Machine Learning Techniques and Stochastic Modeling in Mathematical Oncology

by

Brydon Eastman

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Applied Mathematics

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Hermann Eberl Professor, Dept. of Math & Stats, University of Guelph
Supervisor(s):	Mohammad Kohandel Associate Professor, Dept. of Applied Math, Univ. of Waterloo
Internal Member(s):	Sivabal Sivaloganathan Chair, Dept. of Applied Math, University of Waterloo
	Sander Rhebergen Associate Professor, Dept. of Applied Math, Univ. of Waterloo
Internal-External Member:	Pascal Poupart Professor, Dept. of Computer Science, University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The cancer stem cell hypothesis claims that tumor growth and progression are driven by a (typically) small niche of the total cancer cell population called cancer stem cells (CSCs). These CSCs can go through symmetric or asymmetric divisions to differentiate into specialised, progenitor cells or reproduce new CSCs. While it was once held that this differentiation pathway was unidirectional, recent research has demonstrated that differentiated cells are more plastic than initially considered. In particular, differentiated cells can de-differentiate and recover their stem-like capacity. Two recent papers have considered how this rate of plasticity affects the evolutionary dynamic of an invasive, malignant population of stem cells and differentiated cells into existing tissue [64, 109]. These papers arrive at seemingly opposing conclusions, one claiming that increased plasticity results in increased invasive potential, and the other that increased plasticity decreases invasive potential. Here, we show that what is most important, when determining the effect on invasive potential, is how one distributes this increased plasticity between the compartments of resident and mutant-type cells. We also demonstrate how these results vary, producing non-monotone fixation probability curves, as inter-compartmental plasticity changes when differentiated cell compartments are allowed to continue proliferating, highlighting a fundamental difference between the two models. We conclude by demonstrating the stability of these qualitative results over various parameter ranges.

Imaging flow cytometry is a tool that uses the high-throughput capabilities of conventional flow cytometry for the purposes of producing single cell images. We demonstrate the label free prediction of mitotic cell cycle phases in Jurkat cells by utilizing brightfield and darkfield images from an imaging flow cytometer. The method is a non destructive method that relies upon images only and does not introduce (potentially confounding) dies or biomarkers to the cell cycles. By utilizing deep convolutional neural networks regularized by generated, synthetic images in the presence of severe class imbalance we are able to produce an estimator that outperforms the previous state of the art on the dataset by 10-15%.

The in-silico development of a chemotherapeutic dosing schedule for treating cancer relies upon a parameterization of a particular tumour growth model to describe the dynamics of the cancer in response to the dose of the drug. In practice, it is often prohibitively difficult to ensure the validity of patient-specific parameterizations of these models for any particular patient. As a result, sensitivities to these particular parameters can result in therapeutic dosing schedules that are optimal in principle not performing well on particular patients. In this study, we demonstrate that chemotherapeutic dosing strategies learned via reinforcement learning methods are more robust to perturbations in patient-specific

parameter values than those learned via classical optimal control methods. By training a reinforcement learning agent on mean-value parameters and allowing the agent periodic access to a more easily measurable metric, relative bone marrow density, for the purpose of optimizing dose schedule while reducing drug toxicity, we are able to develop drug dosing schedules that outperform schedules learned via classical optimal control methods, even when such methods are allowed to leverage the same bone marrow measurements.

Acknowledgements

I would like to begin by thanking everyone who has made me coffee over the years, this work is at least partly yours.

To my undergraduate instructors: it is hard for me to think of any individuals who have had a larger impact on my life trajectory than you. I entered undergraduate in a fragile and unsure state and was provided with not only instruction and direction but hope and encouragement: commodities that are all too rare in academia. To Dr. Kevin Vander Meulen, you have impacted me and my career far more than you can ever know. You are the very first person who made me believe that I could be a mathematician and without your initial push it is unlikely that I ever would have pursued graduate school. Thank you for introducing me to this part of my identity. You not only inspired an unsure student and assured me I belonged, but you also taught me the dangers of these "hymns of self-praise"; an exceptionally difficult balancing act that I reflect on often. To Dr. Derek Schuurman, the intention that you bring to your teaching and the care that you bring to your students have made a lasting impression on me: both as a student who desperately needed these qualities and as an educator desperately trying to emulate them. To Dr. Kyle Spyksma and Dr. Darren Brouwer, thank you for nurturing a love of applied mathematics in me that seems unlikely to burn out.

To my masters' supervisor, Dr. Gail Wolkowicz, your many insightful conversations over the years have shaped the minutiae of my work immeasurably but your commitment to mentorship is what has made the largest impression. To Samantha, Theodore, Maddison, and Anjka: thank you for keeping me sane at McMaster, sorry you cannot be the star child.

To the Math Medicine Research Group: Moriah, Cameron, Michelle, Farinaz, Nafiseh, Burak, Somayeh, Dorsah, Emilee, and others: I could not even begin to thank you all enough. From late night zoom calls, weekly group meetings, assistance with experiments, to coursework: you all have been a joy to work with over the past few years. I have learned much from you all and hope that I have returned the favour at least in part.

To my supervisor, Dr. Mohammad Kohandel, your direction for my studies and thesis work has shaped this document in innumerable ways; there are not words enough to thank you.

And finally, to my dear and respected colleague Dr. Kyle Lowry: by putting up 26 and 10 in a finals elimination game you taught me, above all else, hope.

Dedication

I dedicate this thesis to my wife Tabitha without whom this surely would have been completed far sooner and with much less joy.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ANN Artificial Neural Network xvi, 61, 64, 67, 68, 72, 73

APOPT Advanced Process Optimizer 87, 93, 96, 122

CART Classification and Regression Trees 11

CNN Convolutional Neural Network xvi, 18, 26, 59, 62, 67, 69–73

CSCs Cancer Stem Cells iv, 3, 104

CUT Contrastive Unpaired Translation 58, 59

DDQN Deep Double Q Learning 23, 24

DTC Decision Tree Classifier 61, 63, 64, 72, 73

GAN Generative Adversarial Network xiii, 18–21, 58, 59

IPOPT Interior Point Optimizer xviii, 124

KNN k-Nearest Neighbours xvi, 61, 64, 69, 70

MNIST Modified National Institute of Standards and Technology database 21

MSE Mean Squared Error 16

NTNOC Nearest Training Neighbour Optimal Control xviii, 85, 98, 100, 101, 122, 125, 127

ODE Ordinary Differential Equation 106

ReLU Rectified Linear Unit 15, 17, 61, 62, 67

RL Reinforcement Learning 22, 23, 106

SDE Stochastic Differential Equation 7, 77

Chapter 1

Introduction

Machine Learning and Neural Networks have a long history of use in the mathematical sciences, for instance Simon Haykin's classic book on neural networks [43] was originally published in 1994. Over the last few decades, the applicability of learning machines has grown substantially. As a result of improvements to consumer computer hardware and advancements in programming techniques, training large scale neural networks is now feasible without the need for overly advanced equipment. This research proposal is focused on the application of this mathematical tool to a particular biological area. In particular, we wish to investigate a few ways that machine learning in the context of artificial neural networks can be leveraged in mathematical oncology.

Mathematical Oncology as a discipline is concerned with applying various mathematical techniques to describe, model, and predict the behaviour of various biological phenomena under the umbrella of oncology. To mention but a few applications, mathematical oncology can be focused on the use of ordinary and partial differential equations for the purposes of describing behaviours of cancer cells [109], stochastic processes describing invasion of mutant type stem cells [58], models of the evolution of solid tumours [33, 83], or the optimisation of chemotherapeutic delivery [113]. The end goal of these applications may be to gain predictive insights that can be leveraged in the clinical community or to replicate observed results with phenomenological models in order to better understand the key components that drive biological problems. As such, the discipline of mathematical oncology is focused on using any applicable mathematical tool to describe, or understand, any number of complex biological behaviours.

This thesis will focus on a select handful of applications of neural networks and stochastic modeling techniques applied to particular problems in mathematical oncology. In particular,

leveraging the strengths of neural networks as universal function approximators for image recognition techniques in imaging flow cytometry, using deep neural networks to approximate valuation functions in optimising the schedules of cancer treatment plans, and using stochastic models to investigate invasion and metastasis in models of invasive tumours.

1.1 Cancer Biology Background

Cancer is a complicated family of diseases that share certain key characteristics [40, 41, 39]. The hallmarks of cancer include (1) evading growth suppressors, (2) avoiding immune destruction, (3) enabling replicative immortality, (4) tumour promoting inflammation, (5) activating invasion and metastasis, (6) inducing angiogenesis, (7) genome instability and mutation, (8) evading cell death, (9) deregulating cellular energies, and (10) sustaining proliferative signalling. Moreover, in [39] Hannahan goes on to denote four emerging hallmarks and enabling characteristics: unlocking phenotypic plasticity, nonmutational epigenetic programming, polymorphic microbiomes, and the presence of senescent cells.

Hence while cancers themselves differ quite a bit from one another in how they present, cancer cells have been observed to exhibit characteristics from the set described in the above hallmarks. Importantly it is the replicative immortality, genome instability, and phenotypic plasticity hallmarks that motivate the subject of study in Chapter 3 when we investigate the invasive potential of mutant stem cells as a function of phenotypic plasticity. Not only is there heterogeneity in the way in which differing cancer cells behave, there is also considerable inter-patient variability in response to treatment. This can be due to age, gender, etc [69]. This inter-patient variability and the tendency for cancer cells to evade cell death is a key factor for the multi-compartment model used to predict a patient's response to chemotherapy used in Chapter 5.

1.1.1 Cancer Stem Cells

In healthy tissue, stem cells are pluripotent cells with the unique capacity of being able to differentiate into specialised cell types. This differentiation pathway was once considered to be unidirectional, however it is now known that differentiation is a bi-directional process [100, 15]. In particular, stem cells are progenitor cells from which increasingly more specialised cells are derived. While this differentiation process, from stem or stem-like cells to more specialised cells, is often unidirectional, a backwards reaction referred to as de-differentiation is possible. Another difference between stem and differentiated cells is the

capacity for replicative immortality. While differentiated cells are subject to the Hayflick limit, stem cells can divide a sufficiently large number of times [95]. Similarly, stem cells exhibit self renewal wherein division can allow for stem-like progenitor cells in order to maintain a balance of stem to differentiated cells within the particular cellular niche.

Previously, the explanation of heterogeneous cell populations in tumours was explained via clonal evolution [75]. In this traditional view, the heterogeneity is the result of genetic heterogeneity within the tumour. As a result, mutations occur during mitosis that provide an adaptation to the cell within the local microenvironment. Due to the hallmarks of cancer, cancer cells exhibit increased proliferation and genetic instability which further increases the likelihood of advantageous mutations fixating within a particular cellular niche. Hence the heterogeneity of solid tumours is explained by mutations driven by evolutionary natural selection. In the cancer stem cell hypothesis, it is instead assumed that cells exist in a hierarchy. In this hierarchy, there are cells that behave in a stem-like manner. These so called cancer stem cells (CSCs) drive tumour initiation and are responsible for the varied phenotype of a cancerous niche. Importantly, cancer stem cells exhibit resistance to chemotherapy, especially quiescent cancer stem cells [104]. This informs the tumour growth inhibition model of Chapter 5.

Cancer stem cells are observed in various proportions within a tumour. Since cancer stem cells have a high degree of self renewal and un-bounded proliferation potential, they are an incredibly important niche to target by therapeutics. In Chapter 3 it is observed that large probabilities of invasion can be achieved by introducing even a single cancerous stem cell into an otherwise established tumour niche. Hence, since cancer stem cells evade destruction by therapy, cancer stem cells are an incredibly important component of reinfection. Even if radiotherapy or chemotherapy results in a large reduction in tumour mass, those quiescent stem cells that survive can seed an reintroduction of the tumour in the patient.

Cellular plasticity is the subject of study in Chapter 3 and refers to the capacity for cells to change their phenotype. This phenotypic plasticity can be acquired due to randomly occurring genetic or epigenetic changes to the cancer stem cells and hence informs treatment. In the absence of phenotypic plasticity, particular cancer stem cell niches can be targeted for destruction by the clinician. However, in the presence of plasticity cancer stem cell niches are incredibly unlikely to be destroyed as the pool of available stem cells can be refilled via de-differentiation.

1.1.2 Cell Cycle

The eukaryotic cell cycle is an ordered series of events that describe and control cellular division. During division the cell prepares for mitosis, the event by which one cell creates two new daughter cells. Roughly, the cell cycle can be described by three main phases. In the first, G_0, cells are quiescent. During quiescence, cells are non dividing. These inactive cells have exited the cell cycle. Some cells enter quiescence in the presence of an external signal (for instance, cancerous stem cells enter quiescence in the presence of chemotherapeutics). Similarly some terminally differentiated cells do not divide and so enter quiescence permanently.

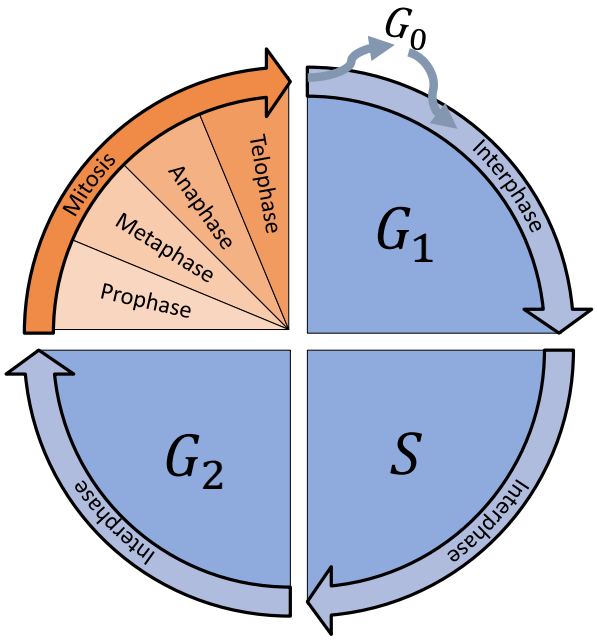


Figure 1.1: Phases of the cell cycle. In G_0 phase the cell is said to be quiescent - this represents a stable, non-dividing state. Then G_1, S, and G_2 represent the growth and synthesis phases. These interphases describe when a cell is preparing for division. Finally M, the mitotic phase, is split into four distinct parts: prophase, metaphase, anaphase, and telophase.

As cells prepare for division, they enter what is known as the "interphase" state of the cell cycle. During interphase cells progress through three sub phases: G_1, S, and G_2. Roughly speaking G_1 and G_2 are the two growth phases of the cell. During G_1

the cell is accumulating chromosomal DNA, proteins, and energy reserves to replicate each chromosome within the nucleus. During G_2, the cell refreshes its energy stores and synthesises the proteins necessary for mitosis. In between these growth stages the cell enters S, or synthesis, phase. In S phase the DNA undergoes replication resulting in two identical copies of each chromosome.

Finally, the cells enter the mitotic phase. In M phase there are (at least) four sub-phases that must be completed in order for the successful division of the cell into two daughter cells. The first of these sub-phases is the prophase during which the nuclear envelope breaks down and chromosomes condense and become visible. Next is metaphase, during which the chromosomes are lined up with one another at the metaphase plate. It is during metaphase that the chromosomes are maximally condensed. During anaphase, the chromatids are split apart and pulled rapidly towards the center, resulting in a visual elongating of the cell. Finally, in telophase, the chromosomes reach the opposite poles and begin to unravel. During this process the results of the previous mitotic phases are undone until nuclear envelopes form around the chromosomes.

1.2 Summary of Thesis

The thesis is laid out as follows: in Chapter 2 we introduce and describe some of the mathematical methods used throughout this work. In Chapter 3 we examine two models of cancer stem cell invasion in a heterogeneous population of stem and differentiated cells. We reconcile two seemingly contradictory results and extend the model to include transit amplifying cells. In Chapter 4 we present a method for performing label free classification of the mitotic phase of Jurkat cells without the need for biomarkers or dyes. In Chapter 5 we present an ordinary differential equation model of cancer stem cell response to chemotherapeutics and use reinforcement learning to find model free schedules for chemotherapy delivery in an adaptive, patient specific manner. Finally, in Chapter 6 we summarize and conclude the results. Code for the various numerical experiments performed in this thesis is hosted at `https://github.com/brydon/phd_thesis`.

Chapter 2

Methods Background

2.1 Introduction

The interplay of mathematical modeling and experimental data in investigating problems in mathematical biology can lead to new insights and techniques that can then be further investigated experimentally. In many cases in mathematical biology, the underlying structures and phenomena are influenced by stochasticity. Moreover the phenomena themselves are often far too complicated for rigorous analysis to be done by hand. Hence computational and statistical techniques are often employed to perform *in silico* experiments. In this chapter we present some of the mathematical and statistical techniques used throughout this thesis.

2.2 Stochastic Model Processes

There are many ways by which one can model biological systems and processes. A standard approach that has seen great success is to use differential equations [25]. Differential equations assume that whatever is being modeled can be effectively represented by a continuous function. In part, then, the appropriateness of differential equations in modeling depends upon how appropriate the continuity assumption is. For instance, if one is modeling the interactions of bacteria in a chemostat, the bacteria populations themselves do not actually obey a continuous function in practice as each bacterium is itself a discrete agent. However, as long as the size of the bacteria population is sufficiently large, the approximation by a continuous function can be a reasonable choice. However, any entropy

or stochasticity involved by aggregating the many individual bacterium into a continuous function is lost. Often these stochastic effects only slightly effect the quantitative dynamics of the phenomenon being modeled, but in some cases large qualitative changes can occur (such as when stochasticity drives the model into a different basin of attraction). Moreover, if the number of bacterium being modeled is small, then the continuity assumption breaks down further. In effect, continuous differential equation models of fundamentally stochastic processes only produce predictions of the mean value of the process. There are many situations in modeling single cell behaviour in which noise plays a large role. In [44] the authors model both single cells and populations of cells stimulated by TNFα to produce oscillations in NF-κB. In this case if one only models the mean oscillatory behaviour, one ends up with model predictions that are both different in amplitude and phase from the oscillations observed in single cell experiments. However, by employing stochastic models the correct amplitudes and phases are re-discovered.

2.2.1 Gillespie's Algorithm

Gillespie's algorithm allows one to solve stochastic differential equations (SDEs) numerically [34]. Instead of modeling a population by gradually, continuously evolving through various states, instead Gillespie's algorithm models the population as undergoing various discontinuous jumps between discrete states in time. Gillespie's algorithm is quite useful for a variety of reasons. For instance, in systems biology applications the number of reactions being considered can be exceptionally large. Furthermore many of these reactions considered can be highly nonlinear in their components. For such a system calculating the master equation or solving the partial differential equation that satisfies the generating function is often far too complicated to be solved in any manner other than numerically. In these situations numerical techniques like Gillespie's algorithm are quite useful. In 1977 Gillespie showed that the solution created by his numerical algorithm is equivalent to the solution of the master equation [34].

Gillespie's algorithm produces an ensemble of solutions. By combining a sufficiently large number of Gillespie simulations, one can approximate an empirical distribution of the model trajectories for each compartment. Suppose we are modeling the population of n cell types interacting with one another. Further, suppose these n different types of cells can interact producing m different reactions. These reactions could be apoptosis given the interaction of two cells, or spontaneous mitosis of a single cell without interaction, etc. Each of these interactions updates the total state vector of the system. Let $\vec{P}(t)$ be the n-dimensional state vector and $\vec{A}_i(t)$ be the change to this state vector effected by reaction i. For example, if $\vec{P}(t)$ corresponds to the number of cells of each cell type and

reaction i corresponds to cell death of cell type j then $\vec{A}_i(t) = -\vec{e}_j$ (where $\vec{e}_j$ refers to the vector where every element is zero except element j which is unitary). Further, define the propensity function of each reaction, a_i, such that $a_i(\vec{P}(t))\,\mathrm{d}t$ is the probability, given $\vec{P}(t)$, that reaction i is the single reaction that will occur in the infinitesimal time interval $[t, t + \mathrm{d}t)$. The "sojourn time", or time between reactions, then follows an exponential distribution.

There are many simplifications of the algorithm to ease computational complexity [35]. The method presented here is the so-called direct method. A single run of the direct method of Gillespie's algorithm follows the following method:

1. Initialize the system by assigning to $\vec{P}(0)$ some starting state
2. For each k do
   (a) Sample r_1 and r_2 uniformly independently from $(0, 1)$
   (b) Calculate τ, the sojourn time
   $$\tau = -\left(\sum_{i=1}^{m} a_i(\vec{P}(t_{k-1}))\right)^{-1} \ln(r_1)$$
   (c) Calculate μ, the index of the next reaction type by determining the value of μ that satisfies
   $$\sum_{i=1}^{\mu-1} a_i(\vec{P}(t_{k-1})) \leq r_2 \sum_{i=1}^{m} a_i(\vec{P}(t_{k-1})) \leq \sum_{i=1}^{\mu} a_i(\vec{P}(t_{k-1}))$$
   (d) Take $t_k = t_{k-1} + \tau$ and $\vec{P}(t_k) = \vec{P}(t_{k-1}) + \vec{A}_\mu(k)$
3. Stop if t_k is bigger than some maximum stop time, if a maximum number of iterations have taken place, or if an equilibrium solution is reached

Typically this process is repeated a large number of times until a distribution can be reasonably resolved.

2.2.2 Moran Model

A Moran model [73] is a discrete model of stochastic events for finite population sizes. If there are two or more species competing with one another for domination, then a Moran model can be simulated (or in some simple cases algebraically solved) to determine the fixation probability of the species. For instance, in a 2 species birth-death process one species has its reproductive rate normalised to 1 and the other at a varied rate r. In such a model at each discrete time moment an individual is chosen randomly (weighted by its proliferation probability) to reproduce and another is chosen to die in a similarly biased manner. From such a process the proliferation probabilities and fatality probabilities can be analytically calculated as a function of the number of members of the relevant species and their relevant fitnesses. Simulation of such a model is performed via Monte-Carlo updates at each individual time step in order to construct a Markov chain based on the proliferation and fatality probabilities of the species. Let K represent the fixed capacity of the Moran model, then it is the case that 0 and K are absorbing states of the system. Hence any state that is not 0 or K is necessarily a transient state and as such the probability of the system not converging to 0 or K for a species on an infinite time horizon is 0 [57]. Hence a Moran model will almost certainly become absorbed in 0 or K as the number of time steps approaches infinity.

2.3 Data Driven Models

Many statistical learning methods are data driven. In this regard one must be careful of how the algorithms interact with the data in order to ensure the trustworthiness of the results.

2.3.1 Data Folding and Generalisation

The essence of statistical learning methods is to generate a mapping between inputs and outputs by regressing onto training data. The goal of this process is to create a method that can utilize lessons from the past and project forward into generalisation in the future. Hence in order to ensure that the algorithm is not "teaching to the test", we should be careful in the way we partition the data we consider. For instance, by evaluating a machine learning network on the same data that it was trained on we can achieve effectively perfect classification importance. Indeed what is happening in such situations is the overparameterised model is learning to effectively memorize each sample and not generalize

between samples well. This is an undesirable quality in machine learning methods and there are many ways to deal with it. Hence the statistical tool of cross validation provides a necessary framework in which to operate. In such a framework supervised machine learning algorithms have the data split into training, test, and validation sets. Training sets, which typically are composed of the largest chunk of the data, are the samples the algorithm will use to set the parameters of the model. Then, testing data is a separate data set upon which the algorithm is evaluated. The quality of our algorithm is more readily demonstrable by ensuring these are separate. Similarly, the validation set is a separate data set upon which any of the hyperparameters of the method are tuned. This includes things like the number of iterations for which the model is trained and any various hyperparameters intrinsic to the model selection process (the architecture of a neural network or the number of neighbours to consider in a k-Nearest Neighbours classifier, for instance). For the same reason that it is important to ensure that the training and testing sets are different, it is also important to ensure that the testing and validation sets are distinct. If they were not, then the supposed performance of our model can be spurious and suspect. Figure 2.1 demonstrates this effect wherein the error rate on the training set continues to decrease even though the error on the held-out validation set is increasing. Such a model is called over-fit. By monitoring the validation error one can strop training early so as to ensure that the model can generalize well.

The goal of machine learning algorithms is to provide strong generalisation. This means that we want to ensure that our method behaves similarly well irrespective of particulars about the metadata. For instance, some machine learning techniques do not perform well in the presence of noise in the data, some algorithms are highly dependent upon the order on which data is presented to it, etc. These drawbacks are not desirable for a model to possess. Much effort has been devoted to ensuring the generalizability of the input data so that issues of domain shift do not occur. For instance, regularisation techniques often induce a stronger generalisation potential, ensemble learning similarly produces models that tend to generalize more strongly, the use of repeating cross-validation of multiple randomly selected subsets of the data and averaging the final results also similarly helps safeguard against issues in generalisation [36, 43, 19]. Furthermore, the success of the machine learning algorithm can be assessed by cross validating on a variety of differing subsets of the data. When the variance in these scores is large, it can be evidence of poor generalisation.

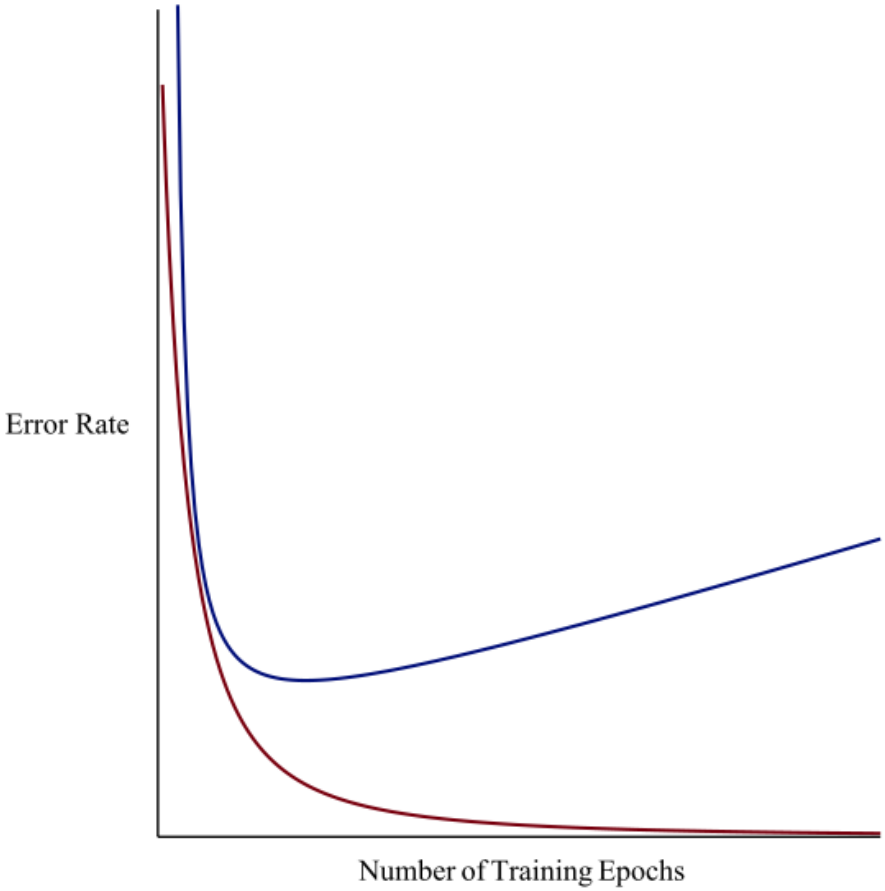


Figure 2.1: An example of early stopping based on validation data in machine learning. If one were to instead blindly minimize the training error, then the resulting model would be overfit to the particular training data and would not generalize well.

2.4 Classical Classification Algorithms

2.4.1 Decision Tree Classifiers and CART

Decision trees are a supervised learning model that utilize a graph theoretic framework for solving classification or regression problems. There are many algorithms by which one can generate decision trees such as CART, ID3, and C4.5 [96]. All of these methods follow the same basic structure but differ in the loss function utilised. The trees are constructed in a

way that minimizes the cost function over the binary tree. The basic algorithm involves collecting all the training samples (represented as vectors) and segmenting them into two classes by determining two numbers: the index of the vector component being considered by the cost function, and the threshold value by which the samples are stratified. Performing this process once will then stratify the N training samples into two separate groups. For each of those two groups the process is repeated again, recursively, as long as continuing to split the nodes results in a reduced cost function value or until a particular stopping criterion is met. Usually the stopping criterion is either: the maximum depth of the tree, the number of samples in each parent node, or the number of samples in each leaf node. For instance, one may only consider splitting a node if there are more than 10 samples in the node, however this splitting process can be further restricted so that at least 5 samples remain in each class. If one instead allows leaf nodes to contain single samples, then there is a danger of overfitting to the data. In effect, it is unlikely that the process is learning features and thresholds that are general for the problem and more likely that the process is just uniquely identifying each sample based on the training data. An example is presented in Figure 2.2 using data from Chapter 4.

The cost function considered is the so-called Gini impurity [61]. The Gini impurity is the sum of the probability p_i of an item of class i being chosen times the probability of the item being miscategorised. The Gini impurity then takes values in the range $[0, 0.5]$ reaching a minimum value when all samples are correctly classified (and a maximum when all samples are exactly incorrectly classified). Suppose there are K classes being considered for classification. Then for a given threshold α and index λ, we let p_i be the weighted proportion of samples labeled with class i in the set as determined by the partition defined by α and λ. Then

$$G(p) = \sum_{i=1}^{K} \left(p_i \sum_{k \neq i} p_k \right) = 1 - \sum_{i=1}^{K} p_i^2$$

is the Gini impurity of the partition p. Hence at each split of the tree, all samples are categorised into two classes and for each of these categorisations there is an associated Gini impurity for that partition p. If it is possible to split the samples again, without having too few samples in either the parent node or the leaves, then such a split is computed. If the resultant split results in a larger weighted average Gini impurity between the two nodes, then the split is rejected.

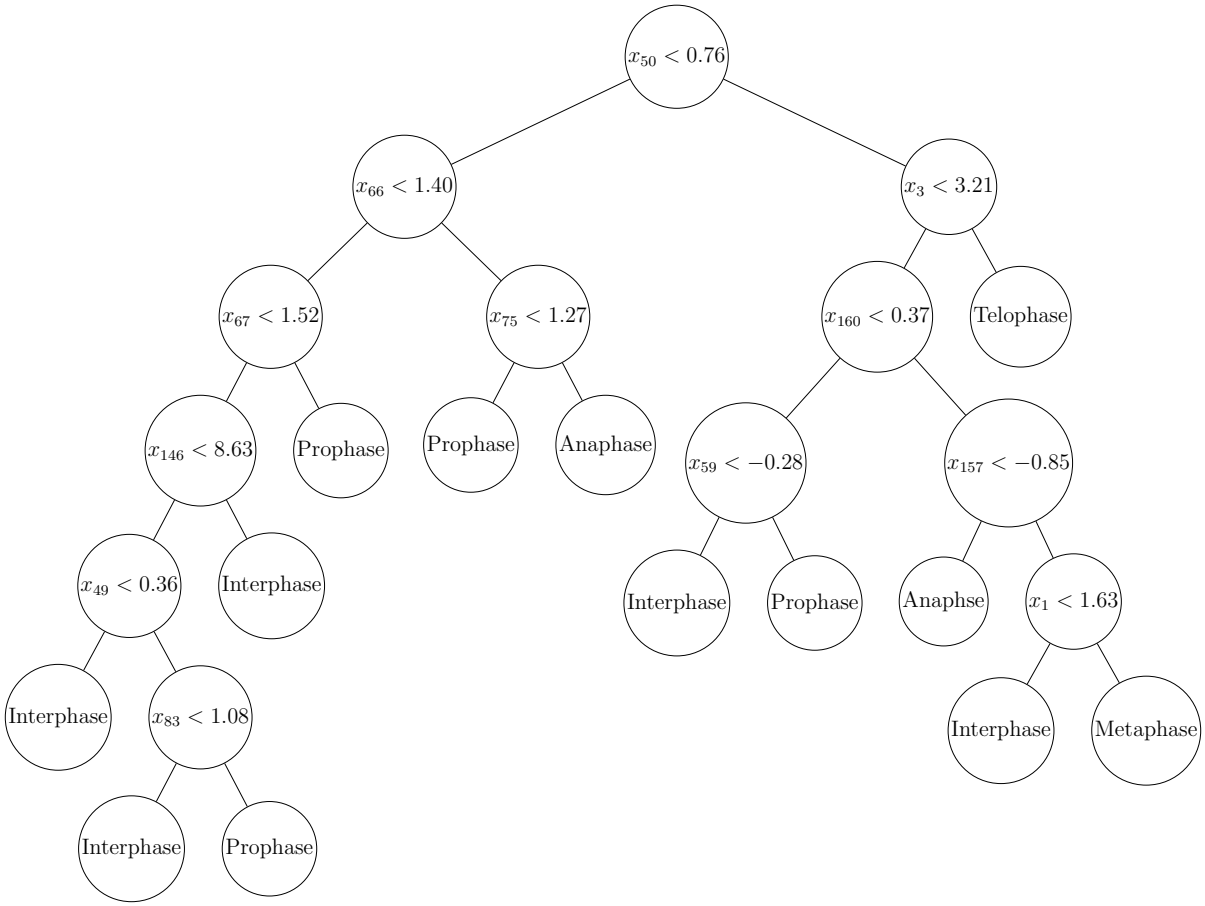


Figure 2.2: Example of a decision tree classifier used for the image classification task in Chapter 4. For input vectors $\vec{x}$, the tree bins the vector into various classification groups depending on the magnitude of certain entries. The convention is that left handed branches of the binary tree are followed when $x_\lambda < \alpha$ and right handed branches when $x_\lambda \geq \alpha$.

