* ISBN 3-900051-07-0, 2014.

Andrey Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Doklady*, 5:1035–1038, 1963.

Alan Tsang and Kate Larson. The Echo Chamber: Strategic Voting and Homophily in Social Networks. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 368–375, 2016.

Alan Tsang, John A. Doucette, and Hadi Hosseini. Voting with social influence: Using arguments to uncover ground truth. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1841–1842. International Foundation for Autonomous Agents and Multiagent Systems, 2015. ISBN 1-4503-3413-X.

Toby Walsh. Uncertainty in preference elicitation and aggregation. In *Proceedings of the 22nd national conference on Artificial intelligence*, pages 3–8. AAAI Press, 2007.

Toby Walsh. Where are the really hard manipulation problems? the phase transition in manipulating the veto rule. In *Proceedings of the Twenty-First International Joint Conference on Artifical intelligence*, pages 324–329. Morgan Kaufmann Publishers Inc., 2009.

Toby Walsh. Is computational complexity a barrier to manipulation? *Annals of Mathematics and Artificial Intelligence*, 62(1-2):7–26, 2011.

Toby Walsh and Lirong Xia. Lot-based voting rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 603–610. International Foundation for Autonomous Agents and Multiagent Systems, 2012. ISBN 0-9817381-2-5.

Daniel Westlake. Strategic voting has long-term costs for progressives. *National Post*, October 2015. URL http://news.nationalpost.com/full-comment/daniel-westlake-strategic-voting-has-long-term-costs.

David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

David H. Wolpert. What the no free lunch theorems really mean; how to improve search algorithms. In *Santa fe Institute Working Paper*. 2012.

David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

Douglas R. Woodall. An impossibility theorem for electoral systems. *Discrete Mathematics*, 66(1):209–211, 1987.

Douglas R. Woodall. Properties of preferential election rules. *Voting Matters*, 3:8–15, 1994.

Lirong Xia. Generalized Decision Scoring Rules: Statistical, Computational, and Axiomatic Properties. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 661–678. ACM, 2015. ISBN 1-4503-3410-5.

Lirong Xia and Vincent Conitzer. Determining Possible and Necessary Winners under Common Voting Rules Given Partial Orders. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 196–201. AAAI Press, 2008.

Lirong Xia and Vincent Conitzer. A maximum likelihood approach towards aggregating partial orders. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 446–451. AAAI Press, 2011. ISBN 1045-0823.

Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.

Dong Yu, Geoffrey Hinton, Nigel Morgan, Jen-Tzung Chien, and Shigeki Sagayama. Introduction to the special section on deep learning for speech and language processing. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):4–6, 2012.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008. ISBN 3-540-68865-X.

Slavoj Žižek. What Rumsfeld Doesn't Know That He Knows About Abu Ghraib. *In These Times*, 21, 2004.