2.4.2 k-Nearest Neighbours

The k-Nearest Neighbours algorithm is another supervised learning technique used in classification and regression. Despite its simplicity the algorithm is quite successful in many applications. For instance, in [7], the authors concern themselves with reconstructing a two-dimension bifurcation diagram using the combinatorial structure of Conley-Morse graphs and a 1-nearest neighbour algorithm. In effect, the k-Nearest Neighbours classifier assumes that samples from the same class can be considered as inhabiting the same neighbourhood

as one another under a particular metric. Often the l^p metrics are used for determining the distance between samples, for an appropriate choice of p. The method works by assigning for every element of the testing data a label determined by weighted consensus voting of the k samples from the training data that posses the smallest distance under the metric. The weighting of the voting can either be determined via pre-allocated sample weights, distance between the samples (the closer of the k nearest samples having a proportionally higher vote in the consensus voting process), or a combination of the two. Hence k, the distance metric, and whether to weight the consensus voting are hyperparameters of the k-nearest neighbours process.

2.5 Artificial Neural Networks

2.5.1 Multilayer Perceptrons

The Rosenblatt-McCulloch-Pitts Perceptron was a simple mathematical model of how neurons in the brain process signals [43]. The simple idea was that a perceptron took in a collection of binary signals $x_1, x_2, \cdots, x_n$. The output of the perceptron was either a 1 or a 0 decided by calculating a weighted sum and comparing it to some threshold (or, equivalent, adding a bias term and thresholding against 0). For weights $w_1, w_2, \cdots, w_n$ and bias b, the perceptron obeyed the following function

$$p(\vec{x}) = \begin{cases} 0 & \text{if } \sum_{i=1}^{n} w_i x_i + b \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

This process was then extended into what are known as multilayer perceptrons, or artificial neural networks, by feeding the same input signal $\vec{x}$ through multiple perceptrons and collecting all their outputs into a single output vector $\vec{y}$. This output vector is then the output of one *layer* of the neural network which can then be fed into another layer of perceptrons to form deeper and deeper neural networks. In this sense, the number of perceptrons per layer is often referred to as the width or breadth of the network while the number of discrete layers is referred to as the depth of the network. The discontinuous binary signal of the original perceptron model was relaxed to instead allow for various continuous functions. However, due to Hahn-Banach theorem, for the purposes of universality it is only required that the activation function has points of discontinuity whose closure is a Lebesgue null-set [6, 16]. While originally the discontinuous step function was replaced with various sigmoid functions like $\tanh(x)$ or $(1 + e^{-x})^{-1}$, recently the most popular form

of function used is the so-called ReLU function, or rectified-linear-unit.

$$\sigma(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{otherwise} \end{cases}$$

These functions, either the step function or the ReLU function or otherwise, are referred to as the activation function (see, for example, Table 1 of [6] for examples of commonly used activation functions). The notation is slightly easier if we consider vectorizing these functions. In that sense $\sigma(\vec{x})$ is the vector achieved by component-wise applying $\sigma(x)$ to each element x_i of the vector. Then, $\sigma(\vec{x})_i$ refers to the ith value of such an operation.

In multilayered networks, the input vector is often referred to as the input layer and the output vector as the output layer. Any additional layers in between are called the "hidden layers" as the values of the neurons at this layer are not directly observed. It is common to use one activation function for the interior layers and a different activation function for the output layers. For instance, in binary classification tasks the output layer is typically a single neuron whose values range between 0 and 1 where the value of the output neuron corresponds to the confidence that the input vector belongs to one class or the other. Similarly, for multi-class classification tasks there are typically as many output neurons as there are classes and the output neurons then represent the probabilities of the input vector inhabiting a particular class. Hence a different activation function is used as the output neurons should sum to 1. Hence the "softmax" function is often used for the neurons in the output layer where

$$\sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}.$$

Notationally, if $\vec{a}^{(0)}$ refers to the input vector $\vec{x}$, $\vec{a}^{(l)}$ to the value of the lth layer in the neural network, and $\sigma^{(l)}$ to the activation function of the lth layer, then for weight matrices $W^{(l)}$ and bias vectors $b^{(l)}$, the value of the ith layer of the network can be computed as

$$a(l) = \sigma^{(l)}\left(W^{(l)}a^{(l-1)} + b^{(l)}\right).$$

That is, the value of each layer is computed by applying the weights and biases of the layer with the output of the previous layer and then feeding *that* as input into the activation function.

Hence for a given input vector computing the output of the neural network is trivial give a set of weights and biases. The problem then in machine learning is to determine the value of these weights and biases. Typically these weights and biases are initialised as Gaussian data [43] and are then trained via stochastic gradient descent optimisation.

To define the optimisation problem various loss functions can be considered for minimisation. For a given n dimensional output value y of our neural network and target value $\hat{y}$, the two loss functions we use most commonly in this thesis are the mean-squared-error (MSE)

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

which measures the Euclidean distance between the two signals and the categorical-cross-entropy function for multi-class classification problems

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{n} y_i \, \log(\hat{y}_i)$$

which is a measure of how distinguishable two discrete probability distributions are from one another. In this sense the categorical-cross-entropy loss function is only used when the final layer is activated via the softmax function so as to ensure a positive, real valued loss function.

Oftentimes machine learning problems are further regularised by including various other terms in the loss function. Typically this takes the form of averaging multiple loss functions. For instance, in [84, 85] the authors train the so-called physics-informed-neural-networks by averaging a MSE loss function with a custom loss function representing how well the function defined by the network solves a partial differential equation.

While the loss functions so far have been represented as functions of the input and output data, in the optimisation context we truly think of them as functions of the weights and biases of the network. Numerous optimisation algorithms can be used to minimize this loss function, in the aforementioned [84, 85] papers the authors used L-BFGS, a quasi-Newton method, to train their networks. In practice, however, it is much more common to use a variant of stochastic gradient descent. The training problem then attempts to estimate the gradient of the loss function with respect to these weights and biases by computing a numerical gradient for each training sample via automatic differentiation and averaging these results to achieve an estimate of $\nabla_{W^{(l)}}\mathcal{L}$ and $\nabla_{b^{(l)}}\mathcal{L}$. Importantly, neural networks lend themselves well to automatic differentiation instead of computing such a gradient estimate using finite-difference equations or similar (as such methods would be computationally intractable). Then for a given learning weight η the weights are updated in accordance to the backpropagation algorithm (where "$\odot$" refers to the entrywise, Haddmard product):

1. Feed the input $\vec{x}$ through the network, calculating $z^{(l)} = W^{(l)}\, a^{(l-1)} + b^{(l)}$ and $a^{(l)} = \sigma(z^{(l)})$ at each layer.

2. Output the error between the output y and target output $\hat{y}$ as $\delta = \nabla\mathcal{L} \odot \sigma^{(l)\prime}(z^{(l)})$ for the output layer.

3. Propagate the error backwards through the network by computing $\delta^{(l)} = ((W^{(l+1)})^T \delta^{(l+1)} \odot \sigma^{(l)\prime}(z^{(l)})$

4. The gradient of the cost function is then calculated as $\frac{\partial\mathcal{L}}{\partial W_{ij}^{(l)}} = a_j^{(l-1)} \delta_i^{(l)}$ and $\frac{\partial\mathcal{L}}{\partial b_i^{(l)}} = \delta_i^{(l)}$.

5. Update the weights at each layer by $W^{(l)} \to W^{(l)} - \eta \nabla_{W^{(l)}}\mathcal{L}$ and $b^{(l)} \to W^{(l)} - \eta \nabla_{b^{(l)}}\mathcal{L}$

In stochastic gradient descent, the gradients are calculated by randomly splitting the training data into various batches of a particular size `bs`. For each batch, the weights are fixed and the gradients are calculated via backpropagation as above. Then, the weights are updated via the gradient rule above but with a learning rate $\eta/\texttt{bs}$. This process is repeated until all training samples have been seen. That constitutes one epoch of training. Typically training is continued for many epochs until the loss function has stopped decreasing or achieved a minimal value. This batching process is performed due to the fact that neural networks lend themselves quite well to parallelisation especially on modern GPU or TPU hardware. Hence this is primarily a computational concern. A popular variant of the stochastic gradient descent algorithm is that of Adam optimisation. The Adam optimizer is very similar except it uses a per-parameter learning rate that is adaptively selected based on both the average and the variance of recent magnitudes of the gradients for the weights [54].

It can be shown (for instance in [43]) that a neural network with sigmoid activation functions and only a single hidden layer can arbitrarily approximate any continuous function by taking the width of the network sufficiently large. This is sometimes referred to as the "arbitrary-width" case of the universal approximation theorem. It was later shown, that for ReLU activated neural networks that any function $f : \mathbb{R}^n \to \mathbb{R}^m$ can be arbitrarily approximated by a neural network F with width of $\max(n+1, m)$ for any Bochner-Lebesgue p-integrable function [53] by taking the depth of the network sufficiently large. This is sometimes referred to as the "arbitrary-depth" case of the universal approximation theorem.

2.5.2 Convolutional Neural Networks

The neural networks considered in the previous section are referred to as "fully connected" neural networks as each neuron in a layer is connected with each neuron in subsequent layers. This results in an exceptionally large number of training parameters that greatly slows down training for very deep neural networks. Moreover, in certain contexts it is

natural to assume that there would be correlations between the weights. For instance, in computer vision tasks instead of connecting every single pixel in a 25×25 image to every single neuron, instead pixels in a small window (for instance 5×5) are convolved and then pooled together and connected to a single neuron. There are many ways that this pooling is achieved, i.e. averaging or taking maximal values, and there are many sizes of this window that can be considered. Furthermore, often the window is not moved pixel-by-pixel throughout the image but instead jumps by various strides, further reducing the number of trainable parameters. Importantly, the weights used in the 5×5 convolution filter are the same across the entire image. This vastly reduces the number of learnable parameters and as a result many convolution filters and pooling layers can be included in the depth of the network. The intuition behind this approach is that the local connectivity follows the concept of receptive fields. By moving the convolution filter across the image in strides, we ensure that the features selected by the convolution filter are translation invariant. This is desirable in vision tasks as individual convolution filters can learn properties like edge detection, brightness detection, or other features that ought to be exhibit translation invariance. Hence CNNs exploit spatial locality of vision problems without the need for many learnable parameters.

2.5.3 Generative Adversarial Networks

Generative adversarial networks, or GANs, are an example of a deep-learning based framework for creating synthetic data. GANs were first described in the seminal 2014 paper by Goodfellow *et al.* [37]. GANs are but one example of generative frameworks, there are others such as variational auto-encoders, flow-based generative models, or (more recently) diffusion models [55, 88, 89]. GANs have seen incredible applications creating synthetic images that are nearly indistinguishable from real images. For instance, the website `www.this-person-does-not-exist.com` showcases generated random human faces created using StyleGAN [52] that, to the naked eye, appear real.

Generative modeling is an unsupervised learning problem where two models are trained simultaneously. Each GAN consists of a discriminator and a generator model. The discriminator model is fed input from two sources: a collection of real images and a collection of synthetic images from the generator model. The discriminator then attempts to determine which of the images are synthetic and which are real outputting a confidence score in $[0, 1]$ where 0 corresponds to a synthetic sample and 1 to a real sample. As the discriminator is trained it gets better at telling the difference between the two images, however simultaneously the generator network becomes more adept at creating realistic looking synthetic images. It should be noted that while generative models are overwhelmingly

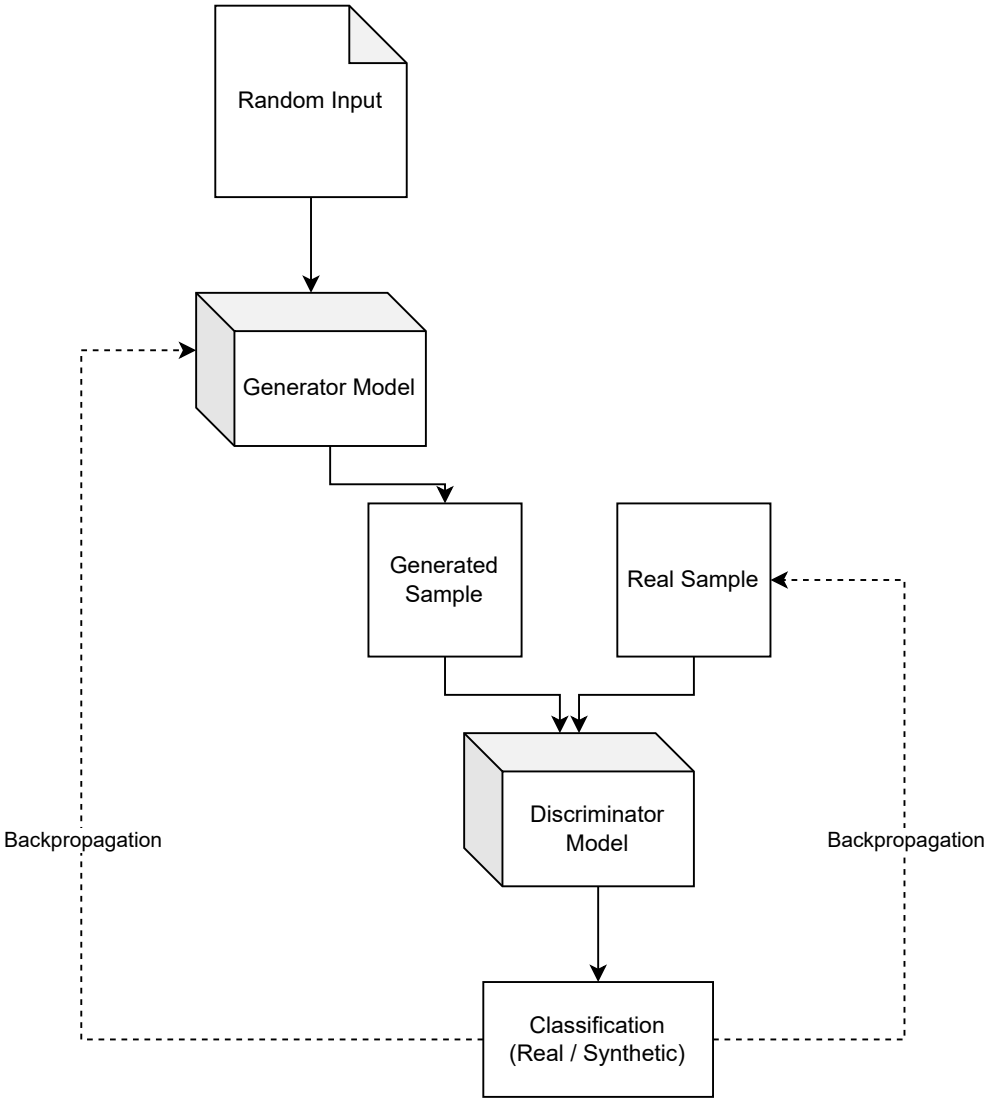


Figure 2.3: An overview of the GAN framework. A discriminator and generator network are trained in tandem. The input to the network is a collection of random input samples with labels 0 and real input samples with labels 1. Throughout learning the generator becomes a more adept counterfeiter while the discriminator becomes increasingly less confident of the veracity of the input.

used to generate imaging data, they could be used to generate any type of synthetic data (be it random vectors from a particular distribution or text tokens in the style of a particular

author).

The GAN framework can naturally be analyzed with the tools of game theory, hence the nomenclature of "adversarial" networks. In effect, the two competing models are playing a zero-sum game. When the discriminator is able to successfully classify samples as real or fake, its weights are not updated (or, in a game theoretic sense, there is no reward). Conversely, the generator model would undergo large changes to the weights in such a case as successful discrimination is emblematic of unrealistic synthetic data. At convergence, the output of the discriminator network is perfectly undecided, outputting a classification score of 0.5. Convergence of this type can be thought of as a Nash equilibrium [30]. When convergence is achieved, the discriminator aspect of the model is discarded and the generator is kept for the creation of synthetic data.

Generative models are useful for a variety of reasons but the primary use-case we consider is for data augmentation during classification problems. In situations where one has limited real data, additional synthetic data can assist in the convergence of many deep learning algorithms. Perhaps more importantly, however, inclusion of synthetic data also provides a strong regularisation effect by increasing the generalisation of the model [42].

GANs are enticing in how simple and natural the framework is, however they can be quite difficult in practice to achieve convergence due to an extreme sensitivity to hyperparameter values [90]. In particular, in situations where the discriminator converges too quickly, both the generator and discriminator networks can exhibit the notorious "vanishing gradient" problem wherein updates to the gradients are too small to move the network out of any local minima in the loss landscape [1]. Another common issue that generative models face is mode collapse. In this situation the generator early in training produces a sufficiently plausible output. In an attempt to continue to fool the discriminator the generator converges on producing *only* that output. This results in the discriminator effectively memorizing that this one particular output is synthetic. Each iteration of the generator over-optimizes for the discriminator and the generator ends up rotating through a small set of output types. The issue of mode collapse is problematic as it is desirable for each random input to produce a sufficiently different, but plausible, random output. Indeed, without additional regularisation techniques many GAN optimisation problems lack Nash equilibria all together [90, 29].

One sub-type of generative problem that we concern ourselves with in Chapter 4 is that of image-to-image translation. In such a task, one considers inputs from two distribution. The classical example is that of pictures of horses and zebras. The translation task is then concerned with taking images of horses and modifying the pixel data to produce an image of a zebra via a process referred to as in-painting. In this problem, we wish to maintain

other features of the original horse picture such as pose, background, relative size, etc. One such regularisation technique used in this problem is that of cycle-consistency [118, 81]. In cycle-consistency the loss function has an additional term wherein the samples attempt to minimize the difference between an original source image and the result of first translating the source image to the output distribution and then translating the output back to the source distribution. In the horse-to-zebra example this would result in first in-painting a horse to appear as a zebra and then in-painting the zebra output to appear as a horse. By ensuring that this cycle-consistency is maintained, image translation problems avoid the issues of mode collapse.

Image translation problems also allow for interpolation between input and output images. For pixel data, it would be natural to assume that interpolation follows a linear interpolation between image pixels, but for translation problems this interpolation takes place over the approximation of the latent manifold of the data. One classic image dataset for use in machine learning is the MNIST dataset of handwritten digits. In this domain pictures of the numeral 0 in a particular handwriting style can be thought of as the source distribution and pictures of the numeral 1 as the target distribution. If we were to merely linearly interpolate between these images, the intermediary steps would not be recognizable as handwritten digits. However, when interpolating over the latent approximation each intermediary step is a plausible digit. It is in this manner that generative networks are able to translate random input data into plausible human faces – by interpolating along the latent manifold between training samples. As a result, each image generated by a GAN implicitly contains information about its training data [19, 115].

2.6 Ensemble Boosting

A standard approach to improving the performance of a classification or regression algorithm is to employ ensemble boosting [116]. In this thesis we are primarily concerned with using ensemble boosting for classification. The most famous ensemble method is probably ADABoost, upon which many ensemble boosting algorithms are based [92]. In ensemble boosting one combines many classification algorithms sequentially in order to improve performance. These classifiers are trained iteratively with each subsequent classifier updating the sample weights during training based upon the successes and failures of the prior iterations. In the end, the outputs of all the classifiers are combined in a weighted sum where the weight depends upon the individual loss values of each classifier. Hence ensemble boosting is not only learning to classify the data, but it is also learning how to improve each classifier from its previous mistakes, and its learning which of these classifiers to

trust most. In imbalanced datasets, it is desirable to use random undersampling. Random undersampling is a process by which one selects samples randomly from each class so that the classes during training are balanced even if the data during testing are imbalanced. The downside to random undersampling is that the algorithm is not leveraging all the available data. In RUSBoosting (Random UnderSample Boosting) the adaptive weights are used to weight the random undersampling selection process. The process follows the following algorithm from [94] for the case where there are n_{clf} classifiers.

1. Initialise the weights $W_1(i) = 1/n_{\text{clf}}$ for all i
2. For $i = 1 \cdots n_{\text{clf}}$
   (a) Use the weights $W_t(i)$ to randomly select, as weighted by $W_t(i)$, an equal amount of samples from each class.
   (b) Fit the classifier f_t to the undersampled dataset
   (c) Calculate the pseudo-loss $\epsilon_t = \sum_{(i,y):y_i \neq y} W_t(i)\,(1 - f_t(x_i, y_i) + f_t(x_i, y))$.
   (d) Calculate the weight update parameter $\alpha_t = \frac{\epsilon_t}{1-\epsilon_t}$
   (e) Update $W_t(i)$ according to $W_t(i+1) = W_t(i)\,\alpha_t^{\frac{1}{2}(1+f_t(x_i,y_i)-h_t(x_i,y:y\neq y_i))}$
   (f) Normalize $D_{t+1}(i)$
3. Return the final ensemble classifier hypothesis $F(x) = \text{argmax}_y \sum_{t=1}^{n_{\text{clf}}} f_t(x, y) \log(\alpha_t^{-1})$

2.7 Reinforcement Learning

In contrast to supervised or unsupervised learning methods, reinforcement learning occupies its own classification of machine learning algorithm. The classical example of reinforcement learning algorithms are those of reinforcement learning agents learning to play video games [71]. As an illustration consider if one wished to use reinforcement learning (or RL) to play a game of Pong. All one needs to provide the environment with is the rules of what actions it can take (in this case, the buttons that can be pressed in pong) and allowing it feedback in the form of rewards (in this case, the score of the pong game). An RL agent will then interact with the environment in an exploratory manner, attempting random actions in order to assess the impact on rewards. In a long time frame, the agent will begin to ascertain which actions result in positive rewards (scoring a point) and which resulted in negative rewards (allowing the opponent to score a point). Hence there is a trade off

between exploration and exploitation: exploration allows the agent to experiment with the effects of its actions and exploitation capitalizes on the high value actions it has taken so far. To this end, often an *epsilon-greedy* algorithm is employed for choosing actions (see, for instance, [99]). In an epsilon-greedy algorithm, the action that seems best given previous experience is chosen with probability $1 - \epsilon$ (the so-called greedy choice), and a random action is chosen with probability ϵ. In this way exploration is still occurring (through the random actions) and the algorithm is still exploiting its knowledge (through the greedy actions). While ϵ can be kept constant during training, in practice ϵ is usually set as a number close to (or equal to) 1 initially and is decayed as training progresses. In this way, the algorithm becomes less exploratory as it gains more insight into the value function during training. Interestingly RL approaches achieve impressive performance at these tasks even in the absence of a direct model of the environment. Model-free RL treats the environment as a black box in the process of solving the control problem. This makes RL of particular interest to mathematical modelers. In the presence of ample enough patient data, an RL agent could learn the actions of chemo-therapy dosing and scheduling without the need of a patient specific model of the drugs effects on the particular patients body. Moreover, a hybrid approach of data and multiple models whose accuracy is different at certain scales can theoretically be employed. The goal of reinforcement learning is to acquire a value function by which the effect of different actions on the long-term cumulative reward can be assessed, this value function can then give rise to a policy of what action to perform in any particular state of the environment. This policy then maximizes not only the immediate benefit, but the long term cumulative desirability of environment states. In terms of cancer modeling, this is desirable for ensuring long term desirability of patient states instead of short term rewards of patient states. In this regard reinforcement learning is a computational approach to automating decision making in a directed manner. In contrast to other approaches, RL emphasizes the learning of an agent with direct interaction with an environment in the absence of supervision or models of the environment.

Reinforcement learning tasks can be described via the framework of Markov decision processes [99]. There are many ways to solve this RL problem including policy iteration, temporal-differenc learning, and deep Q learning. It is the method of deep double Q learning (DDQN) that we employ in this work. In DDQN, the following particular form of the Bellman equations are solved

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma \sum_{s' \in \mathbb{S}} p(s' \mid s_t, a_t)\, Q^*(s', \operatorname*{argmax}_{a' \in \mathbb{A}}(Q^*(s', a'))) \tag{2.1}$$

in order to derive an optimal policy as defined by

$$\pi(s_t) = \operatorname*{argmax}_{a \in \mathbb{A}} Q^*(s_t, a). \tag{2.2}$$

In Equation 2.1 Q represents the perceived long term value of performing action a_t in state s_t where $R(s_t, a_t)$ is the initial, short term reward, $p(s' \mid s_t, a_t)$ is the probability of transitioning into state s' by taking action a_t in state s_t, and $\gamma \in (0, 1]$ is a discount factor for long term rewards. In effect, small values of γ place more importance on maximizing immediate reward and larger values of γ are more concerned with a longer-horizon maximisation of long term, cumulative reward. Note that the form of the optimal Q, denoted Q^*, is highly nonlinear. As such, representing Q^* by artificial neural networks is a powerful approach to solving the Bellman equations in Q learning. The DDQN method from [102] employs two versions of Q^* that lag temporally behind one another. There is the target network and the evaluation network. The target network is periodically updated after a set number of iterations to take the same form as the evaluation network. By updating it less frequently, we avoid the issue of over-estimation of action-state pairs. In contrast, when only using one Q network there is the tendency to spuriously over-estimate Q values which leads to unstable training and a low quality policy [102]. In DDQN the target network is used for action selection and the evaluation network for action evaluation. This changes Equation 2.1 slightly to the following:

$$Q_{\text{eval}}(s_t, a_t) = R(s_t, a_t) + \gamma \sum_{s' \in \mathbb{S}} p(s' \mid s_t, a_t)\, Q_{\text{eval}}(s', \operatorname*{argmax}_{a' \in \mathbb{A}}(Q_{\text{target}}(s', a'))). \tag{2.3}$$

2.8 Hyperparameter Optimisation

Often we use data to train a function towards a particular response. In effect, we desire to fit a function f to some training data $\mathcal{D}_{\text{tr}}$ in order to maximize some metric $g(f(\mathcal{D}_{\text{te}}))$ over the testing data. As the previous sections in this chapter have mentioned, f often depends not only on the parameters we select in our training process, but also upon the so-called hyperparameters of the training process. Hence, our optimisation problem becomes: given a set of hyperparameters θ_{tr} use training data $\mathcal{D}_{\text{tr}}$ to find training parameters θ_{tr} in order to maximize our metric $g(f(\mathcal{D}_{\text{te}}; \theta_{\text{tr}}, \theta_{\text{hp}}))$ over the testing data. Hence we first must tune our hyperparameters before we can tune the model parameters. This hyperparameter tuning process should be evaluated on a different set of data than the testing data in order to ensure that spurious overfitting is not occurring. There are many methods for hyperparameter tuning. In situations wherein there are a (small) finite number of hyperparameter options, one can brute force the method by trying all possible combinations over hyperparameters evaluated on the held-out validation set. Even if the hyperparameter space is continuous and infinite dimensional, sometimes selecting a lattice of points for a hyperparameter sweep is computationally feasible. However, if the data do not lend themselves to a natural lattice

or if the number of potential points to brute force is prohibitively large, then other methods should be considered.

Suppose we have some function $f : \Omega \to \mathbb{R}$ that we wish to maximize on some domain $X \subseteq \Omega$. As an aside, in the context of hyperparameter optimisation usually we are choosing our hyperparameters such that a particular metric is *maximised*. If instead, one was concerned with minimizing f, then one could employ the usual trick of maximizing an auxiliary function g defined as $g(x) = -f(x)$ where the maximum of g coincides with the minimum of f. In cases where we know the function f there are various methods we can use for optimisation (for instance, if we additionally know that f is convex, then the greater class of convex optimisation algorithms apply). If, instead, f is a black-box function, then we can still optimize f using Bayesian optimisation. In particular, Bayesian optimisation is especially useful when the black-box function is particularly expensive to evaluate (as is often the case when f is obtained via fitting a neural network).

In Bayesian optimisation, we construct a probabilistic belief about f and design an acquisition function $a(x)$. Broadly speaking, $a(x)$ is an inexpensive function whose value is related to how desirable evaluating the original function f at x is for the purpose of the maximisation function. As such, to determine the next candidate point x to evaluate, we have to solve a (related, and much less computationally expensive) optimisation problem on the acquisition function $a(x)$. In principle, many acquisition functions could be used. In practice, the so-called "Upper Confidence Bound" acquisition function is typically used [32].

Given observations $\mathcal{D} = \langle \mathbf{X}, \mathbf{y} \rangle$ we condition a Gaussian process prior as

$$p(f) = \mathcal{G}\,\mathcal{P}(y; \mu, V). \tag{2.4}$$

where μ is the mean value of the data conditioned on the data $\mathcal{D}$ and V is the noise-free variance of the data conditioned on $\mathcal{D}$.

The acquisition function for the optimizer is defined as follows where $\beta > 0$ is a tradeoff parameter such that larger values of β correspond to a more explorative optimizer.

$$a_{\mathrm{UCB}}(x; \beta) = \beta \sqrt{V(x)} - \mu(x) \tag{2.5}$$

This acquisition function is then maximised over the predefined hyperparameter space in order to choose the next candidate hyperparameter tuple for optimisation. Hence as the Bayesian optimizer encounters the true value of $g(f(x))$ for various hyperparameter points, it estimates the value of $g(f(x))$ over the entire hyperparameter space in order to only evaluate the computationally expensive function $f(x)$ at candidate points that are more likely to produce higher values under the metric g.

2.9 Why does Deep Learning Work so Well?

There is much about deep learning that is counter-intuitive. This small section of this thesis can not hope to explain all of these issues or even fully survey all the literature on such subjects. Deep learning in particular is overburdened with no shortage of examples of success. Deep learning has presented a categorical leap forward in the applications of statistical learning to various tremendously complex regimes, however there is still as of yet not a truly deep theoretical understanding of why deep learning continues to work so well. The bedrock for deep advancements in learning theory is being laid and certain long-standing hypotheses about machine learning are starting to develop theoretical backing.

The question of why highly overparameterised neural nets do not result in overfitting is a complicated one. Indeed state of the art generalisation performance is observed when almost 2 million parameters are used to fit a CNN to a dataset of 50,000 training images [6, 22]. In the wonderful review article [6] the authors note that this question of the "generalisation puzzle" seems to contradict the standard bias-variance tradeoff of statistical learning [65, 20]. A statistical learning algorithm demonstrating high bias fails to recognize relevant relations between input and output data and is sometimes referred to as underfitting. Similarly, an algorithm with high variance is incredibly sensitive to small perturbations in the training set and may result in regressing to the noise in the training data. Such an algorithm is called over-fit. Typically overparameterisation leads to overfitting and poor generalisation, however empirical evidence abounds suggesting that deep learning techniques avoid this issue.

More issues seem to be avoided in modern neural network learning as well. In particular, stochastic gradient descent and its variants are naïve first order methods that will almost certainly converge to local minima in the presence of non-convex loss landscapes. Indeed, many statistical learning tasks are non-convex [91]. However it has been shown that gradient descent converges with exponential rate arbitrarily well insofar as the width of the neural network is sufficiently large. In addition, during training the weights achieved by the gradient descent algorithm stay nearby the initialisation points [18]. Indeed, in [103] the authors demonstrate that in the case of linear or quadratic activation functions (allowing for piecewise linear activation functions, insofar as the closure of the points of discontinuity form a measure zero set), simply increasing the overparamaterisation of a network actually rids the loss landscape of spurious local minimal. This is not true of non-polynomial, non-negative activation functions in general though, as the authors construct multiple counter-examples where spurious local minima occur. However, the measure of these minima decreases as the width of the network increases at a rate inversely proportional to the hidden layer size.

In part, neural networks avoid the curse of dimensionality due to what is referred to as the manifold hypothesis [63, 10]. The manifold hypothesis supposes that natural data forms lower-dimensional manifolds within its embedding space. In this viewpoint, the task of a classification algorithm is fundamentally to disentangle multiple manifolds in order to stratify data between classes. As such, as long as a network is sufficiently overparameterised such that the dimensionality of the network is larger than the intrinsic dimension of the data, then the lower dimensional manifold can be recovered. Much work (see for instance [63, 10, 60] and especially [31]) has been done to try and verify the manifold hypothesis for various datasets, but at least to our knowledge no such general approach has been developed. Hence, it is best to think of the manifold hypothesis as a heuristic. (Indeed, deep learning in general "approximates solutions to NP-hard problems via heuristics" [5].) In effect the veracity of the manifold hypothesis for a particular deep learning problem appears to rely heavily on intrinsic features of the datasets themselves and less due to particulars of network architecture.

Chapter 3

The Effects of Phenotypic Plasticity on the Fixation Probability of Mutant Cancer Stem Cells

In this chapter we examine two models of phenotypic plasticity using *in silico* experiments. We demonstrate that two contradictory results can be reconciled by considering different mechanisms of varying the rates of plasticity in the models. We demonstrate that increasing fixation probability is not merely due to blindly increasing the plasticity rate of mutant stem cells. Indeed, the fixation probability of a mutant depends in a non-monotone manner on its plasticity rate.

Dr. Dominik Wodarz assisted in the design and interpretation of the study. The work presented in this chapter is published in the Journal of Theroetical Biology [24].

3.1 Introduction

Cancer invasion is a complex process of the cellular ecosystem and micro-environment. Typically, cancerous cells achieve evolutionary success by mimicking, and subverting, the behaviour of regular, healthy tissues in the host [58]. In healthy tissue a distinct stratification of tissue structure is observed where the self-renewal capabilities of terminally differentiated cells rely upon a discrete subclass of the cellular population deemed stem cells [107]. In particular, these stem cells can behave invasively through the process of division and specialisation [9]. As healthy, specialised (or differentiated) cells die, the stem cells

divide producing the required differentiated cell as a byproduct of mitosis. In this way the population of specialised, terminally differentiated cells is maintained [9, 106]. However, in order to maintain tissue homeostasis, this mechanism must be controlled by both positive and negative feedback loops. To that effect there is mounting evidence that differentiated cells can de-differentiate back into adult stem cells in both healthy and non-healthy tissue [106, 100, 15].

This process of cellular invasion in the tissue driven by a particular niche of cells is very similar to the dynamics observed in invasive cancers. This has led to the cancer stem cell hypothesis that posits that the invasion, metastasis, and regulation of solid tumours are driven by a (typically) small sub-population of cells that behave very much like stem cells [76, 87, 97, 67]. In particular, this same behaviour of differentiated cells undergoing some process of de-differentiation and recovering some stem-like properties has been implicated in many forms of cancers experimentally [38, 13, 48, 82].

Many models have been proposed to examine the mechanics of the invasion of these plastic cancer cells in tissue for instance in [117] the authors discover conditions under which de-differentiation is selected while in [51] the authors discuss how de-differentiation influences the time to acquisition of mutations. Another two papers that have mathematically modeled the effect of this stem-cell plasticity and de-differentiation on the evolutionary dynamics of cancer stem cells [64, 109] arrived at contradictory results given similar starting assumptions. Both papers are concerned with how the invasive efficacy of mutated stem cells depends upon the degree of de-differentiation considered. In particular, both investigate a situation where a resident type of stem cells and differentiated cells have fully saturated within a population and by introducing a single mutant type within the population the fixation probability of these mutants is recovered. In [64] the authors conclude that as the rate of plasticity increases in a population of cancer stem cells, the fixation probability of mutant stem cells increases. In [109], however, the author concludes that as the plasticity increases in a population of cancer stem cells, the fixation probability of mutant stem cells decreases. These results are evidently at odds with one another and it is the purpose of this chapter to reconcile and explain the differing dynamical predictions of these models.

3.2 Modeling

The structure of the models is the same in both papers [64, 109]. The authors consider a population of resident stem cells in an equilibrium state. They then introduce a mutant, cancerous stem cell into the wild-type population and investigate the probability that the single mutant achieves fixation as a function of the plasticity of the mutant stem cell. That

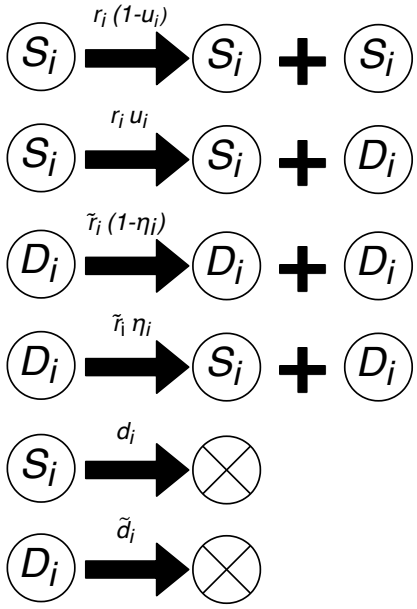
is, the probability ρ_S that as time t tends to infinity the wild type stem cells have all either died or differentiated and the mutant type stem cells remain for a given de-differentiation rate η.

3.2.1 A Discrete Moran Model

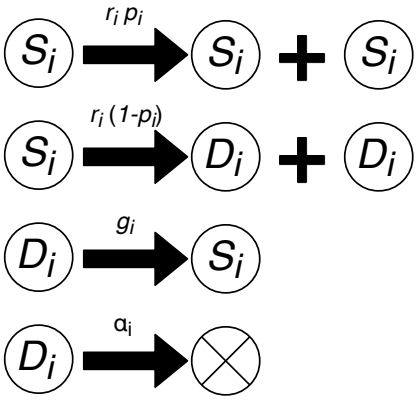
In [64] the authors consider modeling the birth-death process by way of a Moran model. In particular, the authors consider two types of cells, stem cells (S_i) and differentiated cells (D_i) where $i \in \{1, 2\}$ denotes whether the cells in question are wild-type ($i = 1$) or mutant-type ($i = 2$). Cells can react and transition between compartments according the following rules (summarised in Figure 3.1a). Stem cells undergo division at a rate r_i. There is a probability u_i that the stem cell produces both a stem cell S_i and a differentiated cell D_i during division. Similarly, with probability $(1 - u_i)$, the stem cell produces two stem cells as the result of this division. Moreover, differentiated cells can reproduce as well at a rate $\tilde{r}_i$. There is a probability η_i that this division is asymmetric, producing one stem cell S_i and one differentiated cell D_i, and a probability $(1 - \eta_i)$, that the division is symmetric, producing two differentiated cells D_i. Similarly, with probability d_i and $\tilde{d}_i$, the stem cells and differentiate cells die.

Note that in each of these events the population of stem cells and differentiated cells changes by at most 1 in either direction. Moreover, as births and deaths of cells are assumed to occur instantaneously, the state of the system at any point in time is determined entirely by the state at the previous time. Hence, the particular stochastic process being considered is Markovian. Further, the authors assume that each individual compartment has constant population size across wild and mutant types: i.e. the number of stem cells and the number of differentiated cells remain static, merely what proportion of these stem or differentiated cells are wild or mutant type is the variable.

It is important to note that in [64], the model is developed to assume the S_i and D_i compartments correspond to positive and negative biomarker cells. Current biomarkers used do not necessarily correspond directly to stem or non-stem cells. In particular, negative-biomarker (or differentiated-analogue) cells can still be observed to proliferate, as represented in the model. In reality it is often assumed that the only cells capable of driving the growth and invasive potential of a tumour are stem cells and hence differentiated cells cannot proliferate to any meaningful degree. However, since biomarkers are imperfect, some stem-like cells may be classified as differentiated [114]. This is a natural consequence of imperfect biomarkers, but is worth mentioning explicitly to justify why differentiated cells are allowed to proliferate in the model represented in Figure 3.1a.



(a) The suite of reactions captured by the Mahdipour-Shirayeh model. Differentiation and de-differentiation events connect the selection dynamics between the two niches.



(b) The suite of reactions captured by the Wodarz model. Stem cell division is always assumed to be symmetric, producing either two new stem cells or two progenitor cells.

Figure 3.1: In both models the differentiation, de-differenation, and death events corresponding to a four-compartment model where S_i and D_i are the stem-cells and differentiated cells for the wild-type ($i = 1$) and mutant-type ($i = 2$) respectively.

Let N_S denote the size of the stem cell compartment, N_D the size of the differentiated compartment, and $N = N_S + N_D$ the total number of cells considered. Further, we are concerned primarily with the invasion of mutant types; that is, when the number of S_2 or D_2 cells is at a maximum (equal to N_S or N_D). Hence we let n_S and n_D represent the number of S_2 or D_2 cells. Therefore, the number of S_1 or D_1 cells is given by $N_S - n_S$ and $N_D - n_D$, respectively. Let $W_S^+(n_S, n_D), W_S^-(n_S, n_D), W_D^+(n_S, n_D)$, and $W_D^-(n_S, n_D)$ represent the transition probabilities corresponding to an increase or decrease by one (represented by the $+$ or $-$ in the superscript) in the number of stem cells or differentiated cells (represented by the S or the D in the subscript). Then, if $1/N$ is the duration of each time step in this model, one can represent the following master equation for the probability density function

of the stochastic process as follows [64]:

$$\begin{aligned}\frac{1}{N}\frac{\partial p(n_S,n_D;t)}{\partial t} &= W_S^+(n_S-1,n_D)\,p(n_S-1,n_D;t) + W_S^-(n_S+1,n_D)\,p(n_S+1,n_D;t)\\ &+ W_D^+(n_S,n_D-1)\,p(n_S,n_D-1;t) + W_D^-(n_S,n_D+1)\,p(n_S,n_D+1;t)\\ &- (W_S^+(n_S,n_D) + W_D^+(n_S,n_D) + W_S^-(n_S,n_D)\\ &+ W_D^-(n_S,n_D))\,p(n_S,n_D;t)\,, \end{aligned} \tag{3.1}$$

where the transition probabilities are given as follows:

$$W_S^+(n_S,n_D) = (r_2\,(1-u_2)\,n_S + \tilde{r}_2\,\eta_2\,n_D)\left(\frac{N_S-n_S}{N_S}\right) \tag{3.2}$$

$$W_S^-(n_S,n_D) = (r_1\,(1-u_1)\,(N_S-n_S) + \tilde{r}_1\,\eta_1\,(N_D-n_D))\left(\frac{n_S}{N_S}\right) \tag{3.3}$$

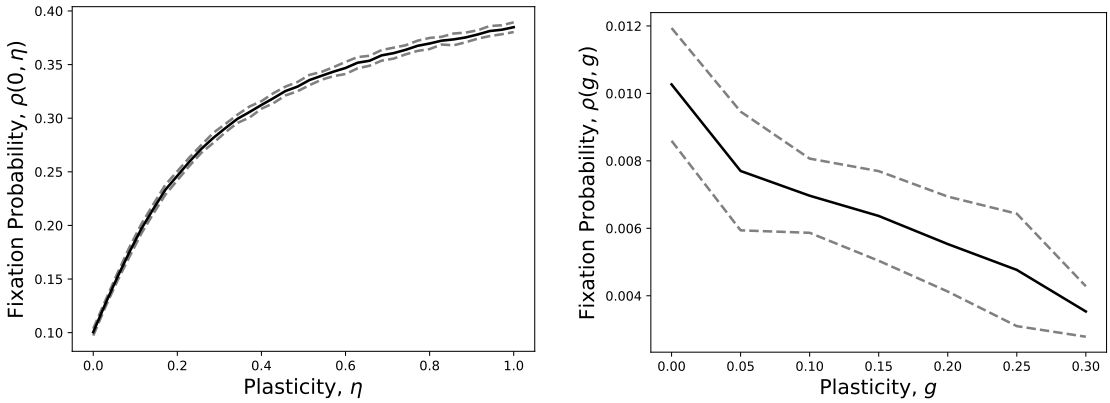
$$W_D^+(n_S,n_D) = (\tilde{r}_2\,(1-\eta_2)\,n_D + r_2\,u_2\,n_S)\left(\frac{N_D-n_D}{N_D}\right) \tag{3.4}$$

$$W_D^-(n_S,n_D) = (\tilde{r}_1\,(1-\eta_1)\,(N_D-n_D) + r_1\,u_1\,(N_S-n_S))\left(\frac{n_D}{N_D}\right). \tag{3.5}$$

The results of Figure 3.2a clearly demonstrate that fixation probability increases as plasticity of the mutant increases. In particular, when the resident wild type cells were given no plasticity and the mutant type cells plasticity was treated as a variable it was observed that increasing the mutant types plasticity gave these mutant types a greater invasive potential when placed in a population dominated by wild types population.

3.2.2 A Continuous Gillespie Model

On the other hand, in [109] the author aims to describe the dynamics of stem cell invasion not merely by considering births and deaths and the selective pressures therein, but by directly considering the positive and negative feedbacks that control differentiation and de-differentiation in the stem cell hierarchy. To this end, the author constructs multiple systems of differential equations that model the behaviour of stem cells and differentiated cells under various assumptions. The key differences between this paper and [64], are in the details of these models. In particular, the author does not limit themselves to a constant population size of stem cell and differentiated cell compartments. Moreover, the author does not explicitly allow differentiated cells to proliferate, as the categories S_i and D_i in this model do directly correspond to stem and differentiated cells directly, and not by way



(a) The Discrete Moran Model (b) The Continuous Gillespie Model

Figure 3.2: This Figure shows the results of a simulation of both the discrete Moran model from [64] and the continuous Gillespie model from [109] where the wild-type differentiated cells are completely non-plastic. The figures show the contrasting results where in 3.2a the fixation probability increases as mutant-type plasticity increases and in 3.2b the fixation probability decreases as mutant-type plasticity increases. The figures are recreated from similar figures in [64, 109]

.

of a biomarker analogue as in [64]. As before, the author considers a four compartment model where S_i represents the stem cells and D_i represents the differentiated cells:

$$\begin{aligned} \frac{\mathrm{d}S_i}{\mathrm{d}t} &= \hat{r}_i\, S_i\,(2\,p_i - 1) + g_i\, D_i \\ \frac{\mathrm{d}D_i}{\mathrm{d}t} &= 2\,\hat{r}_i\, S_i\,(1 - p_i) - \alpha_i\, D_i - g_i\, D_i \end{aligned} \tag{3.6}$$

where $i \in \{1, 2\}$ represents the wild type ($i = 1$) and mutant type ($i = 2$), as before.

While the author of [109] does not frame the model as a reaction network, it is not hard to recover the set of reactions from the differential equations. These are presented in Figure 3.1b.

One should note that if $p_i \neq \frac{\alpha_i - g_i}{2\,\alpha_i}$, then the only equilibrium is $S_i = D_i = 0$. Otherwise, the entire set $[S_i, D_i] = [S_i, (r_i/\alpha_i)\, S_i]$ is an equilibrium. These results do not coincide with what is observed in experiments. The author argues that feedback on the division rate, self-renewal probability, and de-differentiate rate occurs in order to maintain finite-size cell populations. To model this, it is assumed that $\hat{r}_i$, p_i, and g_i are decreasing functions of the

cell population sizes in the following way:

$$\hat{r}_i = \frac{\hat{r}'_i}{1 + h_{i,1}\, D_i^{k_{i,1}}}, \quad p_i = \frac{p'_i}{1 + h_{i,2}\, D_i^{k_{i,2}}}, \quad g_i = \frac{g'_i}{1 + h_{i,3}\, S_i^{k_{i,3}}}$$

In this case, the model permits an equilibrium point internal to the first quadrant that is stable when $p_i > \frac{\alpha_i - g_i}{2\,\alpha_i}$.

To numerically establish the fixation probability the following experiments were run. The author begun by placing S_1 and D_1 at the equilibrium values and then introduced a single mutant stem cell, $S_2 = 1$ and $D_2 = 0$. Gillespie simulations were then performed [34] for many realisations (more than 10^8). The fraction of simulations in which the mutants fixated was recorded and from this the fixation probability was estimated. This was done for various values of de-differentiation rates $g = g_1 = g_2$.

In contrast to the results of the discrete Moran model in Figure 3.2a, Figure 3.2b shows that the fixation probability of a neutral mutant decreases when the de-differentiation rate increases in the continuous Gillespie model.

3.3 Reconciliation of Previous Results

In the Wodarz model there are four parameters considered for each cell-type (mutant or wild) $(p_i, \hat{r}_i, \alpha_i, g_i)$ while in the Shirayeh model there are six $(r_i, \tilde{r}_i, d_i, \tilde{d}_i, \eta_i, u_i)$. The biggest difference between these parameters is that in [109] the numbers p_i and r_i are both functions of D_i, and g_i is a function of S_i.

Moreover, [109] considers four reactions between stem cell and differentiated cell compartments

$$S_i \to S_i + S_i \ , \ S_i \to D_i + D_i \ , \ D_i \to S_i \ , \ D_i \to \emptyset.$$

Whereas in [64] there are six

$$S_i \to S_i + S_i \ , \ S_i \to S_i + D_i \ , \ D_i \to D_i + D_i \ , \ D_i \to D_i + S_i \ , \ S_i \to \emptyset \ , \ D_i \to \emptyset.$$

However, all the reactions from [64] can be recovered in the Wodarz model [109]. In particular, in order for stem cells to die they must first differentiate and then the differentiated cells must both die. In order to obtain asymmetric stem cell division, the stem cells must first divide into two differentiated cells, then one of the differentiated cell must de-differentiate back into a stem cell. Suppose a differentiated cell first differentiates

into a stem cell and then the resultant stem cell symmetrically differentiates. Therefore, the differentiated cell has symmetrically divided. Further, suppose one of those resultant differentiated cells de-differentiates back into a stem cell, then the differentiated cell has asymmetrically divided. In this way all the reaction dynamics of the model from [64] have been recovered by multiple reactions from the model by [109].

This realisation lends credence to the theory that the models should, hopefully, predict similar qualitative dynamics. To that end, we begin by creating differential equations via the reactions given in Figure 3.1a and compare them to the differential equations presented in [109]. By making the mass-action assumption, we derive

$$\begin{aligned} \tfrac{\mathrm{d}S_i}{\mathrm{d}t} &= \tilde{r}_i\,\eta_i\,D_i + r_i\,(1-u_i)\,S_i - d_i\,S_i \\ \tfrac{\mathrm{d}D_i}{\mathrm{d}t} &= \tilde{r}_i\,(1-\eta_i)\,D_i + r_i\,u_i\,S_i - \tilde{d}_i\,D_i. \end{aligned} \tag{3.7}$$

Contrasting (3.7) with (3.6) it is not hard to derive the following relations. To frame the Wodarz model [109] in terms of parameters from [64],

$$\begin{aligned} p_i &= 1 - \frac{r_i\,u_i}{2\,(r_i - d_i)} \\ \hat{r}_i &= r_i - d_i \\ \alpha_i &= \tilde{d}_i - \tilde{r}_i \\ g_i &= \tilde{r}_i\,\eta_i \end{aligned}$$

conversely, framing the model in [64] in terms of parameters from the Wodarz model [109] we leave d_i and $\tilde{d}_i$ as free parameters, then

$$\begin{aligned} r_i &= d_i + \hat{r}_i \\ \tilde{r}_i &= \tilde{d}_i - \alpha_i \\ \eta_i &= \frac{g_i}{\tilde{d}_i - \alpha_i} \\ u_i &= 2\frac{\hat{r}_i\,(1-p_i)}{d_i + \hat{r}_i}. \end{aligned}$$

Hence the qualitative dynamics of (3.7) are dynamically equivalent to those of (3.6), under a renaming of parameters. Moreover, these formulae allow us to more carefully compare the two systems predictions under differing notations and parameter choices, as the formulae provide a way to transcribe parameter values and notational choices from one system into the notation of the other. In particular, note that the two plasticity parameters

g_i and η_i can be compared directly, as they are just linear, increasing functions of one another. Hence an increase in g_i is analogous to an increase in η_i (and vice versa). Moreover, $\eta_i = 0$ is analogous to $g_i = 0$.

Negative feedback on production rates is modeled in the model put forth by [64], but not as explicitly as in [109]. In particular, the last factor in the transition probabilities of [64] provide this bias. If n_S is high, or the number of stem cells is close to maximum, then the transition probability W_S^+ is near zero, due to the $(N_S - n_S)/N_S$ factor at the end. Hence when the number of mutant stem cells is high, the probability of making more mutant stem cells is lowered (this mimics the behaviour observed in the $g_2(S_2)$ function in the Wodarz model [109]). Similarly the n_S/N_S term in the W_S^- transition probability creates pressure where creating more wild-type stem cells is less likely in the presence of many wild-type stem cells (similar to the $g_1(S_1)$ function in the Wodarz model [109]). Analogous behaviour occurs in the $W_D^{\pm}$ probabilities, mimicking the behaviour of the $\hat{r}_i$ and p_i functions decreasing as D_i increases.

In the finite population size model, negative feedback on (de–)differentiation rates and production rates are governed by the select pressures inherit in finite population sizes, whereas in the unbounded population size model these same feedback mechanisms are modeled by non-constant, non-linear reaction rates.

A more subtle difference between the two systems is due, in part, to a lack of clarity in notational differences. In the original presentation of [109] the i subscripts were not present (though different values of the same parameter between compartments was implied in the figure descriptions). Among other things, this obscures the fact that g (the parameter representing de-differentiation, or plasticity) is being altered at the same rate for both the mutant-type population and for the resident-type population. Hence, the plots in Figure 3.2b really plot the fixation probability $\rho(g_1, g_2)$ along the diagonal cross-section of the first quadrant, that is the plots show

$$\frac{\partial}{\partial g}\left(\rho(g_1, g_2)\big|_{g_1=g_2=g}\right) < 0$$

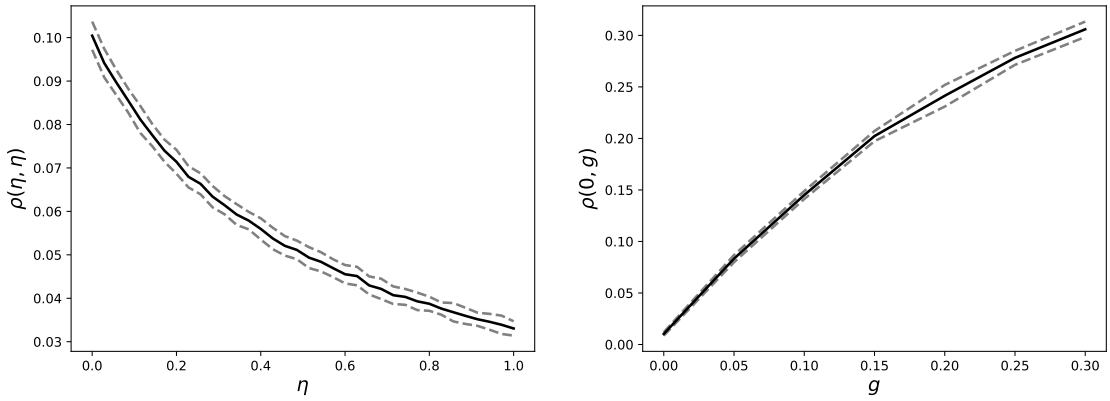
whereas the plots in Figure 3.2a show

$$\frac{\partial}{\partial \eta_2}\left(\rho(\eta_1, \eta_2)\big|_{\eta_1=0}\right) > 0.$$

The tacit assumption in [64] is that part of the phenotypic difference between the mutant stem cells and the resident stem cells is that resident stem cells have no plasticity and mutant stem cells do have plasticity. The analysis then focuses around qualifying the

effects of this plasticity. Contrast this, the tacit assumption in [109] is that the mutant and resident stem cells are phenotypically different in some other way and share the same plasticity rate.

To conclude, we wished to consider measuring $\rho(\eta, \eta)$ for $0 \leq \eta \leq 1$ in the discrete Moran model (that is, the fixation probability from [109] in the model by [64]) and to measure $\rho(0, g)$ for $0 \leq g \leq 1$ in the continuous Gillespie model (that is, the fixation probability from [64] in the model by [109]). In particular, as can be seen in Figure 3.3, we have recovered the results of the contrasting models. That is, in Figure 3.3a we see that $\rho(\eta, \eta)$ is a decreasing function of η as was seen in [109] (Figure 3.2b) and in Figure 3.3b that $\rho(0, g)$ is an increasing function of g as was seen in [64] (Figure 3.2a).



(a) The Discrete Moran Model

(b) The Continuous Gillespie Model

Figure 3.3: The figure indicates that when varying both the wild-type and mutant-type plasticity parameters at the same rate in the discrete Moran models, the fixation probability function is negatively sloped. Similarly, when increasing only the mutant-type plasticity parameter and keeping the wild-type plasticity parameter at zero in the continuous Gillespie model, the fixation probability function increases. Note that these results are similar to what was observed in Figure 3.2, just with the modeling framework switched.

3.4 Further Results

The results of Section 3.3 demonstrate that it is vitally important to consider how a change in plasticity in a mutant type affects a change in plasticity of a wild type, if at all. In particular, the authors of [64] and [109] were effectively answering different research

questions. In [64] the authors were concerned with what the effect of increased plasticity in a mutant-type would have on fixation probability assuming the wild-type stayed the same. This corresponds to a case where the mutant-type gains increased plasticity as part of the mutation. Moreover, this assumes that this increased plasticity in the mutant type does not have any effects on the plasticity in the resident type. Contrarily, in [109] the authors assume that the increased plasticity is common to both the mutant and wild type stem cells. This could be the case where an increased mutant-type plasticity affects the wild-type stem cells by some kind of increased selective pressure, or via a change in the micro-environment thus influencing the wild-type to also become more plastic.

A natural extension to the results of Section 3.3 is to consider decoupling the plasticity parameters all together and contrast the behaviour of the models over this larger parameter range. For computational reasons we proceed by examining the discrete Moran model and derive non-monotone (see Figure 3.4) behaviour before concluding in Section 3.4.2 that this behaviour can not be observed in the continuous Gillespie case.

We consider varying both η_1 and η_2 (the plasticity of the wild-type and mutant-type cells) independently of one another. Since η_1 and η_2 are probabilities, we varied them over the closed unit square $[0,1] \times [0,1]$ in a 36×36 grid. At each point in the grid we calculated the fixation probability of the mutant-type stem cells. Further details of the numerical simulation can be found in Appendix A.1. Note that allowing $\eta_1 > 0$ corresponds to de-differentiation of healthy, wild-type tissue. Mathematically, this is motivated by the original model strategy of [109] but biologically this choice is reasonable in light of experimental evidence that de-differentiation is present not only in malignant tissue, such as was studied in [117], but in healthy tissue as well [100, 15]. Moreover, especially in the case of the discrete Moran model, we wish to reemphasise that the compartment labels of stem and differentiated cells can be thought of as positive and negative stem biomarker cells, in which case the assumption that $\eta_1 > 0$ could be understood to represent imperfect sorting of such compartments by modern biomarkers [114].

The purpose of the experiment was to further investigate the exact interplay between plasticity in the two compartments on the fixation probability. However, there are 6 other parameters to consider, $r_i, \tilde{r}_i$, and u_i. In order to investigate the direct effect of a change in plasticity on the fixation probability in isolation, we considered the case where relative fitness was the same in the stem and differentiated cells (that is, $r_i = \tilde{r}_i$).

We further considered experiments where the relative fitness of the mutant-type cells ($r_2 = \tilde{r}_2$) were varied along with the de-differentiation probabilities (η_1 and η_2). Moreover, to further analyse the stability, we considered varying the differentiation probability (u_2) of the mutant-type cells along with the de-differentiation probabilities.

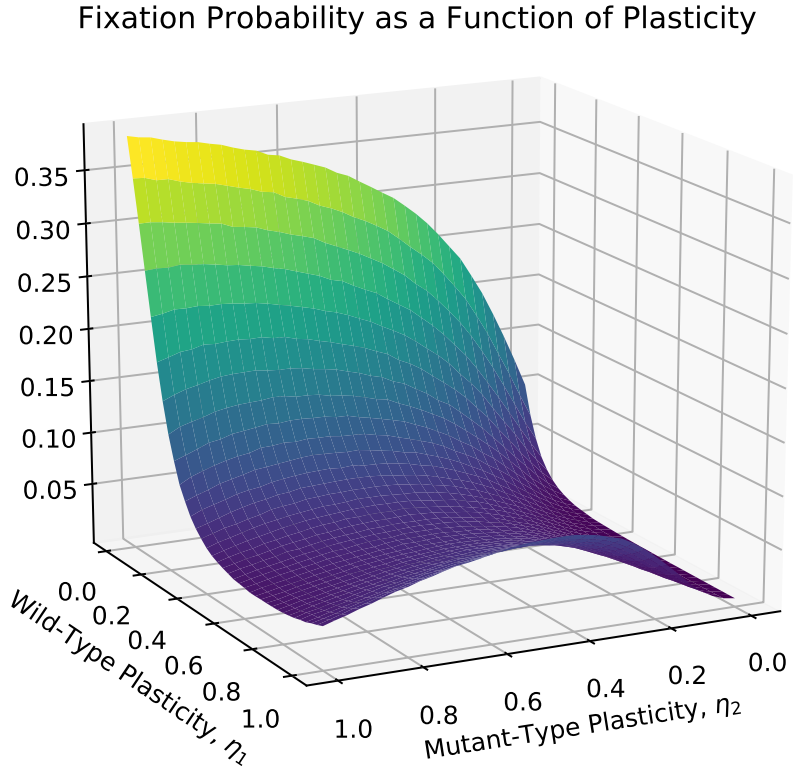


Figure 3.4: A plot demonstrating the presence of a saddle point in the fixation probability as a function of both de-differentiation parameters. It indicates that increasing the mutant-type de-differentiation parameter does not always result in increased fixation of mutant-type stem cells, even when keeping the wild-type de-differentiation parameter constant.

In the case of the neutral mutant (where $r_2 = \tilde{r}_2 = r_1 = \tilde{r}_1$ and $u_2 = u_1$), the results are summarised in Figures 3.4, 3.5, and 3.7. The plot in Figure 3.4 indicates the fixation probability as a function of both de-differentiation parameters. It indicates that in general, increasing η_2, the de-differentiation probability of the mutant cells, increases the fixation probability. Moreover, this fixation probability is maximal for $(\eta_1, \eta_2) = (0, 1)$. However, it also shows that for large enough η_1, increasing η_2 will not always result in an increased fixation probability. In the extreme case, where $\eta_1 = 1$, increasing η_2 initially increases the fixation probability, but eventually this fixation probability decreases as η_2 is closer to 1.

This result can be seen more clearly in Figure 3.5, where the fixation probability is plotted for constant values of η_1 as a function of varying η_2. When $\eta_1 = 0$ the fixation probability is a concave down, increasing, saturating function of η_2. However, for $\eta_1 = 0.2$ the fixation probability is no longer strictly increasing and undergoes a change in concavity early on. As η_1 is increased further the fixation probability is no longer a strictly increasing function of η_2 but initially increases and eventually decreases as η_2 becomes larger. For larger η_1 the fixation probability function decreases over a larger range of η_2 and decreases more drastically. Conversely, as η_1 is increased, the average fixation probability of the mutant-type cells is increased. That is, for $\eta_1 = 0.8$, the maximum of $\rho(0.6, \eta_2)$ is around 0.045 (or 4.5%), but the maximum of $\rho(1, \eta_2)$ is closer to 0.075 (or 7.5%). This can be recovered by recognising the saddle-like behaviour of Figure 3.4. This symmetry can be explained similarly, as η_1 increases, more wild-type differentiated cells are de-differentiating into stem-cells. This removes selective pressure from mutant-type differentiated cells, allowing these mutant-type cells to grow larger and, for appropriate values of η_2, de-differentiate back into mutant-type stem cells.

This non-monotonic fixation probability function is an interesting phenomenon. This implies that for particular parameter sets, it is actually worse for mutant type stem cell fixation to increase the rate at which mutant differentiated cells de-differentiate into mutant stem cells. This is non-obvious, as one would assume increasing the pool of mutant stem cells, by decreasing the pool of mutant differentiated cells, is vital to the fixation of the mutant stem cells. However, by decreasing the pool of mutant differentiated cells, the wild type differentiated cells have a selective pressure removed and can see an increased response in growth. For large enough η_1, these wild type differentiated cells can de-differentiate back into wild type stem cells and thus provide a selective disadvantage to the mutant stem cells. This is an important phenomenon to fully understand. Much of the results of Section 3.4.1 is to indicate that this phenomenon persists over a wide range of parameter space and is not the result of a particular, lucky set of parameters.

The original paper by [64] considered only a particular constant η_1 value while allowing η_2 to vary. Thus far we have considered various constant η_1 values and noted interesting

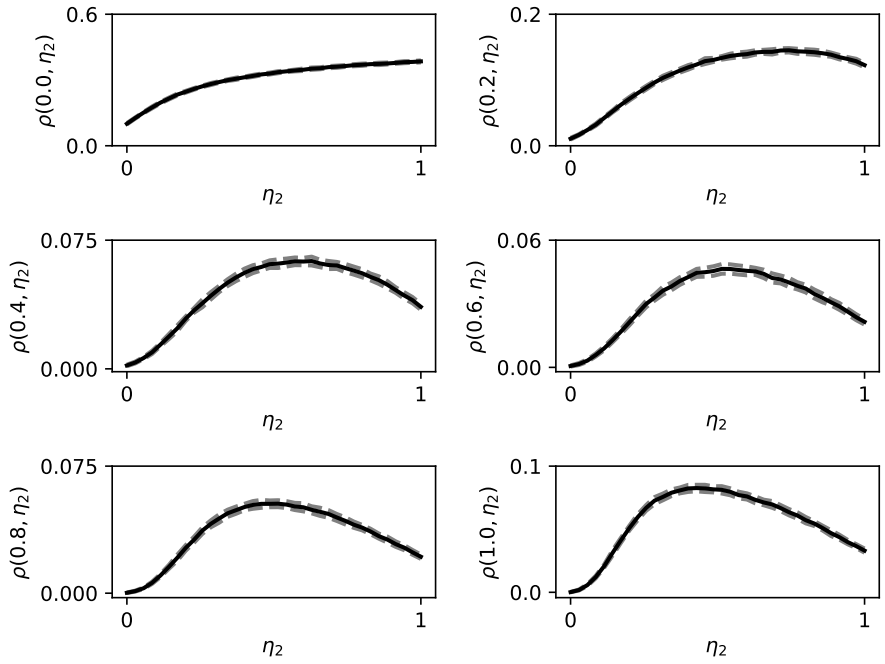


Figure 3.5: A stackplot demonstrating the fixation probability as a function of the de-differentiation probability η_2 for various choices of η_1. The solid line is the calculated fixation probability and the dashed lines indicate one deviation by the standard error of the mean.

dynamics occurring. In the paper by [109], they considered varying η_1 as a linear function of η_2 (in their particular case, η_1 was identically η_2). In the next set of experiments we considered varying η_1 as other linear functions of η_2.

In the first plot in Figure 3.6, we note that, compared to the $\eta_1 = \eta_2$ case, in general the effect is a decrease in fixation probability for η_2 smaller than, approximately, 0.55, and an increase in fixation probability as η_2 reaches its maximum. In this scenario, the initial decrease in fixation probability is intuitive. A more plastic wild-type compartment would allow more wild-type stem cells to be created and hence result in a selective pressure on mutant-type stem cells, resulting in less probable fixation for the mutant cells. However, when η_2 is sufficiently large, than the difference between η_2 and η_1 is less pronounced. This is similar to the theme observed in Figure 3.5. For large enough values of η_2 the selective pressures from the more plastic wild-type stem cells are reduced. Moreover, in these scenarios the smaller de-differentiation value compartment has a selective advantage in production of differentiated cells, and both differentiated and stem cell compartments are necessary for fixation.

In the second plot in Figure 3.6, we note that in general for η_1 smaller than η_2 we, mostly, observe that the fixation probability is increased (compared to the $\eta_1 = \eta_2$ case). This is intuitive, when η_1 is smaller, wild-type stem cells are being produced less quickly than mutant-type stem cells. Hence mutant-type stem cells have a selective advantage for invading. However, when η_2 is near 1.0 we observe a very slight dip in fixation probability. In these scenarios, the mutant cells are almost always de-differentiating. Hence the S_2 compartment is filling faster while the D_2 compartment is draining faster. Even though $\eta_1 < \eta_2$, η_1 is still appreciably non-zero, hence the D_1 compartment has a selective advantage due to the small size of the D_2 compartment. Therefore, both wild types perform well and fixate with slightly more frequency.

In the third and fourth plots of Figure 3.6, the situation behaves similarly to the first two plots. When $\eta_1 > \eta_2$ there is an initial advantage for the wild-type cells that disappears as η_2 becomes sufficiently large. Contrarily, for $\eta_1 < \eta_2$ there is an initial advantage for the mutant-type cells that disappears only when η_2 becomes sufficiently large. What is unique about these two cases is the loss of monotonicity. In each experiment in the third plot of Figure 3.6, increasing the plasticity parameter η_2 also increases η_1, but in this experiment η_1 initially increases fast enough that the selective disadvantage for wild-type cells imposed by larger η_2 is offset by the selective advantage for wild-type cells granted by larger η_1. However, past a certain point the benefit in fixation success for the wild-type cells as a function of η_1 begins to saturate and become outpaced by the selective advantage imposed by larger η_2. The fourth plot behaves in a similar fashion to the third plot. For smaller η_1 initially the fixation probability increases. As η_2 first begins to increase, η_1 increases too

but at a slower rate. The effect of this initial slow increase in η_1 is offset by the growing of η_2, hence the fixation probability function is positively sloped. As η_2 becomes larger yet, the increase in η_1 starts to effect the evolutionary dynamics and the fixation probability starts to decrease.

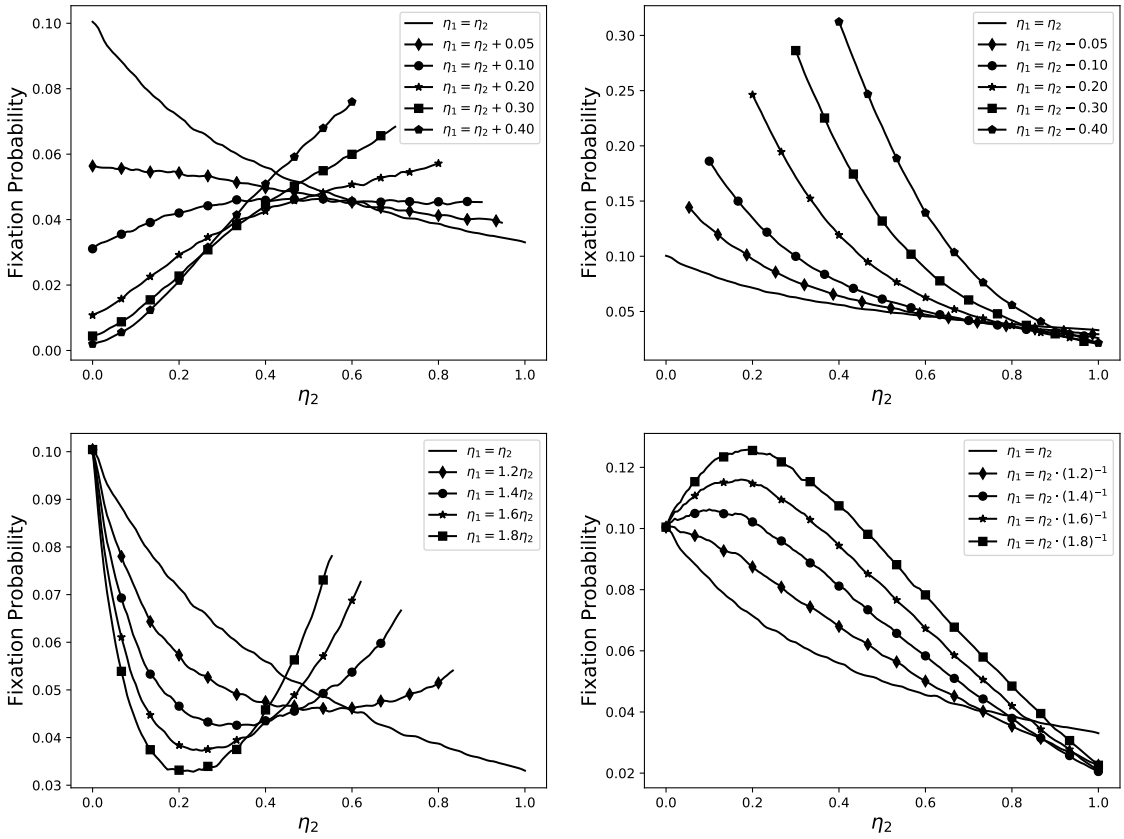


Figure 3.6: The effects on fixation probability by varying η_1 as a function of η_2.

When examining Figure 3.4, it appears that in general the fixation probability increases. Moreover, when the fixation probability is increasing, it obtains a larger value. That is, for increasing η_1 the fixation probability is, in general, decreasing. This effect raises the question that in general, in a heterogeneous population of wild-type stem cells where deviations in de-differentiation probability η_1 can be observed, what might be the expected effect on the fixation probability. To that end, we modeled the average fixation probability as a function of η_2 averaged over all values of η_1. That is,

$$\langle \rho(\eta_2) \rangle = \int_0^1 \rho(\eta_1, \eta_2)\, f(\eta_1)\, \mathrm{d}\eta_1$$

where $f(\eta_1)$ is the probability density function for η_1 in this case taken to be uniform. The result, as presented in Figure 3.7, is a non-monotone function of η_2 with a unique maximum, suggesting that in general, increasing the plasticity rate of the mutant-type stem cell will not always result in increased fixation probability, but may be harmful to the mutant-type cells.

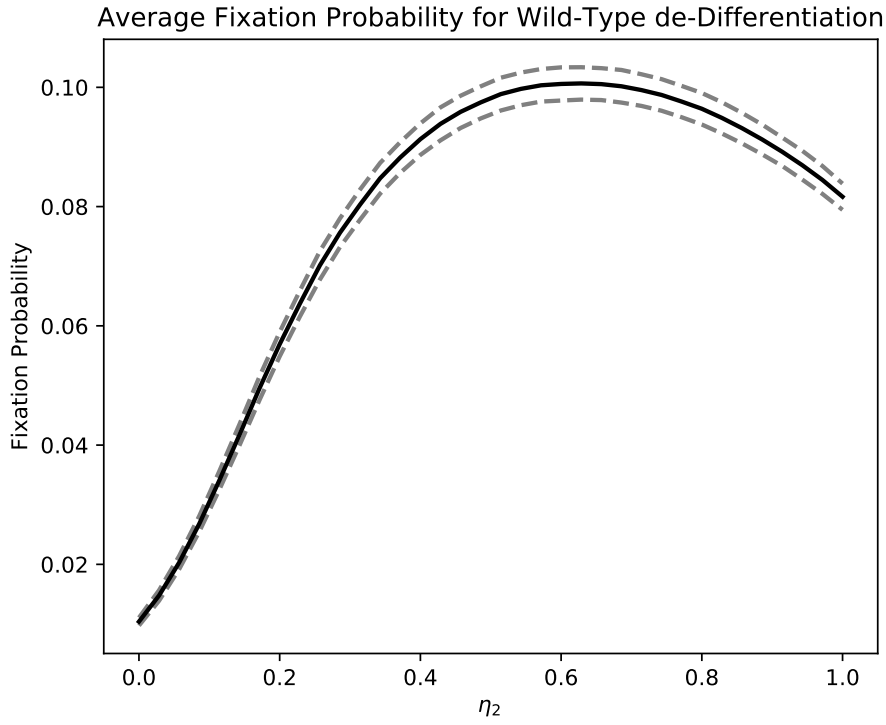


Figure 3.7: The mean fixation probability as a function of η_2 with the mean taken over the parameter η_1 assuming a uniform distribution. The dashed bars indicate a single deviation by standard error of the mean.

3.4.1 Stability Analysis

The previous results were all obtained for the neutral mutant. To demonstrate the generality of these results and their independence on particular parameter choices, we repeated the analysis for varying parameters. In particular, we varied the relative fitness of the mutant-types, $r_2 = \tilde{r}_2$, and the probability of differentiation of the mutant types, u_2. The technical details can be found in Appendix A.1.

The results of Figure 3.8 show that for general mutant increasing the relative fitness increases the fixation probability. For cases where the relative fitness is lower, the de-differentiation parameter η_2 is more important. It is only for relatively larger relative fitness levels that these fixation probability curves begin to demonstrate saturation and diminishing returns as a response to increased mutant cell plasticity. In these cases, saturation is reached quicker for larger $r_2 = \tilde{r}_2$ values. Thus even small values of η_2 are sufficient to maximise the chances of fixation.

However, for relative fitness values that are closer to neutral (i.e. $r_2 = \tilde{r}_2 = 0.75$) we recover the non-monotone behaviour. That is, when there is no substantial advantage between the wild-type and mutant-type stem cells, fixation probability is more complicated than just maximising the plasticity. In these cases, as seen in Section 3.4, the behaviour of the stem cells is governed by competing evolutionary pressures and results in a greater need to balance parameters than to maximise parameters.

In Figures 3.9 the probability parameter u_2 was varied. This parameter corresponds to the probability a mutant stem cell will differentiate. These results corroborate the claim that the non-monotonicity of the fixation probability function is a general phenomenon and not a result of careful parameter choice. For smaller u_2 values the fixation probability function is more right-skewed and for larger u_2 values the function is more left-skewed. In fact, for sufficiently large enough u_2 values the resulting function is monotone increasing.

The left-skew of the small u_2 functions indicates that when the mutant cells do not differentiate often, increasing the de-differentiation parameter is worse for fixation. This could be because in order for the wild-type cells to fixate they must selectively eliminate all mutant stem cells and differentiated cells. Hence, if there are few mutant differentiated cells to begin with, removing them from the micro-environment provides a selective advantage to the wild-type cells. When u_2 is larger, there are enough differentiated cells in the micro-environment that allowing some to de-differentiate (equivalent to $0 < \eta_2 < 1$) provides a selective advantage for the mutant-type cells. However, allowing too many mutant differentiated cells to de-differentiate (equivalent to $\eta_2 \approx 1$) results in the differentiated cell pool depleting too quickly and the resultant selective advantage for the wild-type cells

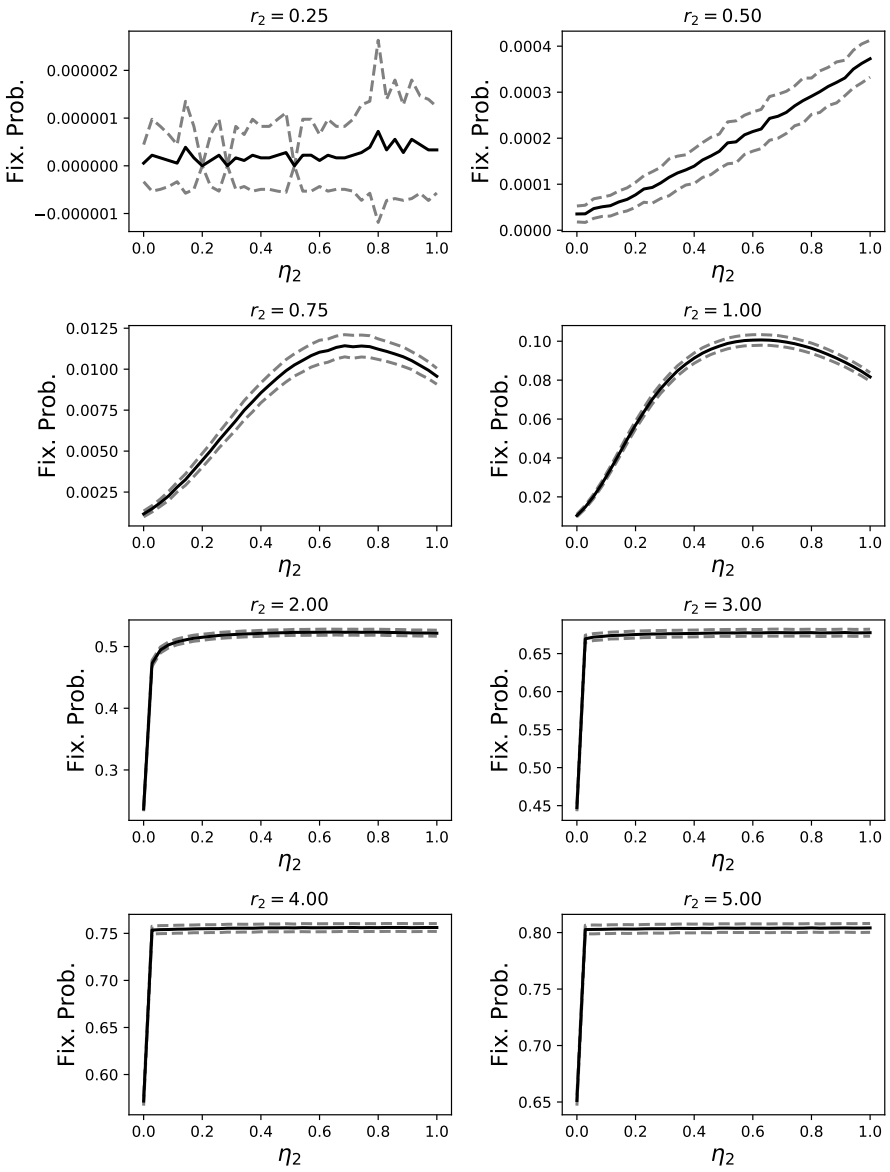


Figure 3.8: Average fixation probability for various relative fitness levels of the mutant-type cells, $r_2 = \tilde{r}_2$. The dashed lines indicate one deviation of the standard error of the mean. The average was taken over wild-type de-differentiation probability η_1.

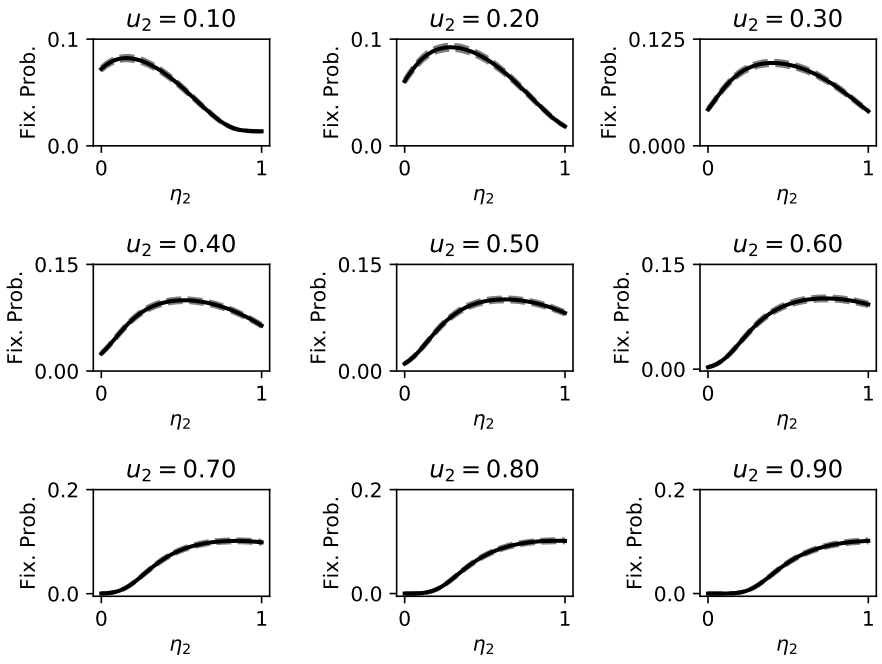


Figure 3.9: Average fixation probability for various values of u_2, the probability the mutant type stem cell differentiates. The dashed lines indicate one deviation of the standard error of the mean. The average was taken over wild-type de-differentiation probability η_1.

provides a selective disadvantage to the mutant-type cells. For sufficiently large u_2 as in the plots for $u_2 = 0.8$ and $u_2 = 1$, mutant stem cells are almost always de-differentiating, hence the only way to refill the mutant stem cell pool is to increase the rate of de-differentiation. For these scenarios the bottle-neck for fixation is recovering enough stem cells to create a selective disadvantage on the wild-type cells. Hence, increasing η_2 is always the most evolutionary effective means of increasing fixation.

3.4.2 Monotone Response Curves

The non-monotonicity observed in Figure 3.4, among others, can be seen as arising from the competing reactions $D_i \to^{\tilde{r}_i \eta_i} S_i + D_i$ and $D_i \to^{\tilde{r}_i (1-\eta_i)} D_i + D_i$. Both reactions are beneficial to the species, as increasing both the differentiated and stem cell compartment is necessary to achieve fixation. As such, one could imagine the non-monotonicity as arising out of an implicit choice being made to further fixate by either creating a new stem cell, with the expectation that this will result in later increases in the differentiated cell compartment, or, instead, by creating a new differentiated cell immediately. This, evidently, is a process unique to the Moran model outlined in Figure 3.1a that has no obvious analogue in the continuous Gillespie model outlined in Figure 3.1b. In particular, in the continuous Gillespie model differentiated cells are not permitted to proliferate, and while the two models produce equivalent dynamical systems, up to a change of parameters, the particulars of the implementation of both are not equivalent due to the differing process by which the models enforce selective pressure (in particular, the selective pressure from the Moran model is a result of fixed population sizes whereas in the Gillespie model selective pressure is governed by feedback on reaction rates). As a result, the continuous Gillespie model cannot achieve non-monotone fixation probability curves.

To illustrate how this behaviour is extended by the discrete Moran model, consider setting $\eta_i = 1$ and using $\tilde{r}_i$ as the plasticity parameter. In this context the reaction that allows differentiated cells to proliferate is removed, and de-differentiation is governed by the parameter $\tilde{r}_i$. Figure 3.11 demonstrates that this process results in a fixation probability curve $\rho(\tilde{r}_1, \tilde{r}_2)$ that is strictly increasing in $\tilde{r}_2$, strictly decreasing in $\tilde{r}_1$, and strictly decreasing for $\tilde{r}_1 = \tilde{r}_2 = \tilde{r}$ (as was observed in [109], seen more clearly in Figure 3.12).

Similarly, in [109], the author introduces a model using transit amplifying cells (Figure 3.10). In this model, stem cells either proliferate on their own, or differentiate into an intermediary state called transit amplifying cells. Transit amplifying cells then either de-differentiate back into stem cells, or fully differentiate into differentiated cells. The only reaction differentiated cells can undergo is death. In particular, note that this model

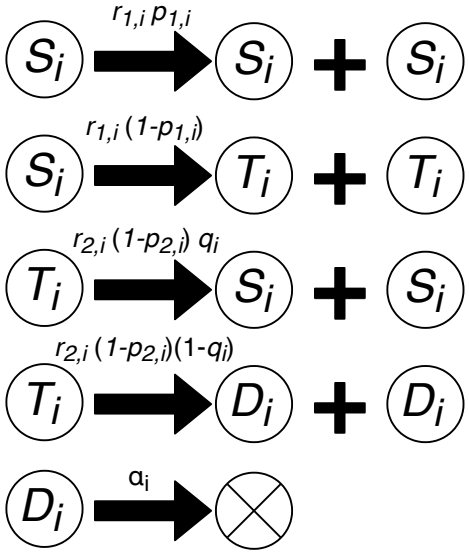


Figure 3.10: A model where stem cells first differentiate into transit amplifying cells before fully differentiating. Transit amplifying cells can then govern the de-differentiation process.

includes similar competition, based on the plasticity parameter q_i, between reactions $T_i \to^{r_{2,i}\,(1-p_{2,i})\,q_i} S_i + S_i$ and $T_i \to^{r_{2,i}\,(1-p_{2,i})\,(1-q_i)} D_i + D_i$. However, since differentiated cells, in this model, do not provide a selective advantage to the species, only one of these reactions directly increases the evolutionary fitness of the species. As a result, varying the plasticity parameter q results in strictly monotone response curves.

In effect, this demonstrates that when differentiated cells are not allowed to proliferate increasing the plasticity of mutant-type cells (or, equivalently, decreasing the plasticity of wild-type cells) results in an increase in fixation probability of the mutant-type cells. This is analogous to the results observed in [64] when $\eta_1 = 0$.

3.5 Summary

In summary, two, apparently contradictory, stochastic models of cancer stem cell behaviour were considered. Both models were questioning the invasive capacity of mutant stem cells in the presence of a pre-existing stem cell and differentiated cell strata. The models were concerned with how the phenotypic plasticity, or de-differentiation rate, affected the invasive potential of these mutant stem cells. A Moran birth-death model was considered and

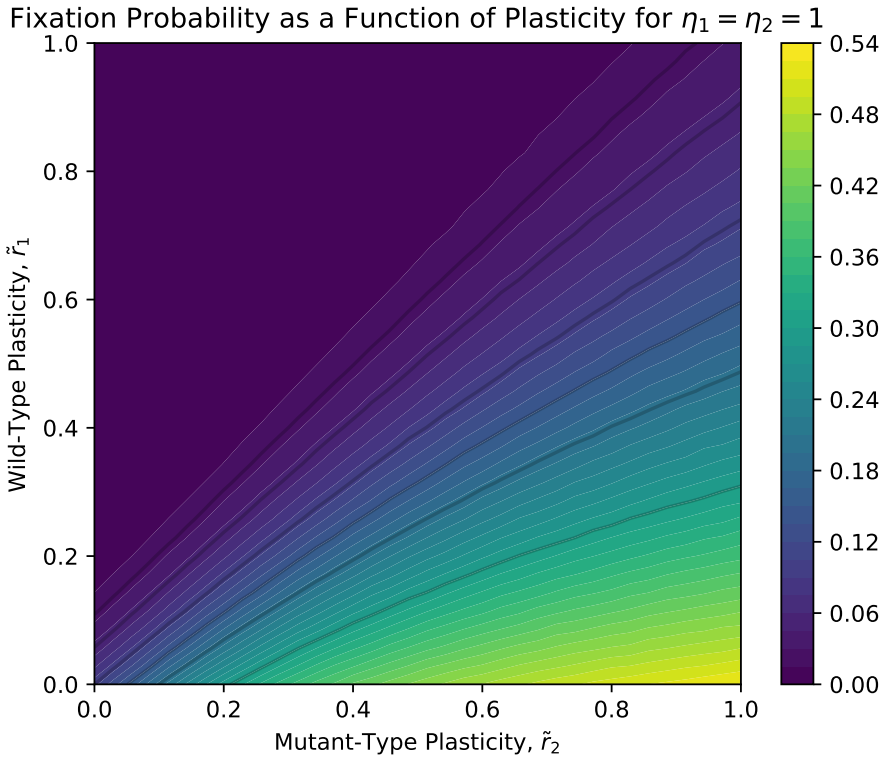


Figure 3.11: A contour plot demonstrating the fixation probability as a function of both plasticity parameters when differentiated cell propagation has been removed. The plot indicates that increasing the mutant-type de-differentiation parameter always increases the fixation probability of mutant-type cells. Likewise, decreasing the wild-type de-differentiation parameter increases the fixation probability of mutant-type cells. In contrast with Figure c3.4, no saddle point is observed.

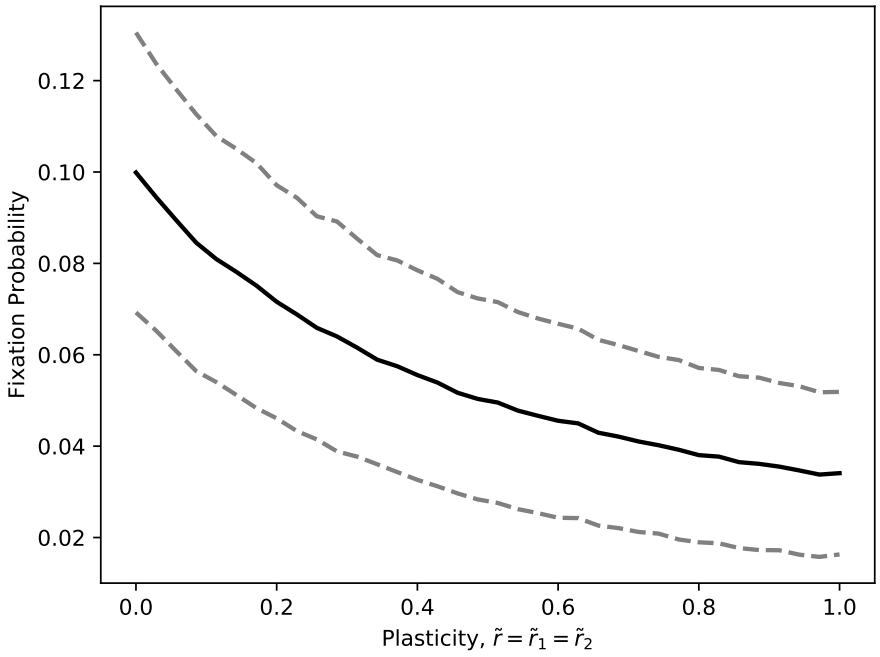


Figure 3.12: A plot indicating that when mutant and wild-type plasticity parameters are kept coupled increasing this plasticity parameter decreases the fixation probability of the mutant-type cells. Dashed lines indicate a difference of one standard error of the mean.

Gillespie simulations of a reaction network were considered. The apparent discrepancy between these models was discovered to be hidden within assumptions determining how both the resident type and mutant type plasticity rates alter. This apparent contradiction was resolved and the behaviour of the Gillespie simulation model was recovered in the Moran context, and vice versa.

It was also observed that increasing plasticity in the mutant type differentiated cells can result in both positive and negative effects for the fixation probability of the mutant stem cells, as long as differentiated cells are allowed to proliferate at some nonzero rate. This seeming contradiction was demonstrated to be stable across differing parameter sets indicating that this phenomenon may be observed in general. Further, when differentiated cells are not permitted to proliferate, then the system is entirely dependent on increasing the fixation efforts of the stem cells. Hence, when differentiated cells are not permitted to proliferate, increasing mutant plasticity results in increased mutant fixation probability. The continuous Gillespie model of [109] is an example of a stem cell model where non-monotone response curves can not be realised as a result of this inability for differentiated cells to proliferate. This elucidates a fundamental difference between the models of [64] and [109]; namely that since the former allows differentiated cells to proliferate, competitive behaviour between stem and differentiated compartments is observed that the latter model does not experience. These results indicate that while the initial contradiction between the two models has been resolved in this chapter, further differences between the two have been revealed, and explained, as a result of differing assumptions regarding the modeling of differentiated cell populations. Moreover the discrete Moran model was shown to be able to recover the non-monotone response curve behaviour of the continuous Gillespie model under a suitable change of parameters.

Chapter 4

Classification of Cells via Imaging Flow Cytometry

In this chapter we develop a method for identifying the mitotic phase of Jurkat cells. We compared the method with previously published approaches and classical machine learning techniques. The effects of the additional layers of complexity are investigated in order to benchmark the predictive accuracy of the method. A manuscript of this work is being prepared for publication.

4.1 Introduction

With techniques like imaging flow cytometry, thousands of images of cells can be acquired relatively quickly. One thing that makes these methods attractive is that the cells themselves need not be destroyed or damaged by the process. Traditionally, binning these cellular images into various classification classes can be accomplished by applying various biomarkers or fluorescent dyes to the cells [12]. Flow cytometry can then quantify the proportion of cells that satisfy the particular biological stratification of interest (for instance, the proportion of mitotic cells within the sample [8, 49]). However, while imaging flow cyotmetery itself does not damage the cell, the addition of biomarkers and dyes can negatively effect the cell in many ways. For instance, the addition of certain fluorescent dyes can cause histone dissociation, damaging the DNA of the live cell [110]. Moreover, even if the cell itself is not destroyed, the addition of these biomarkers can introduce confounding effects in the classification of cells. So while the methodology of flow cyotometry is non-destructive,

learning information about the proportion of single cells in certain biological tasks often requires the cell to be either destroyed or irreparably damaged. Thus, it is desirable to develop methodologies that do not require constant damage being done to the cells in order to perform classification.

In 2017 Blasi. et al [8] produced a methodology that uses the open source software package CellProfiler [14] combined with random forest ensembles to classify cell cycle stages in a label-free, non-destructive manner. By first employing traditional dyes and biomarkers in imaging flow cytometry to produce a ground truth, the authors then train their machine learning classifier to predict the cell cycle phase of these cells with reasonable true positive rates across all classes of interest. A benefit of this method is that while the original dataset required the addition of these dyes and biomarkers to train the algorithm, further cell data can be classified by the algorithm in a biomarker free manner. As a result, this reduces the total cost of performing cell binning. The algorithm relies upon an important step in the process known as "feature engineering", performed by CellProfiler, wherein a biologist produces a pipeline of important features to be extracted from the image. As such, the raw cell image is then transformed into a vector of descriptive features. The strength of the classifier, then, depends critically upon the correct features having been chosen at the outset. This strength also disguises a weakness of these methods: if a classifier is not performing well is it a problem with the classifier, are the wrong features being selected, or is it perhaps just a fundamental intractability of the problem? To this end, much work has been done in the direction of automated feature engineering [68, 28, 74].

Computer vision and image classification tasks have seen rapid advancements in the past five to ten years, with deep learning based approaches ushering in new benchmarks and state of the art performances on classic datasets such as MNIST, Fashion-MNIST and ImageNet [45, 17, 112]. One of the largest strengths of these approaches lie in their capacity to correctly classify the images without the need of a hand tuned, feature engineering step. Indeed, by using neural networks to perform the feature engineering step directly as a part of the classification problem, the neural network is able to not only learn how to classify the data based on the features but also how to extract the correct features for such a task. The difference between these two approaches is visualised in Figure 4.1. While training these complicated neural networks often requires powerful hardware, there are advancements being made to allow inference to be performed on incredibly low-power computing hardware [2, 21, 59]. In effect, once trained these vision based cell-sorting models can be used towards classification on much more mobile, power-friendly hardware.

To that end, we develop a methodology to detect the cell cycle phase of Jurkat cells without the need for feature engineering steps and produce a method that not only produces a higher balanced accuracy score, but also a higher true positive rate and a more diagonally

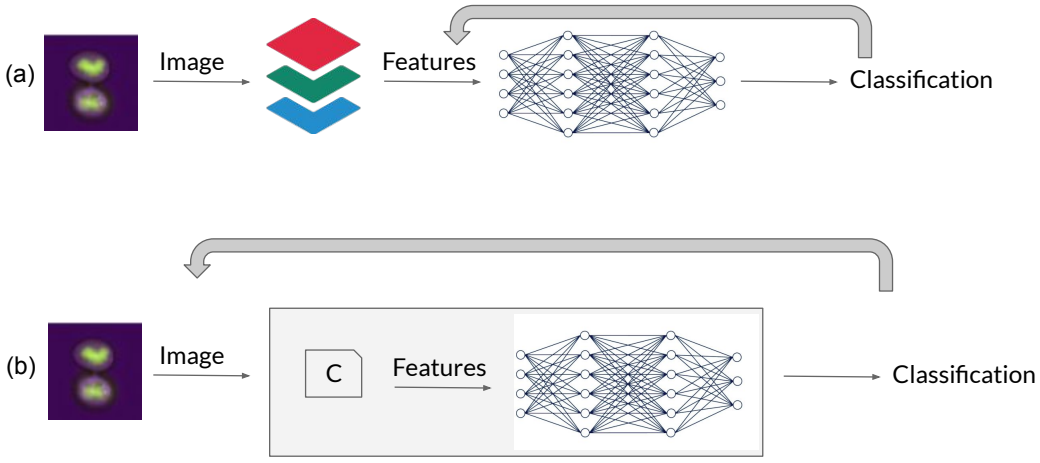


Figure 4.1: Figure demonstrating the difference in training between the manual feature extraction methods and the automatic feature extraction methods. Note the backwards grey arrow denoting the process of backpropagation during model training. In (a) features are extracted from the cell image and the feature extraction process is unaffected by backpropagation. In (b) backpropagation informs what features are being extracted as well as their role in classification.

dominant confusion matrix.

4.2 Methods

4.2.1 Imaging Flow Cytometry

Imaging flow cytometry is an experimental method for obtaining single cell images in a high-throughput, non-destructive manner. This method suspends cells in a fluid that are then injected into the instrument where the sample is focused such that one cell at a time is filtered through a laser beam. Cells that have been stained with biomarkers or other chemical dyes scatter the light in characteristic bands of wavelengths for identification. This data is then fed into the IDEAS software package for darkfield and brightfield image acquisition and ground truth labelling.

4.2.2 Experimental Data

The dataset we considered is courtesy of Blasi et al. [8]. The dataset consists of 32,255 asynchronously growing fixed Jurkat cells treated with 50 μM Nocodazole, a mitotic blocking agent, as well as a PI stain and MPM2 (mitotic protein monoclonal #2) antibody in order to provide a ground truth stratification of these cells into five groups: interphase, prophase, metaphase, anaphase, and telophase. Each of these groups represents the stage of the cell cycle that the particular cell was in during the imaging process. The images themselves are brightfield and darkfield images of the cells in a $55 \times 55 \times 2$ pixel array.

While the dataset itself is quite large, the class membership itself is very imbalanced. In Figure 4.2 this class imbalance is visualised by plotting a histogram on a log-scale. Indeed, nearly 98% of the dataset is comprised of cells in the interphase compartment. For the purposes of classification, this requires care in dealing with the metrics tracked to measure the performance of a classifier. Consider the naïve classifier that assigns a label of "interphase" for every input. Evidently such a classifier is incapable of determining the difference between cells, however if merely the accuracy of the classifier were assessed on the dataset one could erroneously assume the classifier has successfully solved the problem to a high degree of accuracy.

There are metrics other than accuracy that one can consider. One such common metric in the case of an imbalanced dataset is the so-called "balanced accuracy score". This score weights the accuracy for each label by the number of instances within that class. As a result, for the naïve classifier that only assigns a label of "interphase", a balanced accuracy score of 20% would be achieved. The balanced accuracy score is calculated by Equation 4.1 where $\mathbb{1}(\cdot)$ represents the identity function whose output is 1 if the argument is true and 0 otherwise.

$$ba = \sum_i \frac{\mathbb{1}\left(f(g(X_i)) = \lambda_i\right)}{n_{\lambda_i}} \tag{4.1}$$

For the purposes of tuning the hyperparameters of the classification methods, a validation set of 2933 images was selected from the data. In an effort to ensure that the validation data is representative of the larger dataset, the validation data was selected in a stratified manner such that the distribution of labels approximately matches the distribution of the labels in the entire dataset (as visualised in the rightmost plot of Figure 4.2).

For training the classification algorithms, we used stratified 10-fold cross validation on the remaining data. The remaining data were stratified into 10 different subsets whose distribution approximately matches the distribution of the larger dataset, as with the

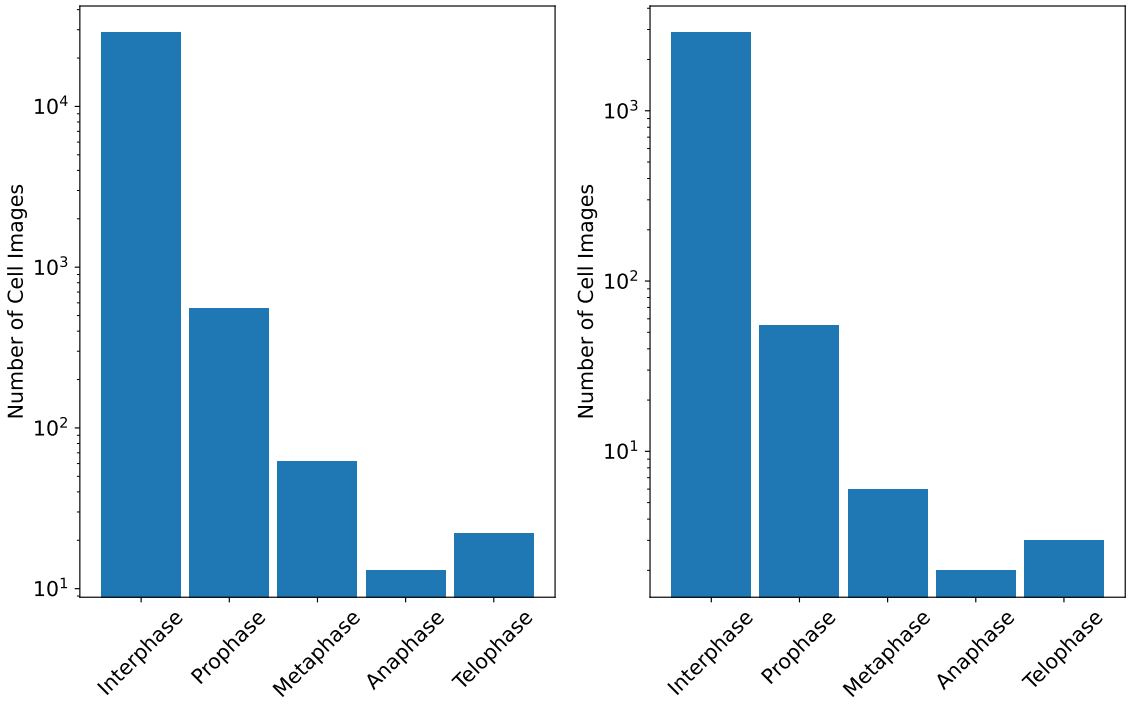


Figure 4.2: A log-histogram as an example of the imbalanced dataset. In total, there are 31,542 cells in interphase, 605 cells in prophase, 68 cells in metaphase, 15 cells in anaphase, and 25 cells in telophase split into a training set (left) and a validation set (right). The training set is further split via 10-fold cross validation where each cross validated fold is roughly the same size and features the same distribution as the validation set.

validation data. In this way, we are able to obtain not only point estimates of the classifiers balanced accuracy score but also distributional information about the metric.

During training we employed random undersampling [94] to further address the class imbalance issue. In random undersampling, one selects the largest subset of the data such that every class is represented the same number of times. While the model is trained on these smaller, balanced datasets, the model is assessed on the larger, imbalanced dataset using Equation 4.1.

Data Augmentation

For the most sparsely populated class of the training data, anaphase, there are only 13 examples to be used in training. Hence, for the 5 classes considered, a random undersample used in training would consist of only 65 training samples. As such, we artificially increased the size of the dataset using data augmentation. This process of data augmentation is sometimes referred to as synthetic oversampling. However, in a traditional synthetic oversampling scheme, one synthetically augments the minority classes until all classes have the same representative size. Given how large the gap is between the majority class and the minority class in our training examples (the ratio of interphase to anaphase samples is approximately 2206) purely using synthetic oversampling would result in a training set where the minority classes are composed almost entirely of synthetic data. Not only would this drastically increase the training time of the methods, the resultant synthetic anaphase class would contain examples that are much more highly correlated with one another than the examples in the interphase class. Instead, we choose an augmentation factor ϕ and sample $\phi \cdot n_{\text{anaphase}}$ examples from each class. If the class itself is not populous enough for us to acquire this many samples, we simply sample all examples from that class and fill in the remaining samples with synthetic data.

Our augmentation techniques considered include the following 4 options chosen uniformly at random. First, we employ random rotations, where a random image is rotated at a random angle in $[0, 2\,\pi)$. Any missing pixels are filled in via nearest neighbours sampling. Given the nature of the data, these pixels that need filling represent the background of the image (and not the actual cell body) and hence the information contained is of little consequence. Secondly, we employ random reflections of the image, randomly selected as either vertical or horizontal reflections at random. Thirdly, we employ a random zoom and crop procedure, where the size of an image is first increased via bi-cubic interpolation from a $55 \times 55 \times 2$ image to a $69 \times 69 \times 2$ image and randomly cropped back to a $55 \times 55 \times 2$ image. These first 3 procedures were selected with the following biological justification: during the imaging flow cytometry process, the orientation of the cell is not guaranteed to be fixed in any way, hence the random rotations and random reflections produce equally valid views of the same cellular information. Moreover, by randomly zooming into the images we provide our algorithms with a different view of the same information. In all 3 examples, the pixel-level data is drastically different between the source image and the augmented image, but the cellular information contained is the same between the two images. Finally, for the fourth procedure, we synthetically generate new images using a generative adversarial network (GAN). This GAN process is accomplished via Contrastive-Unpaired-Translation (CUT) [81], a variant of CycleGAN [118]. CUT, CycleGAN, and other similar methods

operate by translating images of distribution A into images of distribution B. In the context of this work, we can take an image of a cell earlier in the cell cycle (prophase, for instance) and translate it into an example of an image from the next phase of the cell cycle (metaphase). This process results in an image that is unique from the others in the class, interpolating not only the class features from the elements of the target class but also those features from the origin class that are likely to show up in the latter distribution (B). An important aspect of these methods, compared to their progenitors, is that they do not require paired examples of each class (in this context, a paired example would be an image of a cell in prophase and an image of the *same* cell in metaphase). CUT is especially appropriate for this implementation as the convergence speed is substantially quicker than other GAN methods [81]. Since, in order to avoid spurious overfitting, the GAN needs to be trained independently on each fold of training data, the computational benefit of the faster convergence speed is especially crucial. Furthermore, CUT has been shown to be uniquely adapted to few-shot learning, including impressive results given only a single example of images from each class. In the context of these cellular images, on a single training fold there may be as few as 11 examples of a particular class, hence the capacity to perform few-shot learning is exceptionally important.

If $p(X) \in [0,1]^5$ represents the logit prediction of the input image X and X_{rot} represents a random rotation of the target image, X_{flip} a random flip of the target image, *et cetera*, then, for the purposes of prediction, we considered the following function

$$\tilde{p}(X) = \frac{\sum_{i=1}^{w_{\text{rot}}} p(X_{\text{rot}}) + \sum_{i=1}^{w_{\text{flip}}} p(X_{\text{flip}}) + \sum_{i=1}^{w_{\text{zoom}}} p(X_{\text{zoom}}) + \sum_{i=1}^{w_{\text{id}}} p(X)}{w_{\text{rot}} + w_{\text{flip}} + w_{\text{zoom}} + w_{\text{id}}}$$

where the weights w_{rot}, w_{flip}, w_{zoom}, and w_{id} were selected by choosing the integer values (not all 0) in $[0,5]^4$ such that the balanced accuracy score averaged over 3 random seeds of the CNN classifier was maximised over the held-out validation set. The best values coincided with the case such that $w_{\text{rot}} = w_{\text{zoom}} = 2$, $w_{\text{flip}} = w_{\text{id}} = 1$. Note that $\sum_{i=1}^{w_{\text{rot}}} p(X_{\text{rot}}) \neq w_{\text{rot}} p(X_{\text{rot}})$ as each rotation is sampled independently. This is similar for the other random augmentation methods.

4.2.3 Manual Feature Selection

The method of Blasi et al. [8] utilizes a manual feature extractor. This method works by importing the raw imaging data from the imaging flow cytometer into the software CellProfiler [14]. CellProfiler then outputs 128 features for the brightfield images and 85 features for the darkfield images. These features include things like granularity, Zernlike

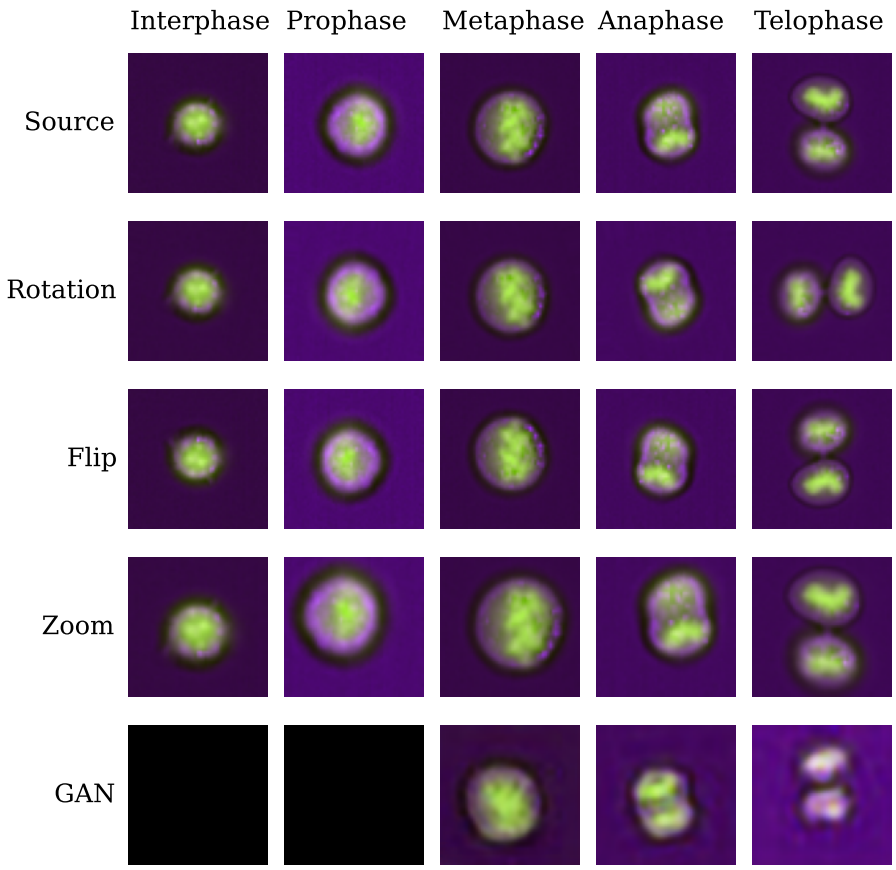


Figure 4.3: Examples of authentic data from the data augmentation process

polynomials, intensity, radial distribution, mean radius, perimeter, area, etc. (for a full list, see the supplementary information in [8]). These morphological features are then used as the input to the classifier algorithms. In effect, CellProfiler acts as a step to convert 2 channel cell images into feature vectors. Once these feature vectors have been selected, we can digitally sort the cells into classification bins using any number of classification algorithms. As in Blasi et al. [8] we a random undersampled boosted ensemble (RUSboost [94]), however Blasi et al. only utilised a hand tuned DTC as a weak learner and we consider various hyper-parameter tuned weak learners for comparison.

Decision Tree Classifiers (or DTCs) are a popular method to use in conjunction with ensemble algorithms for solving classification algorithms. Indeed, these random forests are incredibly powerful tools for solving a variety of classification tasks due to their capacity to produce classifiers with flat loss functions around the optimum (and hence tend to generalize well) [11]. Moreover, DTCs are quite computationally inexpensive to fit, hence in ensemble boosting contexts a large number of classifiers can be combined to produce an ensemble method.

k-Nearest Neighbours classifiers (or KNNs) are another simplistic weak learner to consider. We consider using KNNs for a variety of reasons. Firstly, as a point of comparison to the previously employed DTCs. Secondly, because the KNN algorithm is a sufficiently naïve classifier it provides a helpful baseline of a minimal score of the metric of interest. For instance, for a 2-class classification problem an accuracy score of 0 technically encodes nearly as much information as an accuracy score of 1. Similarly, because the truly most naïve classifier is a random classifier, and a random classifier would expect a $1/n$ on average for an n-class classification problem, considering $1/n$ as a baseline for balanced accuracy score is more useful than 0. As a result, the score obtained by a random classifier represents the "true bottom" of a metric. The KNN algorithm is similarly utilised to provide context towards a "reasonable bottom" of a metric. While random selection is the most naïve classifier, the KNN classifier represents an incredibly naïve classifier that actually uses the feature data for classification. As a result, the KNN score provides a more helpful baseline minimal score.

Finally, we also consider using an artificial neural network (ANN) as a classifier for these feature vectors. ANNs have achieved state of the art scores on various image classification tasks. Moreover, evaluating an ANN on the manually feature selected data allows for a very natural comparison with the automatic feature selection methods. The architecture considered is in part informed by the hyperparameter optimisation performed in Section 4.2.5 but broadly consists of a number of fully connected layers, all activated by ReLU functions, ending in a layer of 5 neurons activated by the softmax function. For a loss function we considered the categorical cross-entropy function. As such, the output of the ANN is a logit

vector where the ith entry corresponds to the probability of the sample inhabiting the ith class.

4.2.4 Automatic Feature Selection

For automatic feature selection these algorithms generally consist of two portions. The first portion represents a feature extractor and the latter portion is responsible for the classification of the output. In this sense, there is a similarity in structure to the method described in Section 4.2.3. However, whereas the feature extractor role of CellProfiler can, in principle, be tuned to select different features, such a process is done manually before training. Whereas by leveraging automatic feature selection algorithms, local spatial information about the image gained by convolution kernels is directly a part of the optimisation problem. Hence, during training the machine learning algorithm can tune the weights to focus the convolution kernels on different aspects of the images in question. As a result, the particular feature vector being used by the classification portion of the algorithm is changing constantly in order to help minimize the loss of the algorithm. While there is not a one-to-one analogue with the manual feature selectors, in practice the methods perform similar roles.

One can train a feature extractor from scratch to perform these tasks, but we employed a popular pre-trained CNN based architecture called ResNet [45]. ResNet is pre-trained on the ImageNet dataset – a dataset consisting of over 14 million images of everyday objects. While ImageNet does not contain any cell images, we are not interested in the classification portion of the network merely the feature selection portion. This feature selection portion is quite similar across many classification problems as the fundamental features needed to be evaluated are similar. That is to say, ResNet has been trained to produce features by recognizing fundamental shape and edge information of objects that is similar in any image classification task. Hence, using a pre-trained ResNet as a starting point will reduce the training time and computational cost of the classification problem. The final structure of our CNN based architecture consists of the pre-trained ResNet base to perform feature selection followed by a number of ReLU activated fully connected layers (informed by the results of Section 4.2.5) and finally, a softmax activated output layer of 5 neurons for each class considered.

4.2.5 Hyperparameter Optimisation

There are numerous hyperparameters to these methods that we need to consider. Not only are there hyperparameters for each of the individual weak learners θ_{wl}, there are also hyperparameters for the ensemble boosting algorithm θ_{ens}. The performance of these methods depends on the values of these hyperparameters - in some cases this dependence is quite crucial. While these hyperparameters can be hand-picked, often better performance is observed when the hyperparameter values are determined via the data in some manner. However, if the hyperparameters were tuned by choosing the values that best improved the metric of choice on the whole dataset, then one could find themselves with spuriously high balanced accuracy scores due to overfitting. Hence, we tune the hyperparameters by only evaluating the methods on the held out validation dataset.

The computational cost of performing hyperparameter optimisation in an end-to-end manner (i.e. by training an entire ensemble on all 10 folds of the cross-validated dataset) is computationally quite expensive, so instead we consider tuning the hyperparameters in a multi-stage process.

1. First we pretune the weak learner by determining θ_{wl} by training on a single learner on random undersamples of the training data. We then evaluate this learner by calculating the balanced accuracy score on the validation data.

2. Then we take the tuned weak learner (tuned according to the values of θ_{wl} discovered previously) and determine θ_{ens} for the ensemble.

The particular hyperparameters that need to be tuned for each weak leaner, θ_{wl}, depend on the particulars of each weak learner being investigated. However $\theta_{\text{ens}} = \langle \alpha_{\text{e}}, n_{\text{wl}} \rangle$ for each weak learner. That is, while the values of α_{e} and n_{wl} will change for each weak learner, there are not any additional ensemble hyperparameters to consider for any of the weak learners. Note the relationship between n_{wl} and α_{e} is such that the two vary as the inverse of one another. As a result, we choose α_{e} manually and train an ensemble for a large number of learners n_{wl}. Then, we calculate the cumulative validation loss of the ensemble as a function of n_{wl} and minimize this function on the ith fold of the data by setting ${n^*_{\text{wl}}}^{(i)}$ as the argmin of the validation loss. This training process is repeated for multiple splits over the training data and the final value of n^*_{wl} is taken as the nearest integer to the mean of the ${n^*_{\text{wl}}}^{(i)}$s.

We first present the results of pre-tuning the weak learners. For the DTC we considered the hyperparameters $\theta_{\text{wl}} = \langle n_{ss}, n_{sl} \rangle$ where n_{ss} is the minimum number of samples required

at a parent node and n_{sl} is the minimum number of samples required at a leaf. We optimised the values of θ_{wl} by brute forcing over the lattice $n_{ss} \in \{2, 3, \cdots, 40\}$ and $n_{sl} \in \{1, 2, \cdots, 20\}$. This hyperparameter space was selected based on a meta-analysis of DTC methods that determined the likely hyperparameters for a classification task fall in this region [66]. For each of the 780 possible hyperparameter combinations, we trained DTC classifiers on 200 different random undersamples of the training data and evaluated these classifiers on the validation data. The results of which are presented in Figure 4.4. Note that for a large portion of the hyperparameter space, roughly bounded by $2 \leq n_{ss} \leq 32$, $1 \leq n_{sl} \leq 11$ the classifier performed similarly well. In effect, this suggests that while there may not be one *perfect* choice of hyperparmeters, there are certainly sub-obtimal choices of hyperparameters by taking values outside of this region.

For the KNN classifier, we considered $\theta_{\text{wl}} = \langle k, p, w \rangle$ where k is the number of neighbours to consider, p is used for calculating the distance between features x and y in $\mathbb{R}^d$ via the l^p norm $\|x - y\|_p = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$, and w is the weighting scheme for determining contributions from neighbours in prediction. We considered $k \in \{1, 2, \cdots, 30\}$, $p \in \{1, 2\}$, and w to correspond neighbouring contributions being either uniform or distance weighted. (For uniform-weighted contributions, the effects of all k neighbours are considered evenly; in contrast, for distance-weighted contributions, the closer of the k neighbours are weighted heavier than those of the k neighbours that are further away). We brute forced over the entire 120 hyperparameter space by training classifiers on 200 different random undersamples of the training data evaluated on the validation data. The results are presented in Figure 4.5. Note the sharp decrease in the balanced accuracy score for the uniform weighted curves suggesting that such methods may not generalize as well in situations where the nearest neighbour is far away from training examples. However, in the two distance weighted curves, the balanced accuracy score plateaus briefly before decreasing, allowing for the algorithm to consider the input of multiple sources. In particular, the optimum choice observed on the validation data corresponds to $k = 2$ with $p = 2$ and distance weighting.

For the neural network approaches, brute forcing the hyperparameter space would be computationally infeasible given the size of the hyperparameter space to consider combined with the computational cost of fitting these methods. As such, we consider a Bayesian optimizer with an upper confidence bound acquisition function. For the ANN we considered $\theta_{\text{wl}} = \langle \alpha, n_{\text{hd}}, d_1, \cdots, d_{n_{\text{hd}}} \rangle$ where α refers to the learning rate of the neural network, n_{hd} represents the number of hidden layers (the depth of the network), and the d_is represent the dimension of these hidden layers (the breadth of the network). Ostensibly, there are other hyperparameters to consider. For instance, the number of epochs trained, the batch size, and the optimizer are all hyperparameters to consider. In the case of our method,

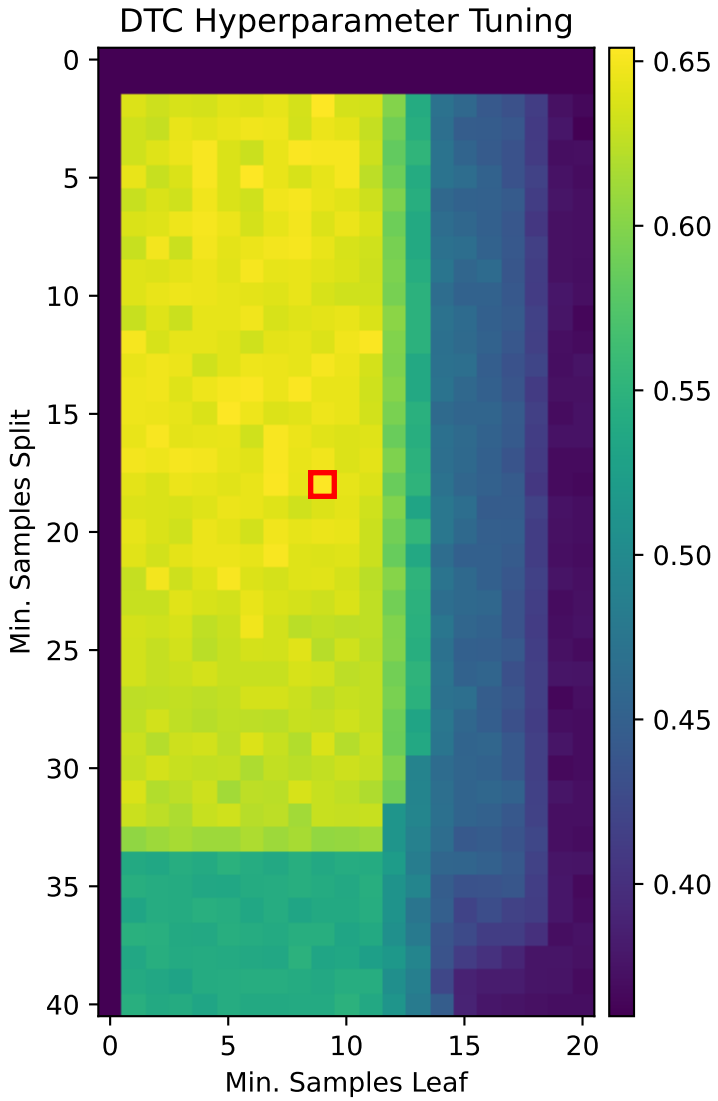


Figure 4.4: A contour plot presenting the results of hyperparameter tuning for a single decision tree classifier. The red box indicates the hyperparameter set that gives the largest predicted balanced accuracy score on the validation data. Note the large swath of hyperparameters for which the metric score achieved is similar.

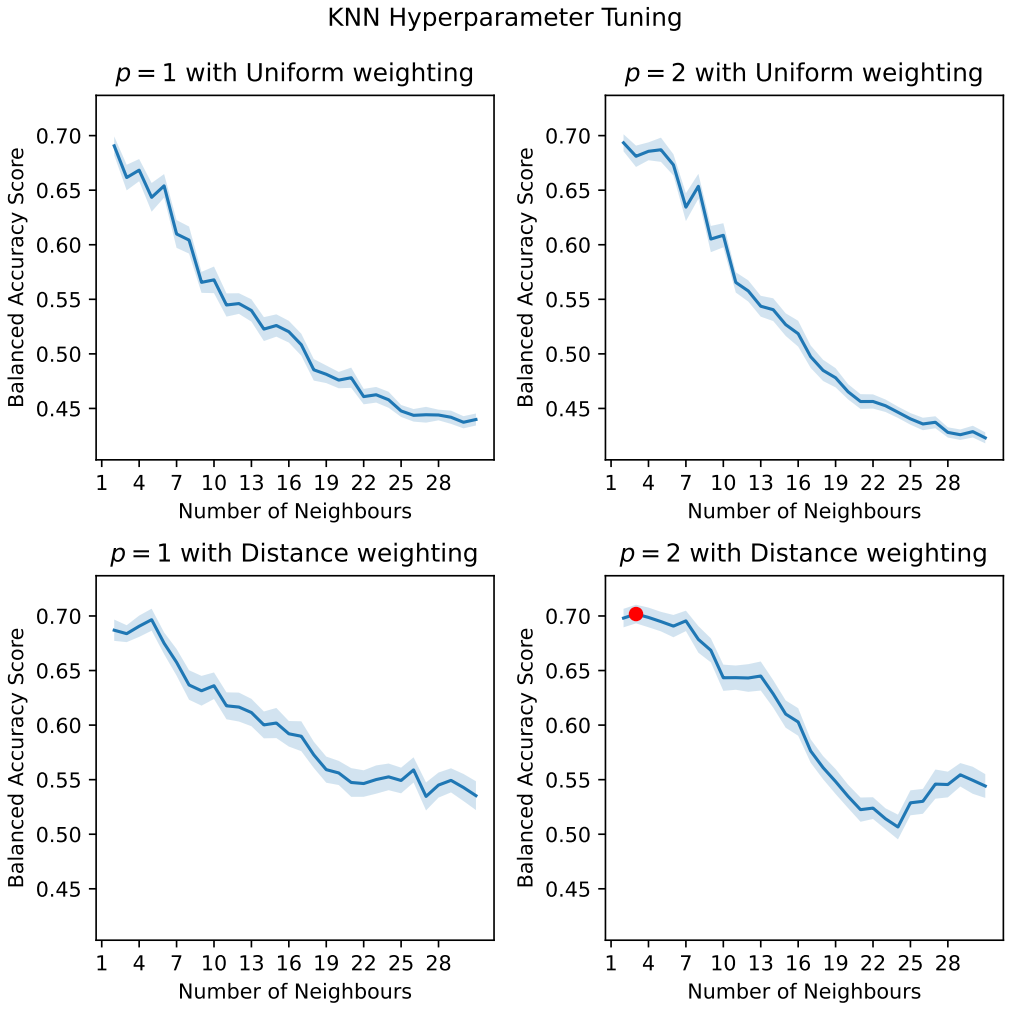


Figure 4.5: Results of hyperparameter tuning for the k Nearest Neighbours classifier. The orange circle indicates the hyperparameter set that gives the largest predicted balanced accuracy score on the validation data. Shaded regions represent 95% confidence intervals.

we fixed the batch size of the network to 32, the number of epochs to 100, and used the Adam optimizer. To justify the choice of fixing the number of epochs, we note that there is an inverse relation between the number of epochs and the learning rate [36]. For a smaller learning rate, more epochs are needed to sufficiently train the network. We took $10^{-6} \leq \alpha \leq 10^{-1}$ and sampled α in a log-sampling manner such that each order of magnitude was equally likely. We considered $n_{\text{hd}} \in \{0, 1, 2\}$ with $d_i \in \{32, 64, \ldots, 512\}$. Note that the choice of n_{hd} determines the architecture of the network and so determines how many of the d_i also need to be chosen. In particular, the choice of $n_{\text{hd}} = 0$ results in a shallow perceptron with no hidden layers (and hence, no hidden dimensions to determine). All hidden layers are activated by the ReLU function and the final layer is activated by the softmax function. We first seeded the Bayesian optimizer by selecting $3 \cdot (n_{\text{hd}} + 1)$ hyperparameters at random. Then, we allowed the Bayesian optimizer to select the next hyperparameter choice in the search space to evaluate. We allowed this process to continue for 100 such trials. For each hyperparameter the data was fit on the 10-fold cross validated training data and evaluated on the validation data. The results of this optimisation are visualised in Figure 4.6. In particular, note that there is an increase in balanced accuracy score between the $n_{\text{hd}} = 0$ and the $n_{\text{hd}} = 1$ case, but no such increase is observed in the $n_{\text{hd}} = 2$ case.

In [98] the authors presented a method called highway networks that uses gated residual connections to increase the number of layers without having performance saturate. To that end, we also considered hyperparameter optimizing a highway network with three hidden layers with dropout on the hidden layers. Here the dropout rate of these hidden layers was considered as an additional hyperparameter in the interval $[0, 1)$ to optimize and was kept as the same rate between all three hidden layers. This approach did not increase the accuracy of the classifier, suggesting that the accuracy of the classifier has saturated due to a fundamental inability to learn more from the data rather than from difficulties in training deep networks.

For the CNN approach we considered $\theta_{\text{wl}} = \langle \alpha, d, \phi \rangle$ where $10^{-6} \leq \alpha \leq 10^{-1}$ is the learning rate of the neural network (log-sampled as in the ANN case), $d \in \{32, 64, \cdots 512\}$ is the dimension of the single hidden layer, and $\phi \in \{2, 3, \ldots, 32\}$ is the augmentation factor. As in the case of the ANN we consider an Adam optimizer with a batch size of 32 where training is performed over 100 epochs. Similarly, the hidden layer is activated by the ReLU function and the final layer is activated by the softmax function. Note that only 1 layer is considered for the CNN as informed by the results of the ANN hyperparameter optimisation. We seeded the Bayesian optimizer by first evaluating 8 separate hyperparameters at random before allowing the optimizer to select the next hyperparameter choice in the search space to evaluate for 100 total trials. Each trial was similarly fit on the 10-fold cross validated

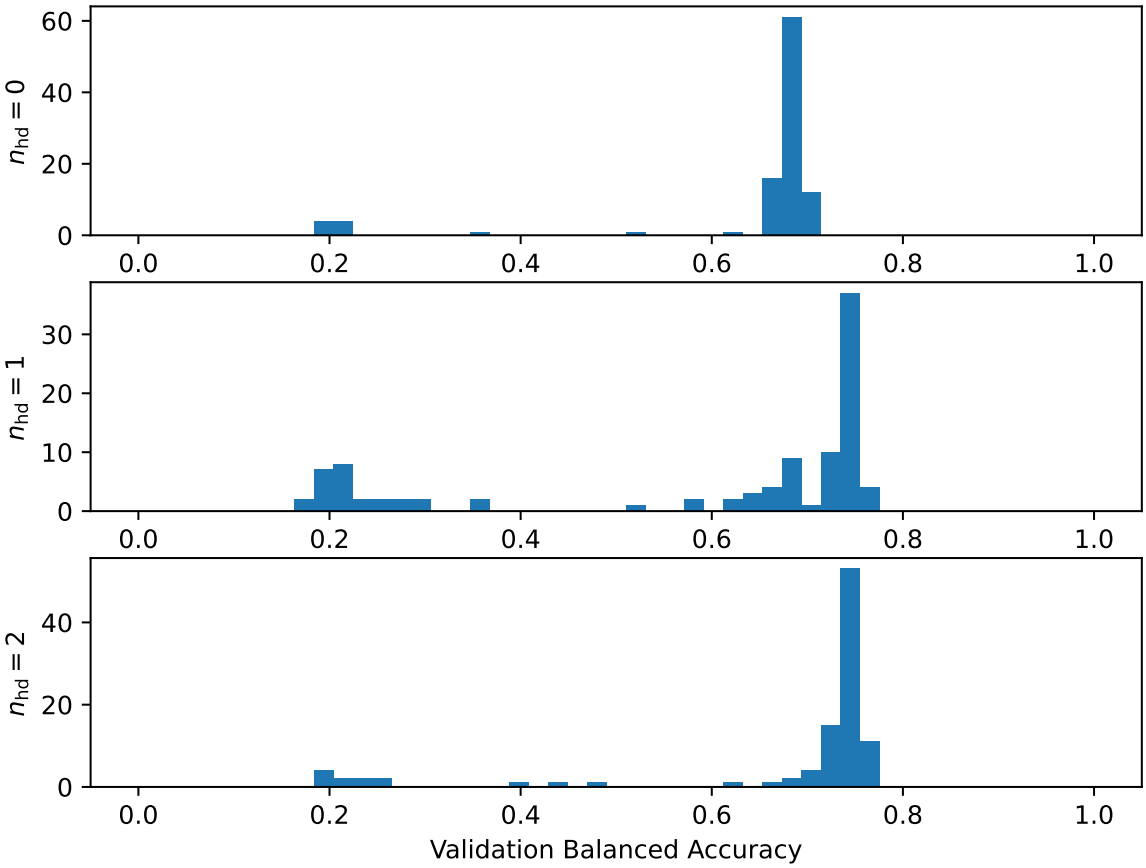


Figure 4.6: Histograms of scores from the ANN hyperparameter optimisation process stratified by n_{hd}. Note that the maximum scores in the $n_{\text{hd}} = 1$ case match the maximum scores obtained in the $n_{\text{hd}} = 2$ case.

training data and evaluated on the validation data.

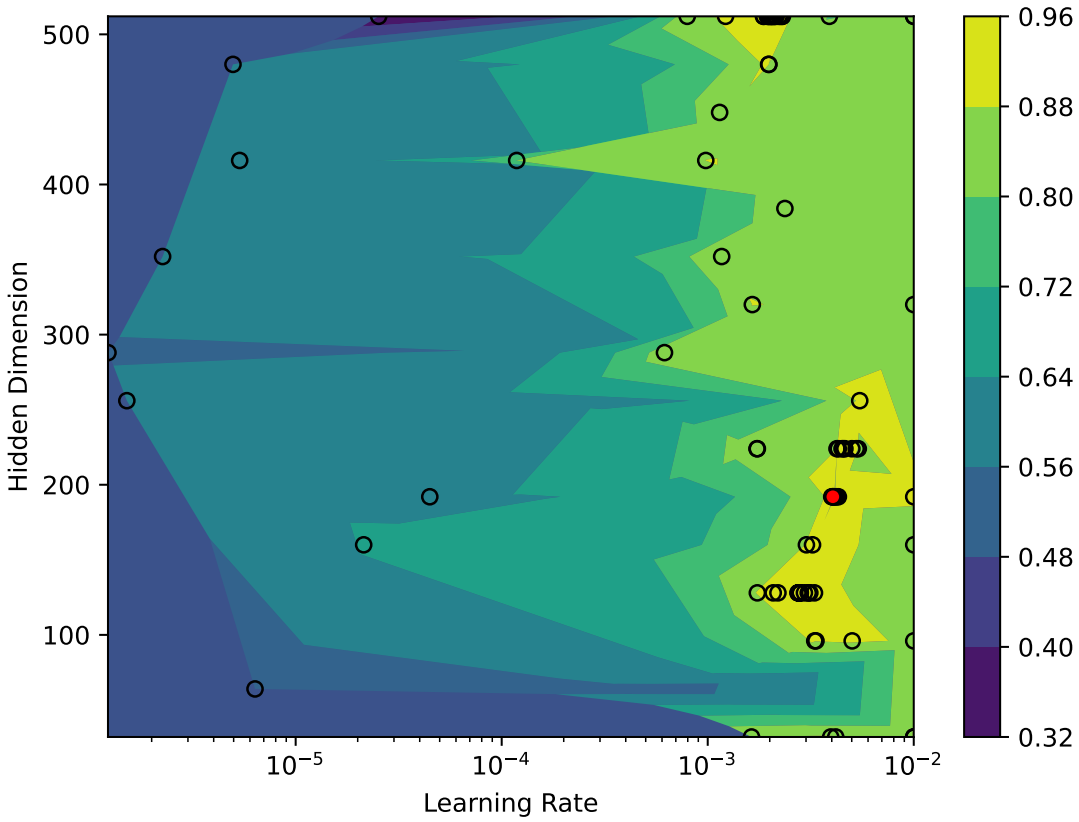


Figure 4.7: A contour plot of validation balanced accuracy scores from the CNN hyperparameter optimisation process. Circles indicate hyperparameters chosen by the Bayesian optimizer. The red circle indicates the hyperparameter set with the highest validation balanced accuracy score.

The result of the hyperparameter optimisation for each method is summarised in Table 4.1

Finally, for each classifier we performed ensemble boosting. In the case of the decision tree, k nearest neighbours, and artificial neural network classifier we further tuned the ensemble specific parameters. An example of the loss function for the KNN classifier is presented in Figure 4.8. The plots look similar for the other classifiers. Note that while the ensemble originally contained 3000 weak learners, the loss function is minimised by

Method	Model HPs	Ensemble HPs
DTC	$n_{ss} = 18$, $n_{sl} = 9$	$\alpha_e = 0.1$, $n_{\text{clf}} = 4706$
KNN	$p = 2$, $k = 2$, Distance Weighted	$\alpha_e = 0.1$, $n_{\text{clf}} = 911$
ANN	$\alpha = 0.0056578$, $n_{\text{hd}} = 1$, $d_1 = 320$	$\alpha_e = 0.1$, $n_{\text{clf}} = 113$
CNN	$\alpha = 0.0040482$, $d = 192$, $\phi = 32$	$\alpha_e = 0.1$, $n_{\text{clf}} = 25$

Table 4.1: Table of the hyperparameters used for each classifier

instead selecting 911 weak learners. These results are included in Table 4.1. For the CNN classifier, we instead ensemble boosted 25 classifiers. The loss function remained flat after 25 classifiers and the training time was exceptionally long such that the expected benefit from continuing to tune the number of classifiers in the ensemble is minimal compared to the computational cost required to construct such a large ensemble. This quick saturation of accuracy as a function of n_{clf} has been observed elsewhere (for instance Figure 5 of [111]).

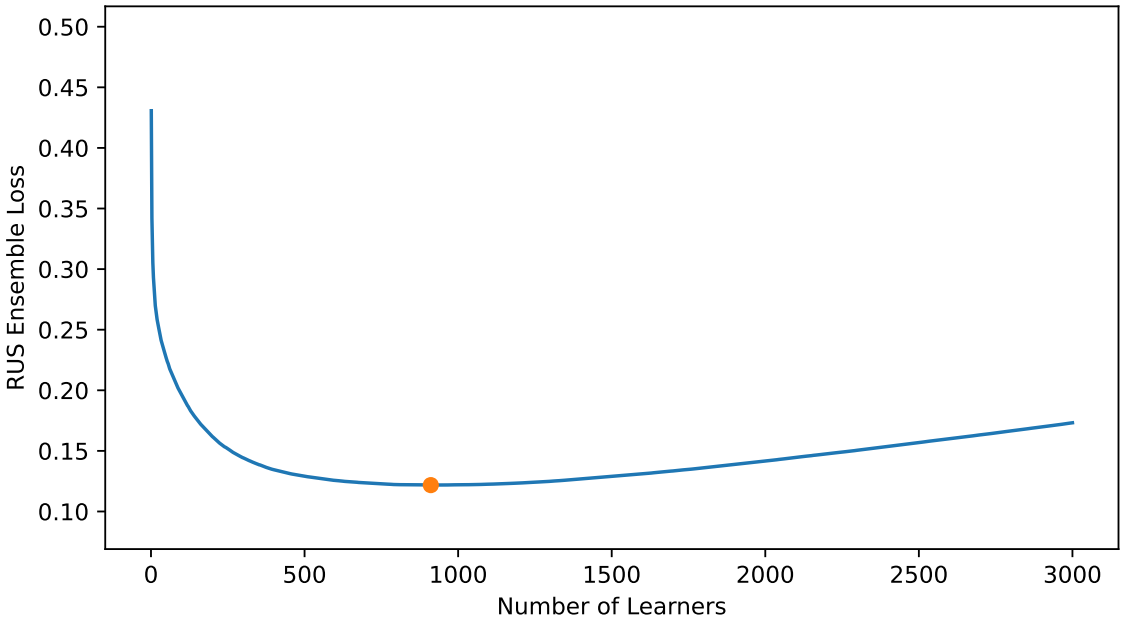


Figure 4.8: Loss curve for the KNN ensemble. Note the characteristic non-monotone shape of the loss function. The orange dot indicates the minimum and hence the optimal number of classifiers to use in the ensemble to maximize accuracy. Moreover, the relatively flat area around the optimum is indicative of a strong generalisation capacity.

4.3 Results

4.3.1 Manual Feature Selection

The results obtained by the various methods are reported in Table 4.2. To generate these results we first compared the output of a single classifier with hyperparameters obtained in 3 ways: their default values, random values from the hyperparameter space, and finally the tuned values. We then compared the ensembled classifiers under the same environment. In effect, this demonstrates the benefit of the (often costly) process of ensemble boosting and hyperparameter tuning on the models.

Classifier	Default HP Score	Random HP Score	Tuned HP Score
KNN	$61.83\% \pm 0.22\%$	$49.45\% \pm 0.16\%$	$\mathbf{66.59\% \pm 0.15\%}$
DTC	$\mathbf{70.00\% \pm 0.19\%}$	$42.83\% \pm 0.17\%$	$67.08\% \pm 0.21\%$
ANN	$63.47\% \pm 0.34\%$	$\mathbf{69.88\% \pm 0.38\%}$	$69.29\% \pm 0.37\%$
Ens. KNN	$74.35\% \pm 0.27\%$	$72.66\% \pm 0.28\%$	$\mathbf{75.19\% \pm 0.15\%}$
Ens. DTC	$\mathbf{76.76\% \pm 0.19\%}$	$74.38\% \pm 0.20\%$	$75.02\% \pm 0.21\%$
Ens. ANN	$73.92\% \pm 0.45\%$	$79.51\% \pm 0.44\%$	$\mathbf{80.30\% \pm 0.40\%}$

Table 4.2: Results for the manual feature selected methods. Error is calculated as the width of the 95% intervals of the value of the mean balanced accuracy score achieved by each classifier. Bold values represent the best version of each classifier.

Note for the ensembled classifiers, pre-tuning the classifier provides a benefit in all but the decision tree classifier case. Given the results of Figure 4.4, it is not surprising that the decision tree classifier performs similarly irrespective of hyperparmeter tuning.

4.3.2 Automatic Feature Selection

For the CNN based classifier the results are presented in Table 4.3. In comparing the results of the classifier with those from the feature engineered data, we note that a single CNN classifier outperforms all the feature engineered examples. Ensemble boosting the CNN classifier increases this advantage even further.

For the ensembled CNN classifier we note that this increased performance comes at a computing cost. Ensemble boosting adds an n_{clf} computing expense not only during training but also during inference. Given the already impressive performance of the CNN classifier, this additional performance benefit is only marginal and incurs a great computational cost.

Classifier	Balanced Accuracy Score
CNN	$87.44\% \pm 2.41\%$
Ens. CNN	$90.08\% \pm 0.97\%$

Table 4.3: Results for the automatic feature selection methods, error is given by the width of the 95% confidence intervals of the value of the mean balanced accuracy score (obtained by bootstrapping the cross validated balanced accuracy scores).

However, beyond just the increase in accuracy, the ensembled classifier tends to provide a more robust classifier. Indeed, the standard deviation of the balanced accuracy scores obtained on each fold of the 10-fold cross validation for the ensembled CNN is 0.0309 as compared to the 0.0767 obtained by the single CNN classifier. For comparison, the ensembled ANN and ensembled DTC classifiers have standard deviations of 0.0350 and 0.0333, respectively.

Finally, comparing balanced accuracy scores is only one such way to evaluate these classifiers. We also present a selection of the confusion matrices for these classifiers.

$$\chi_{\mathrm{D}}^{\mathrm{ens}} = \begin{bmatrix} 0.93 & 0.04 & 0.02 & 0 & 0 \\ 0.29 & 0.48 & 0.21 & 0.01 & 0 \\ 0.21 & 0.18 & 0.40 & 0.21 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 1.00 \end{bmatrix} \quad \chi_{\mathrm{A}}^{\mathrm{ens}} = \begin{bmatrix} 0.84 & 0.08 & 0.06 & 0 & 0.01 \\ 0.12 & 0.53 & 0.33 & 0.01 & 0.01 \\ 0.03 & 0.19 & 0.66 & 0.11 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0 & 1.00 \end{bmatrix}$$

$$\chi_{\mathrm{C}} = \begin{bmatrix} 0.91 & 0.08 & 0.01 & 0 & 0 \\ 0.14 & 0.69 & 0.16 & 0.01 & 0 \\ 0.02 & 0.06 & 0.87 & 0.05 & 0 \\ 0 & 0.08 & 0 & 0.92 & 0 \\ 0 & 0 & 0 & 0.05 & 0.95 \end{bmatrix} \quad \chi_{\mathrm{C}}^{\mathrm{ens}} = \begin{bmatrix} 0.92 & 0.08 & 0 & 0 & 0 \\ 0.11 & 0.77 & 0.11 & 0.01 & 0 \\ 0 & 0.06 & 0.87 & 0.06 & 0 \\ 0 & 0 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & 0.05 & 0.95 \end{bmatrix}$$

where $\chi_{\mathrm{D}}^{\mathrm{ens}}$ represents the confusion matrix of the ensembled decision tree classifier, $\chi_{\mathrm{A}}^{\mathrm{ens}}$ the confusion matrix of the ensembled ANN classifier, χ_{C} the confusion matrix of the single CNN classifier, and $\chi_{\mathrm{C}}^{\mathrm{ens}}$ the confusion matrix of the ensembled CNN classifier. The rows and columns of each confusion matrix correspond to interphase, prophase, metaphase, anaphase, and telophase cells, respectively. In effect, the first two matrices compare the feature engineered result using a tuned version of the model from [8] with the best feature engineered result seen in this study. Similarly, the final two matrices compare the results of a single classifier without feature engineering with the best ensembled result without feature engineering.

We note that, when compared with the DTC ensemble, the ANN ensemble sacrifices

accuracy in the interphase compartment for increased accuracy in both the prophase and metaphase compartments. Comparing the feature selected methods, we see that the ensembled CNN classifier outperforms (or matches) the single CNN classifier in true positive accuracy for all cell phases. Moreover, the ensembled CNN classifier outperforms all the classifiers observed for true positive accuracy in all but one case: the ensembled DTC classifier achieves a 1% better true positive accuracy for the interphase compartment. Moreover, the ensembled DTC classifier achieves perfect accuracy in the minority compartment (anaphase) and strong accuracy of 95% and 87% in the next two smallest compartments (telophase and metaphase) and produces a more diagonally dominant confusion matrix than the other methods considered.

4.4 Summary

By utilizing the power of convolutional neural networks a classifier is built that can correctly classify the mitotic phase of Jurkat cells with a balanced accuracy score of over 90% as compared to the score achieved by a previous, feature engineering approach of roughly 77%. By comparing the CNN ensemble with an ANN ensemble on the feature engineered data we demonstrate that this increase in score is due to more than just the additional complexity of neural network classifiers, but is directly linked to the capacity of allowing features to be learned for cellular identification. The trustworthiness of this method is validated due to the strong agreement between the predictions and the ground-truth data known in this supervised learning task. The method was further validated by how tight the distribution of balanced accuracy scores was when calculated over multiple different training folds (as presented in the error bounds in Table 4.2 and Table 4.3.

Chapter 5

Reinforcement learning derived chemotherapeutic schedules for robust patient-specific therapy

In this chapter we develop a method for producing schedules of chemotherapy delivery in a model-free manner and contrast this with classical control methods. We demonstrate that the proposed reinforcement learning method is robust to perturbations in individual patient response. In effect, this allows for an adaptive, patient specific chemotherapy schedule to be developed that behaves strongly even when evaluated on patients that differ substantially from those patients to whom the model was originally calibrated. This method uses an underlying ordinary differential equation model to drive the *in silico* experiments, but the chemotherapy scheduling method is in fact model free. This means that such a scheduling approach could be applied to patients in clinic without the need to first calibrate an underlying mathematical model to the individual patient. Dr. Michelle Pzedborski assisted in the design of the study. The work presented in this chapter was published in Scientific Reports [23].

5.1 Introduction

In mathematical models of cancer treatment, one is often concerned with finding a chemotherapeutic dosing schedule that is optimal in some capacity [80, 113, 50]. Each different model allows distinct forms of this optimality metric. This shaping of optimality metric will often

involve a trade-off of some variety: incredibly high doses of a potent chemotherapeutic can certainly annihilate the cancerous cells in tissue but in so doing will largely cause a great deal of harm to the patient. Modelers, then, are concerned with mathematically formulating this optimality metric in a way that preserves the health and longevity of their patient (virtual or otherwise). For instance, in [113] the authors were concerned with maximizing the reduction in the total number of cancerous cells with the minimal total chemotherapeutic dose. They achieved this control by sampling 200 virtual patients from a particular parameter distribution, training 50 different reinforcement learning agents on differential equations representing the tumour growth of these patients, and applying these agents to these patients. In contrast, in [80] the authors concerned themselves with maximizing the chemotherapeutic dose while minimizing the damage to healthy, proxy cells in the bone marrow. Practically, these are two (similar and related) methods for achieving the same ends, but the particulars of their formalisation can lead to drastically different qualitative results.

In any situation, the models used to represent the delivery of the chemotherapeutic and the associated reduction in cancer cells can inform the choice of optimality metric. So too can the choice of model inform the method by which a modeler can find such an optimal dosing schedule of a chemotherapeutic. One such method, as employed in [80], is that of optimal control theory. When the model that governs the behaviour we are trying to assert some control over is codified by differential equations, optimal control theory can provide a methodology for finding the dose delivery function that maximizes whatever optimality metric the modeler chooses (if such a metric has a maximum) [86]. In some situations, this optimal control can be accomplished analytically as in [80], in others (such as the objective functional presented in (5.3)) numerical techniques such as those employed by the GEKKO package may be required [3].

A reinforcement learning approach can also be employed to maximize a given optimality metric (see, for instance, [113]). In a reinforcement learning context, when the state of the model at a given time t is known, one can construct a controller function via the learned optimal policy. In contrast to optimal control, reinforcement learning more easily lends itself to situations where the model behaviour is governed by systems more complicated than just those that can be represented with differential equations (for instance, reinforcement learning has had great success in solving Atari games, arcade games, Backgammon, etc. [102, 101, 4]). In particular, as illustrated in Figure 5.1, a reinforcement learner need only be provided with the action space of the environment; all other details about the environment are effectively a black-box. The agent takes an action, the environment then changes as a result according to some rule-set the learner need not have access to, and a reward is issued. The reinforcement learner then evolves to maximize total cumulative reward, not

just the immediate reward benefit. Importantly, the environment may be governed by a deterministic set of differential equations, by a stochastic agent-based model, or by rules entirely determined by data. In this capacity, the black-box nature of the reinforcement learner environment is enticing to applied mathematicians as it allows the capacity to perform numerical learning experiments in a regime that was previously untractable. Indeed, recent advancements in computing power have allowed for the tractability of model-free reinforcement learning [77]. With the advent of big data sets and quantitative medicine, reinforcement learning can be used to leverage real world data as well as deterministic, validated models in order to learn a control in complicated contexts. Presently, we consider the environment to be governed by a system of simple differential equations to establish a framework methodology that can, in the future, be extended to other, more realistic domains.

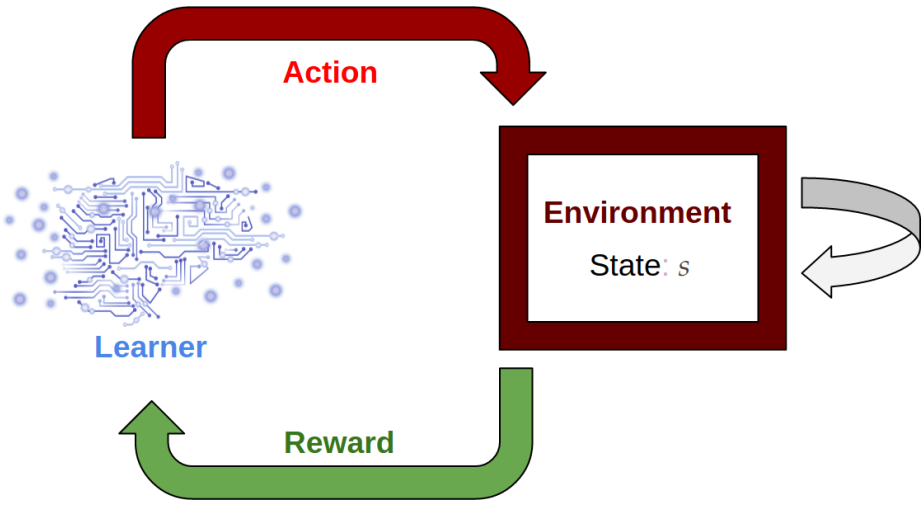


Figure 5.1: A reinforcement learning agent interacts with an environment as if the environment were a black-box. This process potentially changes the state of the environment and results in some reward for the learner. All that the learner needs to be provided with is the action space and a suitable reward function to determine an optimality metric.

In this study, we consider a simple differential equation model [80, 79, 78, 26]. This phenomenological model describes the growth of breast and ovarian solid tumours at a cellular level within a particular patient. The parameters of the model describe rates of cell-to-cell interaction and are incredibly difficult to measure in practice. In particular, the methods used in [78, 80] to parameterize this model only allow the discovery of nominal, mean values of such parameters from multiple mouse models. While these parameters can help to capture the qualitative behaviour of the response of a tumour to a particular

chemotherapeutic, the model can certainly not be considered to be a validated model in human cancers. However, even for a validated phenomenological model the issue of patient-specific parameter identification still remains. Whenever the parameter values used for these models are determined by population-level data the modeler may not know *a-priori* the particular patient-specific parameters. In contexts where there is a demonstrable sensitivity to small perturbations in the parameter values, there is a concern that the nominal parameters (and any chemotherapeutic control thereby derived) may not robustly describe the most optimal response for a particular parameterisation. To that end, in this paper we explore how chemotherapeutic controls derived from mean value parameters can be used on models of patients with perturbed, unknown parameter values. In particular, we leverage the power of deep double Q learning [102] to derive the chemotherapeutic control in a manner that provides learned dosing schedules that are robust to perturbations in parameter values in this sensitive system. Importantly, the reinforcement learning agent is unaware during training of the patient-specific parameter values on which it is evaluated. This is in contrast to [113] where multiple agents were trained on systems encoded by these parameter values exactly. In [27] a continuous control problem is considered for both single and combination therapy of chemotherapeutics where the dynamics are described by an Ito stochastic differential equation (SDE). Importantly, the author employs a reinforcement learning method (the deep deterministic policy gradient method [62]) and notes that the corresponding control appears robust to the stochasticity inherent in the SDE. In [80] the authors analytically derive the continuous optimal control of the tumour growth model used in this work under a particular objective functional. Here we consider a similar optimal control problem but wherein both the drug dose and time are discretised.

The chapter is organised as follows. In Section 5.2 we first introduce the differential equation model and provide the mean parameter values that comprise the nominal virtual patient. We then define the optimal control problem considered and the objective functional by which the optimal scores are deduced. We then describe the method by which virtual patients were created for testing and training purposes. Next, we lay out the training process used for solving the reinforcement learning problem and the discrete optimal control problem. Finally, we discuss the hyperparameter tuning process of the deep double Q learning algorithm.

In Section 5.3, we first derive an analytic characterisation of the initial conditions used in the model simulations as a function of these parameter values. Then, we perform local sensitivity analysis demonstrating that the model we consider is quite sensitive to local perturbations of the parameter values. We next present the results of the control agents by first allowing the agents to learn offline on an environment parameterised by the nominal patient. We then apply these agents to environments parameterised by the perturbed testing

patients. We measure the optimality of these schedules by logging the value of the objective functional achieved divided by the theoretical maximal value of this objective functional for each testing patient. We consider the optimality of schedules proposed for these perturbed testing patients for both the reinforcement learning agent and the traditional nominal optimal controller. In the former case, the relative bone marrow of these unknown patients was leveraged for the purpose of customizing the dosing schedule and reducing drug toxicity whereas, in the latter case, the same, nominal schedule is applied to all testing patients. We then extend the optimal schedules to leverage relative bone marrow measurements as well by employing a version of nearest neighbour interpolation on the optimal control of various training patients (whose particular patient specific parameter values are treated as known) to personalize dose schedules for testing patients. This nearest testing neighbour optimal control is then compared with the previous reinforcement learning agent. Finally, we present commentary and a summary of the work in Section 5.4.

5.2 Methods

5.2.1 Tumour Growth Inhibition Model

We consider the two-compartment mathematical model of cell-cycle specific chemotherapy first introduced in [78], which is an extension of earlier work [79]. The model consists of a population of proliferating cells and a population of quiescent cells, where the time evolution of cell populations is depicted in Figure 5.2 and is governed by the following set of coupled ordinary differential equations:

$$\begin{aligned} P'(t) &= \big(\gamma - \delta - \alpha - s\, f(t)\big)\, P(t) + \beta\, Q(t) \\ Q'(t) &= \alpha\, P(t) - (\beta + \lambda)\, Q(t) \end{aligned} \tag{5.1}$$

In the model, $P(t)$ represents proliferative cells and $Q(t)$ represents quiescent cells. The model captures the growth of proliferative cells at a constant rate γ, the transformation of proliferative cells into quiescent cells at a constant rate α, and the apoptosis of proliferative cells at a constant rate δ. Similarly, quiescent cells leave quiescence and become proliferative at a constant rate β and undergo apoptosis at a constant rate λ. The time-dependent function $f(t)$ represents the dosing schedule of a chemotherapeutic where s represents the relative strength of the administration of such a chemotherapeutic. In particular, it is assumed that $f(t) \in [0, 1]$. While parameters γ, δ, α, β, and λ are patient-specific parameters

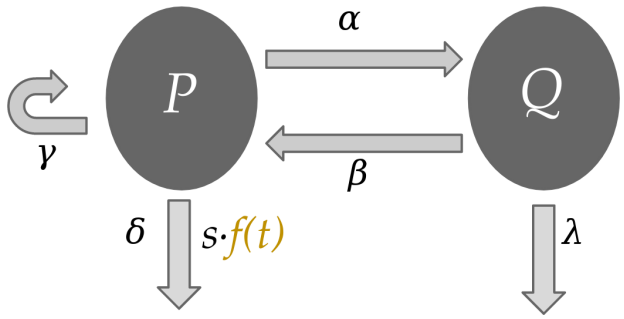


Figure 5.2: The two-compartment tumour growth inhibition model described by (5.1). Proliferative (P) cells and quiescent (Q) cells can both die naturally at the constant rates δ and λ, respectively. However, only proliferative cells can self-renew (at the constant rate γ) and be killed by the dose of a chemotherapeutic $f(t)$. Moreover, proliferative cells are allowed to become quiescent (at constant rate α) and quiescent cells are allowed to become proliferative (at constant rate β).

depending on the nature of the disease being modeled, parameter s is a phenomenological hyper-parameter of the model describing the relative strength of the chemotherapeutic.

The proliferating cell compartment contains cells at each of the four phases of cell cycle (gap period G1, synthetic period S, second gap period G2, and mitosis M) to reduce the complexity of the cellular states. While resting cells are affected to a small extent by cell-cycle specific chemotherapy, the model (5.1) assumes that the chemotherapy $f(t)$ affects only the proliferating cells. The model does not include details from other aspects of the patient's context, notably it ignores the effects of age, sex, spatial information of the tumour, and any applicable comorbidities.

In [79] the authors parameterize (5.1) with values that are suitable for describing breast cancer and ovarian cancer, as determined by mouse models. In [80] the authors provide an additional parameter set for determining the effect of chemotherapy on healthy bone marrow cells. This allows one to, for a given chemotherapy dosing schedule $f(t)$, model the effect of chemotherapy on both the healthy bone marrow cells and the malignant cancerous cells. Hence, by evolving two de-coupled copies of (5.1), one parameterised with values corresponding to a particular cancer and the other with bone marrow parameter values, we can monitor the cancer-killing effects of a chemotherapeutic schedule and the associated chemotherapeutic toxicity in the patient. The parameter values are summarised in Table 5.1. The system in (5.1) has only one equilibrium point: the extinction equilibrium when $P = Q = 0$. Under the parameter values from Table 5.1 the system in (5.1) has at least one negative eigenvalue irrespective of the value of $s\,f(t)$ (as long as $s > 0$, which is

a reasonable constraint given that s represents a relative strength of chemotherapy and is non-negative by assumption). In the absence of treatment, $s = 0$, the system has one positive eigenvalue as well. This indicates that the equilibrium is unstable in the absence of treatment. Indeed, the system predicts exponential growth under all parameter sets in the absence of treatment. If treatment is instead taken to be maximal $f(t) = 1$ and the eigenvalues are considered in the limit as $s \to \infty$, then under all three parameter sets both eigenvalues are negative. Hence, for strong enough treatment, the disease free equilibrium is achieved, otherwise the cell compartments grow without bound.

Bone marrow		
Parameter	Nominal value	Units
γ	1.470	days^{-1}
δ	0.000	days^{-1}
α	5.643	days^{-1}
β	0.480	days^{-1}
λ	0.164	days^{-1}
ρ_p^*	0.103	-
Breast cancer		
Parameter	Nominal value	Units
γ	0.500	days^{-1}
δ	0.477	days^{-1}
α	0.218	days^{-1}
β	0.050	days^{-1}
λ	0.000	days^{-1}
ρ_p^*	0.200	-
Ovarian cancer		
Parameter	Nominal value	Units
γ	0.6685	days^{-1}
δ	0.4597	days^{-1}
α	0.2225	days^{-1}
β	0.0500	days^{-1}
λ	0.0000	days^{-1}
ρ_p^*	0.3600	-

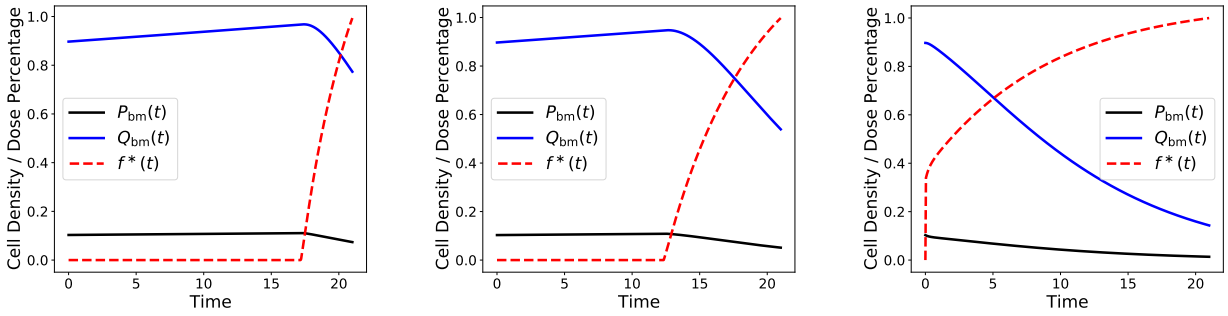
Table 5.1: Parameter values for breast cancer cells, ovarian cancer cells, and bone marrow cells as obtained from [79, 80], and values of ρ_p^* as determined by (5.10).

5.2.2 Chemotherapeutic Control

To determine the optimal chemotherapeutic control, we follow [80] and introduce an objective functional with the form

$$J_b(f) = \int_0^T \left[P_{\text{bm}}(t) + Q_{\text{bm}}(t) - \frac{b}{2}\left(1 - f(t)\right)^2 \right] \mathrm{d}t. \tag{5.2}$$

Maximizing this objective functional enables the derivation of an optimal chemotherapy dosing schedule of duration T days for a particular patient. In the notation of (5.2), P_{bm} and Q_{bm} refer to the proliferative and quiescent compartments of (5.1) parameterised to describe the behaviour of bone marrow. In effect, this leads to a chemotherapeutic schedule that biases the optimizer toward applying a larger dose of chemotherapeutic, as governed by $f(t)$, while also maximizing the total number of bone marrow cells in the patient (to reduce drug toxicity). The non-negative hyperparameter b is then a scaling factor representing the relative importance of these two mechanisms. If $b \gg 1$, then delivering the largest chemotherapeutic dose possible is the most desirable action for the optimizer, even at the cost of decimating the bone marrow cell count. Contrarily, if $0 \leq b \ll 1$ then the optimizer is biased toward preserving bone marrow even at the cost of lower cancer kill. Plots of such a chemotherapeutic dosing function f, obtained via the analytical method for deriving the continuous optimal control as in [80], for various b values are presented in Figure 5.3.



(a) $b = 0.5$ (b) $b = 1$ (c) $b = 2$

Figure 5.3: A plot of the proliferative cell proportion (black), the quiescent cell proportion (blue), and the optimal chemotherapeutic control $f^*(t)$ (dashed red) for different values of b. The objective functional used to achieve this optimal control is given via (5.2). Small values of b correspond to weighting preservation of the bone marrow as more important and larger values of b correspond to weighting total drug delivery as more important.

In this work, the particular functional form of (5.2) is not of primary concern. In particular, the presence of power of 2 in (5.2) ensures that the objective functional is

concave (and so has a unique maximizer) [80]. Certainly other forms could be suggested to achieve similar qualitative goals. For instance, consider the functional

$$J_b(f) = \int_0^T \left[P_{\text{bm}}(t) + Q_{\text{bm}}(t) - b\left(P_{\text{bc}}(t) + Q_{\text{bc}}(t)\right)\right] \mathrm{d}t. \tag{5.3}$$

In the notation of (5.3), P_{bc} and Q_{bc} refer to the proliferative and quiescent compartments of (5.1) parameterised to describe the behaviour of solid breast cancer tumours. Hence, the functional in (5.3) describes the minimisation of breast cancer cells while preserving the healthy, bone marrow cells. In this functional, the dependence on the chemotherapeutic dosing schedule f is implicitly included in the trajectories of P_{bm}, Q_{bm}, P_{bc}, and Q_{bc}. In any case, the exact formulation of this objective functional is an incredibly important choice for any modeler in a clinical context as it determines the metric by which the control is considered maximal and is outside the scope of this chapter.

While there are many methods in the field of optimal control theory that provide a methodology for obtaining such schedules, one could also employ techniques from reinforcement learning to discover chemotherapeutic dosing schedules. For instance, for a time t given the state vector s_t, to be defined later, and chemotherapeutic dose $a_t \in [0, 1]$, we define the immediate reward function as

$$R(s_t, a_t) = \int_t^{t+1} \left[P_{\text{bm}}(s) + Q_{\text{bm}}(s) - \frac{b}{2}\left(1 - a_t\right)^2\right] \mathrm{d}s \tag{5.4}$$

in order to elicit an analogous response in the reinforcement learner as achieved by the objective functional in (5.2). Explicitly, $J_b(f) = \sum_{t=0}^{T-1} R(s_t, a_t)$, where J_b is given in (5.2) for appropriate piecewise constant functions f. To proceed, we use the reward function in (5.4) in the following form of the Bellman equations (see, for instance, [99])

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma \sum_{s' \in \mathbb{S}} p(s' \mid s_t, a_t)\, Q^*(s', \operatorname*{argmax}_{a' \in \mathbb{A}}(Q^*(s', a'))) \tag{5.5}$$

to derive an optimal policy as defined by

$$\pi(s_t) = \operatorname*{argmax}_{a \in \mathbb{A}} Q^*(s_t, a). \tag{5.6}$$

This policy can then be used to derive an optimal chemotherapy dosing schedule according to

$$f^*(t) = \pi(s_t). \tag{5.7}$$

We provide a brief explanation of Equations (5.5-5.7 but direct readers to a more thorough source such as [99] for full details. As mentioned earlier, $R(s_t, a_t)$ represents the

immediate reward an agent receives for performing action a_t while in state s_t. In contrast, $Q^*(s_t, a_t)$ represents a valuation of performing action a_t while in state s_t. Notably, $Q^*(s_t, a_t)$ encodes the immediate reward $R(s_t, a_t)$, but also encodes the discounted future rewards. Similarly, for a given Q^* function the policy $\pi(s_t)$ describes the optimal action to perform in a given state s_t. As a result, $\pi(s_t)$ chooses an action a_t to maximize $Q^*(s_t, a_t)$ in a global manner as compared to the local process of choosing a_t to maximize immediate reward $R(s_t, a_t)$. As such, the maximal action in a given state, as valued by Q^*, may be one for which the payoff is not immediately obvious for multiple timesteps. The factor γ in (5.5) is the discount factor of future rewards. The parameter γ is taken such that $\gamma \in [0, 1]$ where $\gamma = 0$ corresponds to an agent that is focused on maximizing the immediate reward of their action and $\gamma = 1$ corresponds to an agent more concerned with increasing future reward than immediate. In general, a model describing a reinforcement learning environment may not be deterministic. In that regard, $p(s'|s_t, a_t)$ corresponds to the probability of ending up in state s' after taking action a_t in state s_t. For the model in (5.1), no such stochasticity exists. As such, it is assumed that $p(s'|s_t, a_t) = 1$ for exactly one $s' \in \mathbb{S}$ (namely $s' = s_{t+1}$). One can derive the drug dosing schedule $f^*(t)$ as in (5.7) by observing the state in some manner and then evaluating the policy at this state. For the case of the nominal patient, where the patient specific parameters are known, observing the state is as simple as integrating (5.1). For an already trained model, one would construct the state vector (5.8) for a patient (virtual or otherwise) and then evaluate the policy at this state.

For a continuous time reinforcement learning agent, the optimal dosing schedule learned by this process for a given parameterisation of (5.1) is identical to that derived via optimal control theory for that same parameterisation, as in [80]. Of particular importance, however, is that as (5.7) demonstrates, once a policy has been learned, one can derive an optimal chemotherapy schedule by merely evaluating the policy at the state. Importantly, the state one evaluates the policy at need not be a state seen during the learning process. Indeed, in our study we concern ourselves with training the reinforcement learning agent on only the nominal virtual patient and developing chemotherapy schedules for 200 different testing virtual patients. By leveraging state vector information from these 200 different testing virtual patients (patients which encode an environment over which the agent has not trained) the reinforcement learning agent is able to personalize the dose delivery function. As a result, it is important that we define our state vector as something that is both practically measurable and phenomenologically linked to the objective functional we wish to optimize. As indicated by (5.8), when deriving the optimal chemotherapy dosing schedule according to (5.7), we are passing the optimal policy a `wl` length window of measurements corresponding to bone marrow count relative to a time before treatment began as well as

the current day of treatment (in order to satisfy the Markov property).

$$s_t = \langle t, P_{\text{bm}}(t)+Q_{\text{bm}}(t), P_{\text{bm}}(t-1)+Q_{\text{bm}}(t-1), \ldots, P_{\text{bm}}(t-\texttt{wl}+1)+Q_{\text{bm}}(t-\texttt{wl}+1)\rangle \quad (5.8)$$

The particular value of `wl` is another hyperparameter to the process to consider. In contrast with the other hyperparameters listed in Table 5.2, this hyperparameter was chosen empirically to be $\texttt{wl} = 10$ as in [113] wherein the authors used a length 10 window of mean tumour diameters as the state vector of their learning agent.

5.2.3 Perturbed Virtual Patients

We generate sets of virtual patients according to the following strategy. We consider parameter values from Table 5.1 and construct virtual patients by perturbing these parameter values. To begin we note that $\delta = 0$ for the bone marrow parameters. As a result, we do not consider perturbing this value and treat δ as zero for all virtual patients. These perturbations are performed by scaling the mean parameter values in Table 5.1 by factors sampled from the space $[1-k, 1+k]$ uniformly with Latin hypercube sampling [56]. Latin hypercube sampling, a space filling technique for drawing random samples, is especially important when the number of samples drawn is small in comparison to the size of the sample space and when the parameters of interest are uncorrelated. Given the phenomenological nature of the remaining parameters we can assert that these parameters are uncorrelated. In particular, modifying any one of these parameters will create a distinct system under (5.1) that cannot be recovered by modifications to any number of the remaining parameters.

We now introduce some notation for describing the virtual patients. To begin, we let ξ_0 represent the nominal virtual patient. That is, $\xi_0 = (\gamma_{\text{bm}}, \delta_{\text{bm}}, \alpha_{\text{bm}}, \beta_{\text{bm}}, \lambda_{\text{bm}})$ from Table 5.1 where the "bm" subscript refers to the bone-marrow parameter values. In this work, we generated virtual patients at perturbation levels of $k = 0.15$, 0.20, and 0.25, where k corresponds to the percent-change strength of perturbation. We generated six total sets of virtual patients. For the purpose of interpolating the nearest training neighbour optimal controller, we generated 1000 virtual patients at the 15%, 20%, and 25% perturbation strength level for testing purposes which we denote by ζ_i^k where $1 \leq i \leq 1000$ denotes the index of the virtual patient and $k \in \{0.15, 0.20, 0.25\}$ denotes the maximal strength of the perturbation. Similarly, we generated 200 virtual patients at the 15%, 20%, and 25% perturbation strength level for the purpose of testing the controllers, these patients we denote by θ_i^k where $1 \leq i \leq 200$ again represents the index of the virtual patient and $k \in \{0.15, 0.20, 0.25\}$ represents the maximal strength of the perturbation. The reinforcement learning agent was only trained on the nominal virtual patient and not virtual

patients from the training or testing sets. The training virtual patients were only utilised for the crafting of the NTNOC optimal control strategy discussed in Section 5.3. In this regard, the testing virtual patients serve as a metaphor for the unknown patient-specific parameters of any particular patient in clinic. Both the testing and training virtual patients, for the non-zero parameters γ, α, β, and λ, are visualised in Figure B.1.

5.2.4 Training Process

We numerically solve the Bellman equation, (5.5), by employing neural networks as in the deep double Q-learning algorithm [102]. Double deep Q-learning is not the only algorithm by which one can solve this form of the Bellman equation. We chose this algorithm for a number of reasons: primarily, we expect that both the presence of the experience-replay buffer and the $Q(s, a)$ valuation is crucial for the algorithm to be able to successfully approximate the optimal action in the presence of environmental noise. Indeed, the success of this method relies on the ability of the algorithm to approximate the value of state/action pairs that differ from those seen in the optimal treatment of the nominal patient. By maintaining a buffer and valuation of the state/action pairs explored while deriving the nominal treatment, the algorithm is able to better approximate a larger swathe of the domain of Q. There are many algorithms that satisfy these requirements such as deep deterministic policy gradient or (single) deep Q-learning [62, 72]. However, deep policy gradient is a more computationally expensive method that can produce continuous controls, whereas we are primarily concerned with discrete dose values in this study. Similarly, deep Q-learning has been observed to be more likely to select overestimated values, resulting in overtly optimistic value estimates, in a capacity that is avoided in deep double Q-learning. [102]. In particular, we represent the Q function from the Bellman equation, (5.5), as a neural network. As a result, after training the network, the specific form of Q is the same for each testing virtual patient. However, as in (5.7), by supplying the bone marrow measurements for the testing virtual patient, the network can produce a personalised dose schedule that is different for each virtual patient. In terms of the architecture of the Q network, we take the network to have 10 inputs neurons (as determined by the length of the state vector), $\texttt{hd}_1$ neurons in the first hidden layer, $\texttt{hd}_2$ neurons in the second hidden layer, and 11 neurons in the output layers (corresponding to dose strength range from 0 to 1 inclusive in 0.1 increments). Each neural layer is activated with a rectified linear unit. We use batch-learning with a batch size of `bs` to minimize the mean squared error between the right and left hand sides of (5.5) via an Adam optimizer with learning rate α. In Section 5.2.5 we discuss how we decide upon the values of these hyperparameters and list the particular values in Table 5.2. To train the network we first parameterize the

system in (5.1) by the nominal parameter set ξ_0. Next, we run 5,000 exploratory time steps of the simulation performing random actions in order to fill the experience replay buffer. After every $T = 21$ time steps, the environment is reset by returning the differential equation model, (5.1), back to its initial conditions (as dictated by (5.10)). In particular, the initial state for the reinforcement learning algorithm is a length eleven vector where the first entry is a 0 and the remainder are unit entries (as the (5.10) initial conditions sum to 1). This window length of 10 for the bone marrow measurements was chosen empirically, as in [113]. For each time step of the simulation we choose a chemotherapy dose according to our network via an ϵ-greedy algorithm. We anneal ϵ linearly from $\epsilon = 1$ to $\epsilon = 0.01$ over 25,000 time steps. The ϵ-greedy algorithm was only implemented during training, i.e. during evaluation the policy selection is deterministic as in (5.6). After selecting a dose $a \in \mathbb{A} = \{0, 0.1, \ldots, 1\}$, we apply the chemotherapy dose to the patient by holding $f(t) = a$ constant over the timestep and evolving (5.1) (as such, we discretize not only in dose but in time as well). Next, we record a tuple of the state, action, reward, new state values. We then select a random batch of previously observed tuples and use them to approximate the right hand side of the Bellman equation, (5.5), in order to obtain a target for training the network.

Hyperparameter Values		
Parameter	Value	Description
$\texttt{hd}_1$	64	Dimension of first hidden layer in the neural network
$\texttt{hd}_2$	96	Dimension of second hidden layer in the neural network
γ	0.9553	Discount factor from the Bellman equation (5.5)
α	0.003809	Learning rate for the Adam optimizer
`bs`	96	Batch size for the Adam optimizer
`wl`	10	Window length for relative bone marrow measurements

Table 5.2: Hyperparameter values for the learning process. The parameters $\texttt{hd}_1$, $\texttt{hd}_2$, γ, α, and `bs` were determined by the Bayesian optimizer whereas `wl` was chosen empirically.

This process is eventually stopped if the differential equation environment has been reset 50,000 times or if the best reward has not improved over the last 500 epochs. This constitutes one training run of the system. We perform 5 such training runs recording the run that achieved the largest objective functional score under (5.2).

5.2.5 Hyperparameter Tuning

While our system has many model specific parameters, there are also a number of hyper-parameters introduced during the training process. These are the learning rate for the Adam optimizer (α), the dimension of the two hidden layers ($\texttt{hd}_1$ and $\texttt{hd}_2$, respectively), the discount rate γ in the Bellman equation (Eq. (5.5)), and the batch size of the Adam optimizer used for learning (`bs`). In order to ensure optimal convergence and stability of the resultant networks, we must carefully select these values. For a single set of these five hyper-parameters we must execute the entire training process over again. Such a process is computationally extensive rendering a brute-force grid-search approach to hyperparameter optimisation unfeasible. To that end, we instead use Bayesian optimisation to explore this five-dimensional hyperparameter space more efficiently. We allow our Bayesian optimizer to sample 100 such hyperparameter samples from this hyperparameter space and perform the training process for each hyperparameter set. The Bayesian optimizer chose $\texttt{hd}_1$ and $\texttt{hd}_2$ from the set $\{64, 96, \ldots, 256\}$, `bs` from the set $\{32, 64, \ldots, 128\}$, α from the interval $(10^{-4}, 10^{-1})$, and γ from the interval $(0, 1)$.

In Figure 5.4 we see the distribution of the objective functional score under (5.2) for the 100 reinforcement learning agents under this hyperparameter tuning process. In particular, we note the cluster of 36 agents that converged to the network architecture with the theoretical maximal objective functional value, as determined by running a discretised version of the optimal control problem from [80] with the APOPT algorithm (as implemented by GEKKO [3, 46]). For a point of comparison, we calculated the expected score achievable by a random agent by calculating the mean value of the score obtained in 1,000,000 simulations where a dose from $\{0, 0.1, \ldots, 1.0\}$ was uniformly selected at each time step. This resulted in a mean objective functional value of 0.6806 with a standard deviation of the mean of 0.04811.

To ascertain the identifiability and stability of these parameters, we consider the distribution of parameters that result in such objective functional value. To that end, in Figure 5.5 we consider the distribution of each individual hyperparameter and contrast this with the distribution of such hyperparameters from the agents that converged to network architecture that achieve an objective functional value value within 5% of the maximal possible reward. Importantly, we recognize that of the five hyperparameters, there is not a tight distribution after conditioning on objective functional value score. In fact, only the discount factor γ produces a conditioned distribution that is statistically different than the un-conditioned distribution (two-sample Kolmogorov-Smirnov p-value of approximately 0.001 [47]). This suggests that the particular values of the size of the hidden dimensions, learning rate, and batch size are not terribly sensitive parameters for the training of this

reinforcement learning agent.

The Bayesian optimizer determined an optimal hyperparameter choice of $\texttt{hd}_1 = 64$, $\texttt{hd}_2 = 96$, $\gamma = 0.9553$, $\alpha = 0.003809$, and $\texttt{bs} = 96$. Though, as the above discussion demonstrates, it is only the choice of γ that appeared to have any particularly strong impact on the convergence of the training process. A γ value close to 1 can be interpreted as representing an agent with a far horizon [99]. In particular, such an agent is less concerned with the immediate reward of a particular action and more concerned with the long-term, cumulative reward obtained by maximizing (5.2) over all time.

5.3 Results

5.3.1 Derivation of the Proliferative Fraction

We begin modeling the proliferative and quiescent components of (5.1) under the assumption that the tumour has evolved in the absence of any chemotherapeutic agent until a steady state, in terms of the proportion of these cells, has been reached. To that end, we define the proliferative ratio of the tumour at time t and the steady-state proliferative ratio as

$$\rho_p(t) = \frac{P(t)}{P(t) + Q(t)} \quad \text{and} \quad \rho_p^* = \lim_{t \to \infty} \rho_p(t),$$

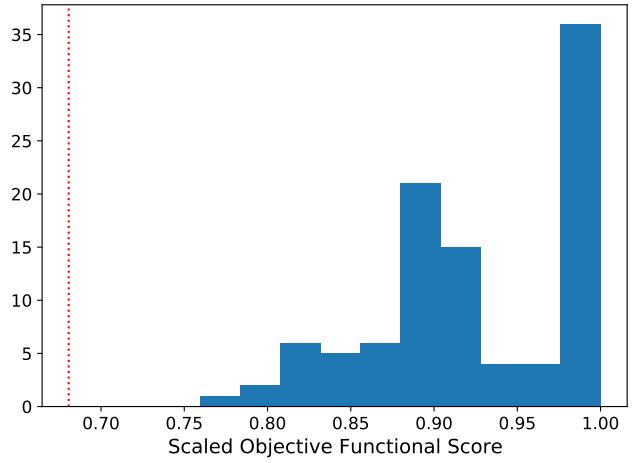
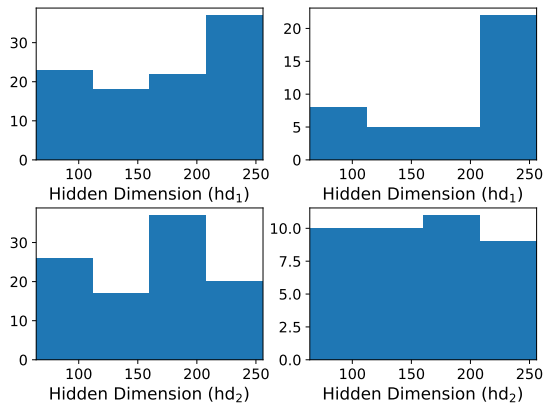
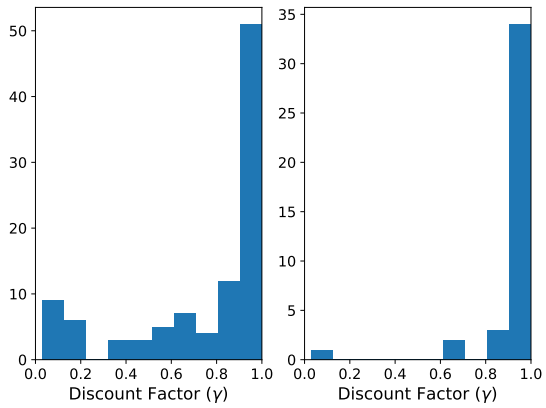


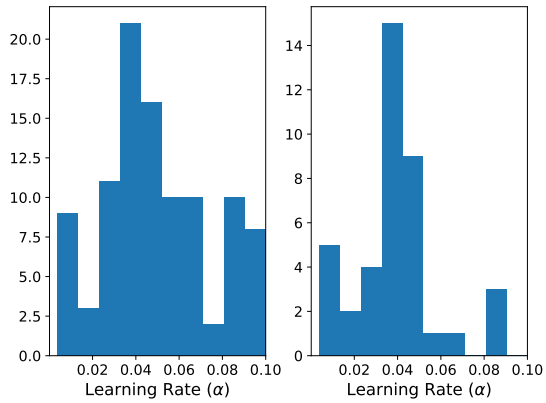
Figure 5.4: A histogram demonstrating all the scores obtained via the reinforcement learning process. The red dotted line indicates the expected score of a random agent.



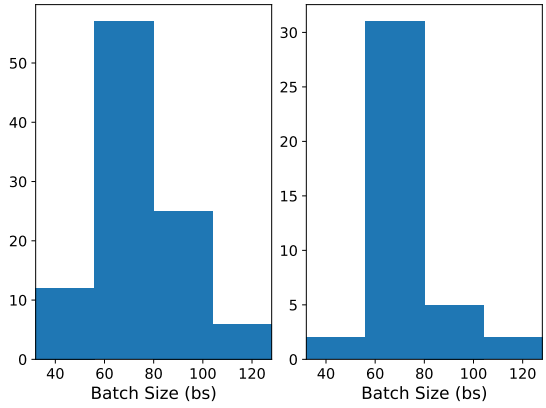
(a) Distributions of the size of the hidden dimensions of the neural network architecture.



(b) Distributions of the discount factor γ used in the Bellman equation, (5.5).



(c) Distributions of the learning rate used in the Adam optimizer for the neural network.



(d) Distributions of the batch size used in Adam optimizer for the training of the neural network.

Figure 5.5: In all figures, the distributions on the left represent the total distribution of the hyperparameter explored by the Bayesian hyperparameter optimizer. In contrast, the distributions on the right in each figure indicate the distribution of the hyperparameter conditioned on the objective functional value being within 5% of the maximal theoretical score.

respectively. To analytically calculate the closed form solution of the steady-state proliferative ratio in the absence of a chemotherapeutic, we set $s = 0$ and consider

$$
\begin{aligned}
0 &= \rho_p'(t) \\
&= \frac{P'(t)}{P(t)+Q(t)}\,\frac{Q(t)}{P(t)+Q(t)} - \frac{Q'(t)}{P(t)+Q(t)}\,\frac{P(t)}{Q(t)+P(t)} \\
&= \frac{P'(t)}{P(t)+Q(t)}\,(1-\rho_p^*) - \frac{Q'(t)}{P(t)+Q(t)}\,\rho_p^* \\
&= (\delta-\gamma-\lambda)\,\rho_p^{*2} + (\gamma+\lambda-\beta-\alpha-\delta)\,\rho_p^* + \beta. \qquad (5.9)
\end{aligned}
$$

Thus, ρ_p^* is a root of the quadratic in (5.9). For the parameter values presented in Table 5.1 the quadratic in (5.9) has only one positive (real) root, namely

$$
\rho_p^* = \frac{1}{2}\,\frac{-\lambda-\gamma+\alpha+\delta+\beta-\sqrt{(\lambda+\gamma-\alpha-\delta-\beta)^2-4\,(\delta-\lambda-\gamma)\,\beta}}{\delta-\lambda-\gamma}. \qquad (5.10)
$$

The values of ρ_p^* corresponding to the parameters for ovarian cancer, bone marrow, and breast cancer are included in Table 5.1. Thus the initial data for (5.1) considered in this study are given by $P(0)=\rho_p^*$ and $Q(0)=1-\rho_p^*$.

5.3.2 Local Sensitivity Analysis

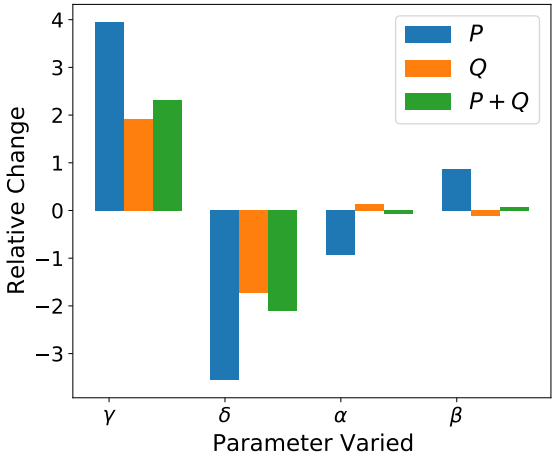
Here we investigate the sensitivity of the outputs for the tumour growth inhibition model, (5.1), to perturbations in the nominal parameter values of the model-specific parameters presented in Table 5.1. To compute the sensitivities, we change the values of the parameters $\gamma, \delta, \alpha, \beta,$ and λ one-at-a-time by a small amount, Δp. We take Δp to be $+1\%$ of the nominal parameter value p_0. Then the relative sensitivity of each model population $x = \langle P(T), Q(T)\rangle$ for the parameter is calculated as follows:

$$
R_{x,p} = \frac{(x-x_0)/x_0}{(\Delta p)/p_0}, \qquad (5.11)
$$

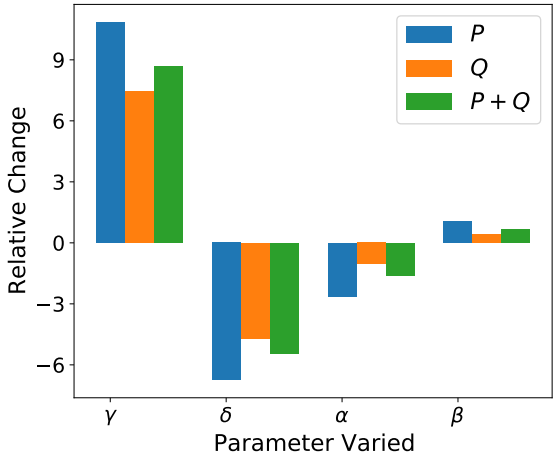
where subscripts denote nominal values. The initial conditions of the simulations were recalculated according to (5.10) for each perturbed parameter value and the simulations were run until $T = 21$ days. While a different value of T could have been chosen, the results are qualitatively similar. In particular, in [80] the authors consider $T = 7$, 14, and 21 and demonstrate the similarity of all three choices. By taking $T = 21$, we are allowing more time for the drug to accumulate ensuring a greater degree of cell kill. However, if T were

taken to be very large, then the exponential growth due to the instability of the equilibrium would become a more dominant effect. In this way the choice of $T = 21$ is a convenient trade-off that matches previous work and is qualitatively similar to other, nearby, values of T. Certainly other choices of T are possible. In practice, treatment length is something that would be informed by both standard of care for the particular cancer and other factors from the particular patient and clinician. As a result, we did not consider optimizing the choice of T in any capacity. We plot the results in Figure 5.6.

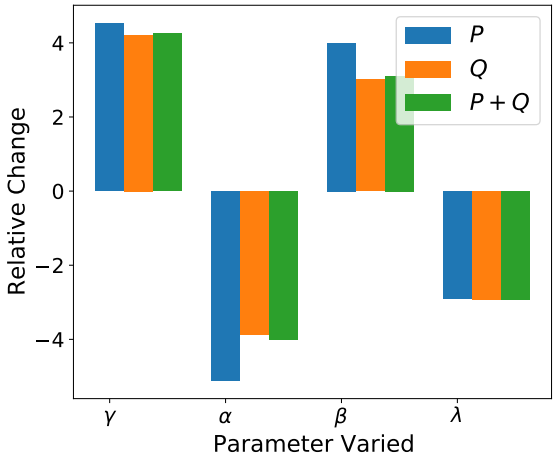
Importantly, the results of Figure 5.6 indicate that the model exhibits substantial sensitivity due to relatively small perturbations in the patient-specific parameter values. This type of sensitivity is common in models that experience regimes of exponential growth, which are common in cellular models of cancer [26]. For the parameter sets corresponding to breast cancer, Figure 5.6a indicates a mean (absolute) change of roughly 2.3% in P cells, 0.97% in Q cells, or 1.14% in all cell types given a 1% perturbation to a singular parameter. For ovarian cancer the model is even more sensitive, demonstrating a mean (absolute) change of roughly 5.3% in P cells, 3.4% in Q cells, or 4.1% in all cell types given a 1% perturbation to a singular parameter. For bone marrow, similar extreme sensitivities are observed with a mean (absolute) change of roughly 4.1% in P cells, 3.5% in Q cells, or 3.6% across all cell types. Importantly, Figure 5.6 demonstrates that even for the least sensitive parameter set (the breast cancer parameter set), small perturbations to individual parameters can still elicit large differences in the evolution of a tumour if one is unlucky enough that such a perturbation occurred in either γ or δ (the self-renewal and death rate of proliferative cells, respectively). Perturbing parameters in this way also perturbs the initial conditions, which is contrary to what is typically seen in the sensitivity analysis. The reason we consider this is because we consider the initial conditions (the proliferative fraction of the tissue) to be a noisy measurement as well. If instead we were to perform the above sensitivity analysis but using the initial conditions of the unperturbed parameters for a model with perturbed parameters, the results will not change quantitatively by much at all. The maximum $P + Q$ change would go from roughly 2.32% in breast cancer to 2.28% when the initial conditions are not perturbed. Similarly the maximum change in $P + Q$ for the ovarian cancer parameter set would decrease from 8.7% to 7.9% and for the bone marrow parameter set from 4.24% to 4.23%. Hence this choice of perturbing initial conditions is experientially informed by the particular context and does not have a large quantitative effect on the output of the sensitivity analysis.



(a) Relative sensitivity of the breast cancer parameter values.



(b) Relative sensitivity of the ovarian cancer parameter values.



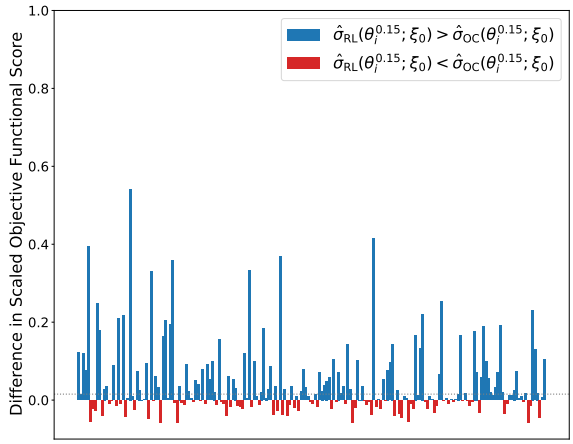
(c) Relative sensitivity of the bone marrow parameter values

Figure 5.6: Relative sensitivity of (5.1) under the parameter sets from Table 5.1. Parameters with zero value (δ for bone marrow and λ for breast and ovarian cancer) were ignored and not displayed in this figure.

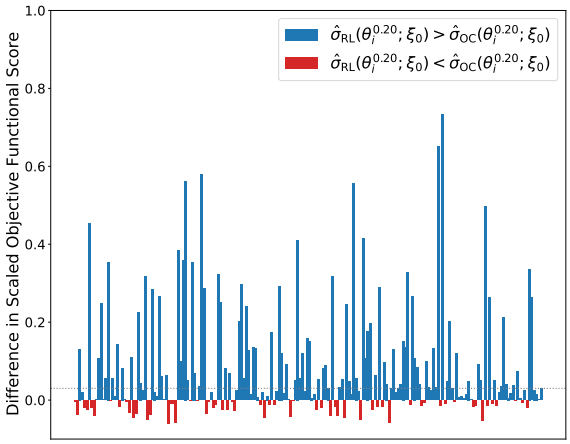
5.3.3 Contrasting a nominal reinforcement learning agent with a nominal optimal controller

We first begin by training a reinforcement learner on the nominal parameter set from Table 5.1 using the hyperparameters for the training method from Table 5.2. During training, the reinforcement learning agent only interacted with the environment from (5.1) parameterised by the nominal set. Similarly, as a point of comparison, we used the APOPT algorithm from the GEKKO Python library to solve the discretised optimal control problem on the nominal parameter set [3, 46]. These two agents, one a reinforcement learning agent and the other a traditional optimal controller, were kept blind to the testing and training virtual patients. That is, the reinforcement learning agent was trained offline on an environment parameterised by ξ_0 before the policy derived was tested on environments parameterised by θ_i^k for all i and k. Similarly, the traditional optimal control was derived for patient ξ_0 before being applied to the testing patients θ_i^k for all i and k. We then applied chemotherapeutic dosing schedules derived from both methods on the 600 testing virtual patients (200 virtual patients each at the 15%, 20%, and 25% perturbation strength level). We define $\sigma_{\mathrm{RL}}(\theta_i^k;\xi_0)$ to refer to the value under the objective functional in (5.2) achieved by this reinforcement learning agent when applied to patient θ_i^k after training the agent on an environment parameterised by the nominal patient ξ_0. We similarly define $\sigma_{\mathrm{OC}}(\theta_i^k;\xi_0)$ to be the score achieved by applying the optimal control for patient ξ_0 to patient θ_i^k. In order to scale the scores of these trials, we separately solved the discretised optimal control problem on these testing virtual patients using the APOPT algorithm from GEKKO. Importantly, these 600 solutions were only used to ascertain the maximal possible objective functional value in order to scale the result of the blind agents. We define $\sigma(\theta_i^k)$ denote the maximal value of the objective functional in 5.3 when parameterised by patient θ_i^k. We let $\hat{\sigma}_{\mathrm{RL}}(\theta_i^k;\xi_0) = \sigma_{\mathrm{RL}}(\theta_i^k;\xi_0)/\sigma(\theta_i^k)$ denote this scaled objective functional score. As a result, a score of 1 is the maximal score theoretically obtainable under the objective functional for the discrete problem by either solution method. In general, $\hat{\sigma}_{\mathrm{RL}}(\theta_i^k;\xi_0) \leq 1$. Similarly $\hat{\sigma}_{\mathrm{OC}}(\theta_i^k;\xi_0) = \sigma_{\mathrm{OC}}(\theta_i^k;\xi_0)/\sigma(\theta_i^k) \leq 1$ represents the scaled score of the optimal control agent. Finally, we compared the blind agents results by applying their derived chemotherapy strategies to the testing virtual patients, scaling the output according to the previously ascertained maximal possible reward. The results of this are presented in Figure 5.7 for the 3 different perturbation strength levels.

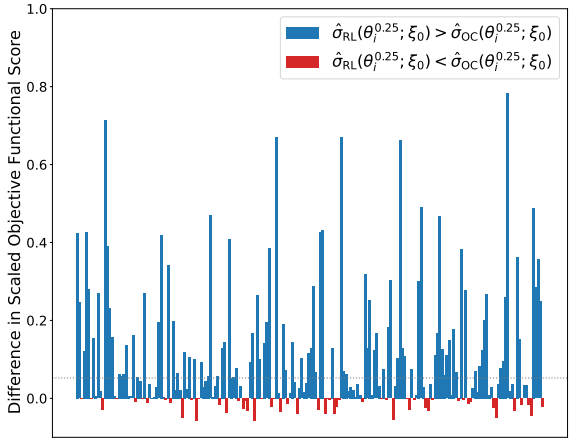
Notably, the chemotherapy dosing schedule determined via optimal control for the nominal parameter set is a particular function f^* that is the same for each virtual patient. In effect, each virtual patient is treated with the therapy schedule that is optimal for the mean-valued patient. In contrast, in the reinforcement learning derived schedule, the policy



(a) Difference in scaled objective functional score for virtual patients at 15% perturbation strength.



(b) Difference in scaled objective functional score for virtual patients at 20% perturbation strength.



(c) Difference in scaled objective functional score for virtual patients at 25% perturbation strength.

Figure 5.7: Bar plots of the difference between the scores obtained by the reinforcement learner derived policy and the scores obtained by the optimal control derived policy on all test virtual patients (i.e. bar plots of $\hat{\sigma}_{\mathrm{RL}}(\theta_i^k;\xi_0) - \hat{\sigma}_{\mathrm{OC}}(\theta_i^k;\xi_0)$). Testing patients where the reinforcement learner outperformed the optimal controller are marked in blue and patients where the optimal controller outperformed the reinforcement learner are marked in red. The dotted grey lines in each plot indicate the difference of the median scaled scores of the reinforcement learner and the optimal controller.

from (5.6) is the same for each virtual patient, but that policy is being fed a 10 day window of relative bone marrow cell counts from each virtual patient as a state vector. In particular, the nominal optimal controller is an open-loop controller whereas the reinforcement learning agent is a feedback controller. As a result, we are allowing the reinforcement learner to refine its dosing schedule given this information. By doing so we are able to acquire a dosing schedule that is more robust to perturbations in these unknown, assumed to be unmeasurable, patient-specific model parameters by allowing refinements to be made based on a more easily measurable aggregate metric. Hence, the fact that the reinforcement learning agent is a feedback controller is exactly why it is able to utilize information from the state vector in the design of these dose delivery schedules. Importantly, the state vector for the reinforcement learner is the sum of the P_{bm} and Q_{bm} compartments of (5.1) at discrete time points (in this case, daily) and not the individual measurements of P_{bm} and Q_{bm} separately. Given the scaling of (5.1) under the initial conditions from (5.10), these measurements are taken relative to the bone marrow mass prior to treatment, and so absolute measurements are not required. See Figure B.4 for a visualisation of these schedules on different virtual patients.

To quantify the performance differences of the two dose schedule processes over the testing virtual patients, we compared the distributions of scores with the non-parametric one-sided Wilcoxon signed-rank test [108]. For the cases represented in Figures 5.7a, 5.7b, and 5.7c we considered the alternative hypothesis to be that the median scaled score obtained by the reinforcement learner is larger than the median scaled score obtained by the optimal controller (i.e. the alternative hypothesis is median $(\hat{\sigma}_{\text{RL}}(\theta;\xi_0)) >$ median $(\hat{\sigma}_{\text{OC}}(\theta;\xi_0))$). We found at the 15% perturbation strength level a Wilcoxon statistic of 14681 corresponding to a p-value on the order of 10^{-9}, at the 20% perturbation strength level we found a Wilcoxon statistic of 16528 corresponding to a p-value on the order of 10^{-15}, and at the 25% perturbation strength level we found a Wilcoxon statistic of 17551 corresponding to a p-value below machine precision. In all cases, we reject the null hypothesis and conclude that the reinforcement learning agent produces chemotherapeutic schedules with a higher median score on perturbed patients than the optimal controller. We notice that as the perturbation strength increases, the difference in the median and mean scaled scores increases as well from a difference in medians of 0.011 in the 15% case (difference of means of 0.045) to a difference in medians of 0.044 (difference of means of 0.108) in the 25% case. Hence, as the strength of the perturbation increases over this range, the reinforcement learner outperforms the optimal controller even further.

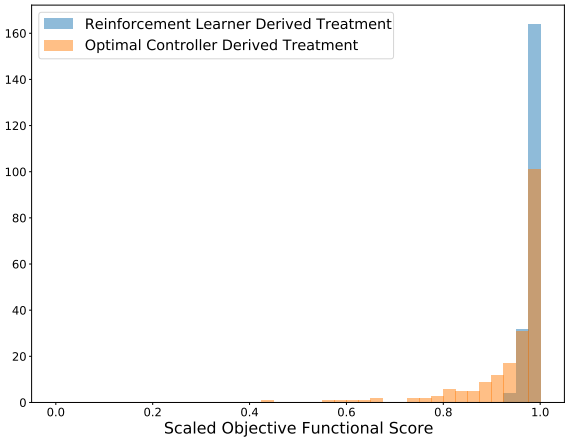
In Figure 5.8 we present the histograms of the scaled scores for each blind agent on the 600 different virtual patients. The histograms are semi-transparent in order to aid comparison where the blue colour represents the reinforcement learning agent and the

orange colour the APOPT derived optimal controller agent. We note that the reinforcement learning agent has a large cluster of treatments in the 97.5% – 100% optimal bin (164 out of 200 in the 15% case, 165 out of 200 in the 20% case, and 161 out of 200 in the 25% case) and all treatments fall within 7.5% of the theoretical maximum. These scores are achieved without training on these virtual patients directly. In contrast, the optimal controller derived treatment is much more diffuse. As the strength of perturbation increases, the average score of the reinforcement agent derived schedule remains within 1.2% of optimum, while the average optimal control derived score decreases dramatically from 0.941 in the 15% case, to 0.902 in the 20% case, and finally to 0.879 in the 25% case. In particular, this suggests that the increase in performance of the reinforcement learner as a result of perturbation strength is due to the reinforcement learning agents' capacity to remain non-sensitive to these perturbations, in contrast to the sensitivity seen in the schedules derived by the optimal controlling agent.

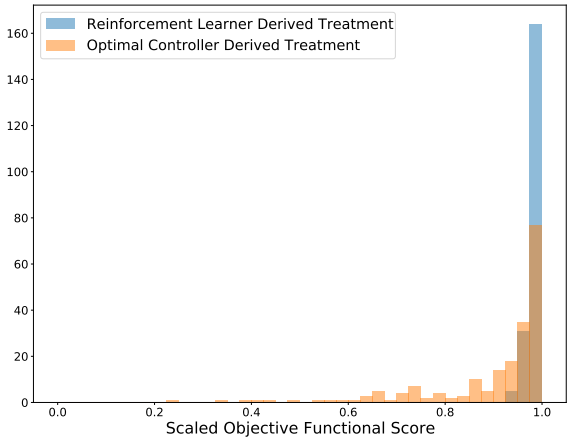
5.3.4 Contrasting a nominal reinforcement learning agent with a nearest neighbour interpolated optimal controller

The results of the previous subsection indicate that the reinforcement learning agent produces schedules that are more robust to perturbations in the unknown parameters. We noted that the reinforcement learning agent is capable of customizing these schedules for each individual patient, not by measuring the patient specific parameters directly, but by customizing the response via a more easily measurable metric. In this section, we consider a different training process that allows the optimal controller agent a comparable level of customisation.

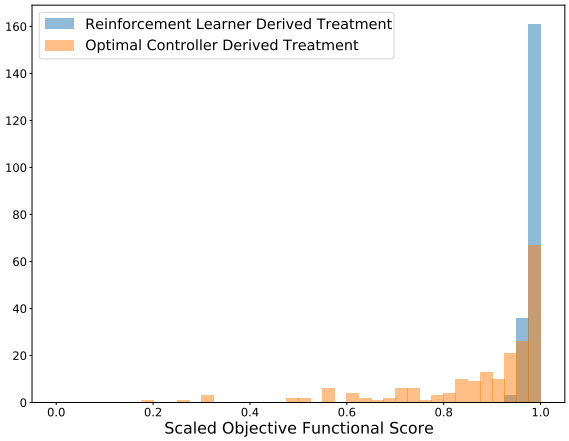
For this comparison, we kept the reinforcement learning agent exactly the same as in the previous section: the agent was trained offline on an environment parameterised by ξ_0 before being applied as a test to environments parameterised by θ_i^k for all k and i. For the optimal controller comparison, we begin by solving the discrete optimal control problem on all 1000 training virtual patients for each perturbation strength (i.e. the optimal control was determined for patient ζ_i^k for all k and all i). We then log the state vector from (5.8) for each timestep of treatment. For a fixed perturbation strength k we first calculated the state vector s_{t_i} for each of the 200 testing patients. We then applied a chemotherapeutic dose by consulting the table of states from the training patients at time t_i. The dose was selected from the training patient whose state vector was closest to the current testing patient state vector (where distance was measured by the Euclidean metric). Note that the state vector is as in 5.8 and as such contains a length $\texttt{wl} = 10$ moving window of relative



(a) At 15% perturbation strength the reinforcement agent produces a schedule that produces median scaled scores of 0.991 while the optimal controller derived schedule produces median scaled scores of 0.976.



(b) At 20% perturbation strength the reinforcement agent produces a schedule that produces median scaled scores of 0.992 while the optimal controller derived schedule produces median scaled scores of 0.962.

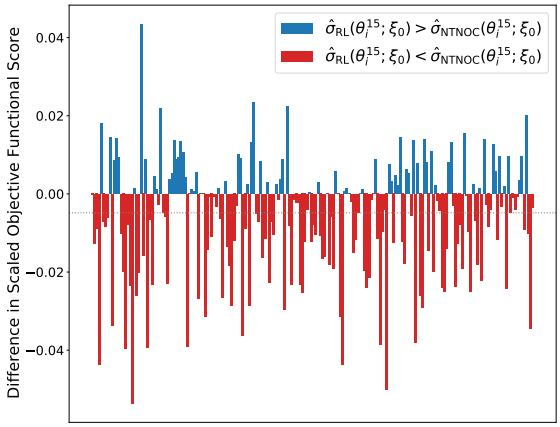


(c) At 25% perturbation strength the reinforcement agent produces a schedule that produces median scaled scores of 0.991 while the optimal controller derived schedule produces median scaled scores of 0.940.

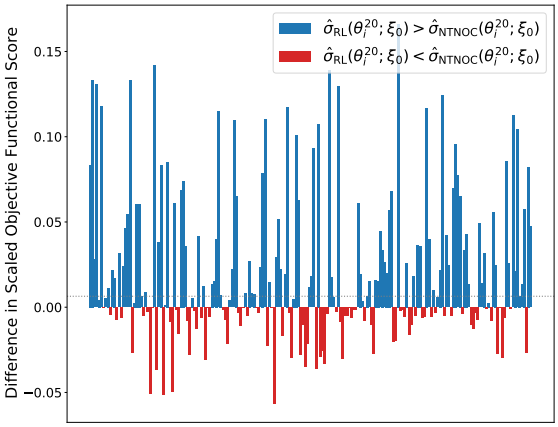
Figure 5.8: Histograms of the scores achieved by the various agents on the 200 testing virtual patients. Bin sizes were chosen to correspond to 0.025. In particular, the reinforcement learning agent is much more robust toward perturbation in parameter values, consistently producing dosing schedules scoring within 7.5% of the theoretical maximal score.

bone marrow measurements. The result of this nearest training neighbour optimal controller (NTNOC) was an agent that could also customize chemotherapeutic dosing strategies for each of the 200 testing virtual patients (θ_i^k) based off of knowledge gained by traversing the training virtual patient space (ζ_i^k). Hence, while the optimal controller presented in the previous section was an open-loop controller, the NTNOC is able to incorporate feedback from the environment. We define $\sigma_{\text{NTNOC}}(\theta_i^k; \zeta^k)$ to represent the score obtained in an environment parameterised by patient θ_i^k under the objective functional in 5.2 achieved by an NTNOC agent trained on the set $\zeta^k = \{\zeta_i^k \mid 1 \leq i \leq 1000\}$. Similarly, we define $\hat{\sigma}_{\text{NTNOC}}(\theta_i^k; \zeta^k) = \sigma_{\text{NTNOC}}(\theta_i^k; \zeta^k)/\sigma(\theta_i^k) \leq 1$ to be the scaled objective functional score. The reinforcement learning agent that the NTNOC agent is being compared to only ever interacted with a differential equation environment parameterised by the nominal parameter set ξ_0. Ostensibly, more distribution level information is directly afforded to the NTNOC than was afforded to the reinforcement learning agent. The reinforcement learning agent is only able to customize treatment strategies based off the states learned by providing non-optimal doses to the nominal virtual patient (ξ_0) during training. The results of this comparison are presented in Figure 5.9. In particular, we note that the same general trend from Figure 5.7 is repeated: namely, as the perturbation strength increases the relative performance of the reinforcement learning agent also increases. However, in contrast to Figure 5.7, we note that at the 15% level the nearest training neighbour optimal controller outperforms the reinforcement learning agent (with a one-sided Wilcoxon signed-rank test p-value on the order of 10^{-5}). Indeed, the mean value of the differences plotted in Figure 5.9a occurs at approximately -0.007, indicating that, in a mean value sense, the nearest training neighbour optimal controller produces strategies that are 0.007 points closer to the optimal score of 1 than the scores of the schedules produced by the reinforcement learning agent.

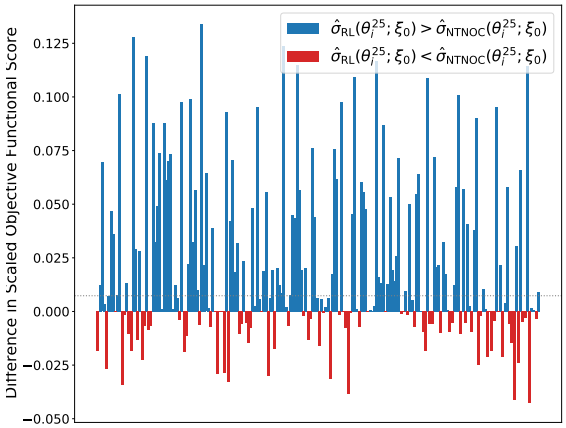
Again we compare the distributions of scores with the one-sided Wilcoxon signed-rank test, though for the case represented in Figure 5.9a, we consider the alternative hypothesis to be that the nearest training neighbour optimal controller produces schedules with higher median scaled score than that of the reinforcement learning agent (i.e. the alternative hypothesis is median $\left(\hat{\sigma}_{\text{NTNOC}}(\theta; \zeta^k)\right) >$ median $\left(\hat{\sigma}_{\text{RL}}(\theta; \xi_0)\right)$). We found at the 15% perturbation strength level a Wilcoxon statistic of 6315 corresponding to a p-value on the order of 10^{-5}. Hence we reject the null hypothesis and conclude that, at the 15% perturbation level, that the NTNOC produces chemotherapeutic schedules with higher median scaled score than those produced by the reinforcement learning agent. For the cases represented in Figures 5.9b and 5.9c we consider a different alternative hypothesis: namely that the reinforcement learning agent produces chemotherapeutic schedules with higher median scaled score than those produced by the NTNOC (i.e. the alternative hypothesis



(a) Difference in scaled objective functional score for virtual patients at 15% perturbation strength.



(b) Difference in scaled objective functional score for virtual patients at 20% perturbation strength..

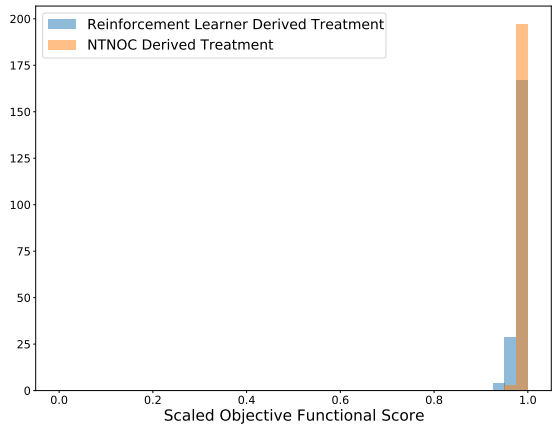


(c) Difference in scaled objective functional score for virtual patients at 25% perturbation strength.

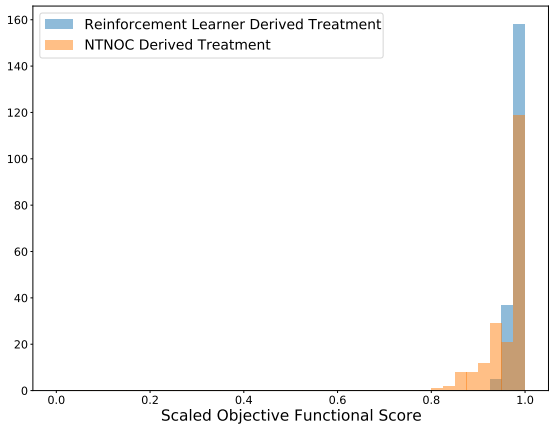
Figure 5.9: Bar plots of the difference between the scores obtained by the reinforcement learner derived policy and the scores obtained by the nearest training neighbour optimal controller on all test virtual patients (i.e. bar plots of $\hat{\sigma}_{\text{RL}}(\theta_i^k;\xi_0) - \hat{\sigma}_{\text{NTNOC}}(\theta_i^k;\zeta^k)$). Testing patients where the reinforcement learner outperformed the optimal controller are marked in blue and patients where the optimal controller outperformed the reinforcement learner are marked in red. The dotted grey lines in each plot indicate difference in the median values of the scores obtained by the reinforcement learning agent and those obtained by the nearest training neighbour optimal controller.

is median $(\hat{\sigma}_{\mathrm{RL}}(\theta;\xi_0)) >$ median $\left(\hat{\sigma}_{\mathrm{NTNOC}}(\theta;\zeta^k)\right)$. Then, at the 20% perturbation strength level we found a Wilcoxon statistic of 13023.5 corresponding to a p-value on the order of 10^{-8}, and at the 25% perturbation strength level we found a Wilcoxon statistic of 14593.5 corresponding to a p-value on the order of 10^{-9}. In these two cases we reject the null hypothesis and conclude the reinforcement learning agent produces chemotherapeutic schedules with higher median scaled score on perturbed patients than the NTNOC. In this situation, the nearest training neighbour optimal controller is able to produce schedules more competitive with the reinforcement learning agent than those produced by the nominal optimal controller. In the 15% case, the NTNOC outperforms the reinforcement learning agent by a small margin (difference in median scores of 0.003 in favour of the NTNOC) whereas the reinforcement learner outperforms the NTNOC in the 20% case (difference in median scores of 0.077 in favour of the reinforcement learner) and the 25% case (difference in median scores of 0.060 in favour of the reinforcement learner). While the NTNOC produces more robust schedules for small perturbations, such schedules seem to only slightly outperform the schedules produced by the reinforcement learning agent. In contrast, for medium perturbations around 20% and 25%, the reinforcement learning agent outperforms the NTNOC.

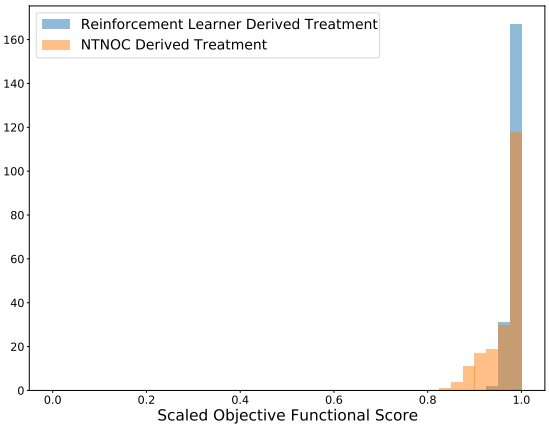
In Figure 5.10 we concern ourselves with, once again, examining the histograms of the scores of these two agents. As before, we note that the reinforcement learner derived schedules are robust to these perturbations in patient specific parameter values, which is the source of the success in Figure 5.9b and Figure 5.9c. However, in contrast to Figure 5.8, we note that the nearest training neighbour optimal controller produces schedules whose scores produce a histogram that is less diffuse than that produced by the nominal optimal controller (standard deviations of $(0.007, 0.06, 0.06)$ for the nearest training neighbour optimal controller at the 15%, 20%, and 25% perturbation strength compared to standard deviations of $(0.09, 0.14, 0.16)$ for the nominal optimal controller and $(0.01, 0.01, 0.01)$ for the reinforcement learning agent). The end result is, by allowing the optimal controller access to more distribution-level data, it is capable of customizing schedules in a way that is more robust to perturbations in the patient specific parameters. However, for sufficiently high perturbations in the strength of the parameters, these personalised schedules are still less optimal than the personalised schedules produced by the reinforcement learning agent. Again, we note that the success of the reinforcement learning agent is due to the increased diffusivity of the distribution of scores obtained by the NTNOC as perturbation strength increases as contrasted with the more stable distribution of reinforcement learning agent derived scores under the same perturbation strengths.



(a) At 15% perturbation strength the reinforcement agent produces a schedule that produces median scaled scores of 0.991 while the optimal controller derived schedule produces median scaled scores of 0.995



(b) At 20% perturbation strength the reinforcement agent produces a schedule that produces median scaled scores of 0.992 while the optimal controller derived schedule produces median scaled scores of 0.916.



(c) At 25% perturbation strength the reinforcement agent produces a schedule that produces median scaled scores of 0.991 while the optimal controller derived schedule produces median scaled scores of 0.986.

Figure 5.10: Histograms of the scores achieved by the various agents on the 200 testing virtual patients with a 0.025 bin size. While the reinforcement learning agent is more robust towards perturbation in parameters at the 20% and 25% perturbation level, the NTNOC produces schedules within 5% of the theoretical maximum at the 15% perturbation level.

5.4 Discussion

In summary, we examined a model of breast cancer, ovarian cancer, and bone marrow density under treatment by a chemotherapeutic for which the continuous time optimal control has been analytically derived. We discretised the optimal control problem of chemotherapeutic dosing schedule under the objective functional in (5.2) to apply different doses every day with dose strength discretised to be in 0 to 1 inclusive by steps of size 0.1. By solving this discretised problem on 200 testing virtual patients, we were able to establish ground truth levels for theoretical maximal objective functional scores. We then contrasted a reinforcement learning agent trained on the nominal parameter set with a traditional optimal controller on the nominal parameter set. We noted that since the reinforcement learning agent trains a fixed policy, it can customize the corresponding dose schedule to testing virtual patients, even when the patient-specific parameterisation of the differential equation environment from (5.1) is unknown, by leveraging additional data that is easier in practice to collect. In this case, this meant providing the reinforcement learning agent with a window of relative bone marrow density mass (relative to before the treatment process began). We noted that the reinforcement learning agent produces schedules that are closer to the theoretical optimum in a mean sense when measured against unknown patients who differ from the nominal parameter set by 15%, 20%, and 25%. In particular, we note that as the strength of perturbation increases, the net benefit of using the reinforcement learning derived schedules also increases. Moreover, as the perturbation strength increases, the collection of scaled optimality scores stay clustered between 0.925 and 1. In contrast, as the perturbation strength increases, the collection of scaled optimality scores for the optimal controller become more diffuse. The fact that the optimal control problem can be uniquely solved (due to the convexity in (5.2)) is what allows us to scale these scores. The fact that these scores stay so tightly clustered around 1 validates that this method is producing near-optimal treatments even in the absence of patient specific parameters.

We also allowed the optimal controller derived schedules to leverage the longitudinal relative bone marrow density information by training 1000 such optimal controllers on perturbed parameter values that were treated as known. When we compared this nearest training neighbour agent to the reinforcement learning agent, we discovered that the reinforcement learning agent still outperformed the other agent at the 20% and 25% perturbation strength level, but at the 15% perturbation strength level the nearest training neighbour agent was more optimal. However, this nearest training neighbour optimal controller was still prone to reduced performance level and a more diffuse histogram of dose-schedule scores at the higher perturbation levels, something that we did not observe in the reinforcement learning agent.

We conclude by noting that reinforcement learning provides an agent that can be used to personalize dosage schedules in the absence of patient specific parameter data in a manner that is not prone to wild fluctuations (as evidenced by the tight histograms of Figure 5.8 and Figure 5.10) and did so by only requiring the mean values of the patient specific parameter distributions. In contrast, an optimal control derived agent could be improved to allow personalisation of dosing schedule as well, but at the cost of requiring more samples from these patient specific distributions and the end result was still prone to fluctuations in schedule optimality.

While this particular study was focused on a situation where little patient data was used (outside of the data used to determine the nominal parameters from Table 5.1) one could also extend this work by allowing a reinforcement learner to learn directly from patient data, since the environment a reinforcement learning agent interacts with is effectively a black box. This approach might still require an underlying mathematical model in order to predict the effects when an agent chooses an action that results in a tumour state outside of the training data. This model could be a validated differential equation model or, if data is ample enough, a simple interpolation model could be employed. Moreover if a reinforcement learning agent has been trained on a model-driven environment, then that Q network could be used for transfer learning to the real data in order to speed up convergence and increase accuracy. This method described is general and can be used for optimising schedules for other treatments as well (i.e. radiotherapy fractions or immunotherapy treatment schedules). Moreover, this study was focused on a particular model in a mathematical oncology context, however we believe this result can be applied to other mathematical models used in cancer research and can also be extended easily to other mathematical biology contexts, or into any context wherein one needs to create a control for a system where high level distribution information about the parameters is known, but particular parameter values are unknown or prohibitively difficult to ascertain.

Chapter 6

Conclusion

In this thesis we used mathematical theory combined with *in silico* experiments to investigate various aspects of mathematical oncology especially as it relates to cancer stem cells and inter-patient variability.

In Chapter 3 we investigated cancer stem cells and the confusing role of phenotypic plasticity on the fixation probability of mutant cancer stem cells. In particular, we demonstrated that phenotypic plasticity produces a non-monotone effect on the invasion potential of mutant cancer stem cells. CSCs play a key role in the invasion of various cancers in an otherwise healthy tissue micro-environment. Cells that select for a greater degree of phenotypic plasticity, then, are not necessarily more invasive. The non-monotonicity of the response functions demonstrates that blindly increasing plasticity can result in cells that are less likely to fixate in an otherwise stable environment.

Initially, cellular plasticity is important and increasing the plasticity results in increased fixation probability. In regimes where the plasticity is expected to be low, then local increases result in increased fixation probability, however as the plasticity increases cells are more likely to de-differentiate. This results in less selective pressure in the differentiated cells compartment resulting in a decrease in fixation probability. However, when de-differentiation is governed by transit amplifying cells, varying the plasticity rate results in a strictly monotone increase in fixation probability due to the removal of competition between differentiated cells.

Moreover, in Chapter 3 we demonstrated that the two models considered were able to produce differing qualitative effects. In particular, the seeming contradiction observed between the Moran and Gillespie models was reconciled by an appropriate change of parameters. However, these differences highlight the effect that particular assumptions in

the model selection process have on the potential qualitative behaviour of the model. In particular, when the method by which de-differentiation is modeled gives rise to competition between compartments, then non-monotone response curves were observed. However, when de-differentiation was investigated absent of any competition (either by taking $\eta_1 = \eta_2 = 1$ and varying $\tilde{r}_1$ and $\tilde{r}_2$ instead or by modeling de-differentiation via transit amplifying cells), then the response curves were strictly monotone.

To expand upon the work of Chapter 3 it would be interesting to investigate the role of spatiality on the fixation probability. That is, it would be interesting to investigate in models of mutant stem cell invasion if the inclusion of spatiality effect the monotonicity of the fixation probability curve. Finally, it was observed that including an extra heterogeneous compartment between stem and regular cells (in the form of transit amplifying cells) changed the monotonicity of the fixation probability curve, if this heterogeneity/pluripotency of the cell type was instead considered as an additional continuous, dimension then the system could be thought of as dynamic in both time and pluripotency. It would be interesting to see if the monotonicity of the fixation probability curve remains in such a partial differential equation model.

In Chapter 4 we explored the cell cycle of Jurkat cells and demonstrated a technique capable of predicting the mitotic phase of the cell cycle with a high degree of accuracy. By examining this technique, and the effects of various refinements, we develop a framework capable of producing high throughput cell binning based on the mitotic phase without the need of various biomarkers or additional experimental agents. We also demonstrated that even in this niche regime, automatic feature selection outperforms the biologist-informed tabular data.

For a future addition to Chapter 4 we could consider applying the approach to other datasets from the imaging flow cytometer. For instance, determining if the the distinction between stem and differentiated cells or between drug resistant and drug sensitive cells can be determined via imaging alone.

In Chapter 5 we produce a method to schedule chemotherapeutic delivery in patients with breast or ovarian cancer. Importantly, while the model requires a nominal patient during training, we demonstrate that it is incredibly resilient to perturbations in patient-specific parameter values. In effect, by training the model on representative virtual patients, we can apply the therapy in an adaptive manner to patients without needing to calibrate a patient specific model. Moreover, the approach does not require a specific tumour growth inhibition model to be employed. In effect, this means that the model can be tuned directly from data. This intersection of data driven models with theory-informed models is incredibly attractive for the deployment of patient specific therapies, especially as patient specific data

becomes more easily accessible. Finally, we compared this approach with various models using classical optimal control and noted that the reinforcement learning based approach is robust to inter-patient variability.

As a future work, the model of Chapter 5 could be extended to a more complicated model of spatial optimisation of radiation beams (such as in [70]). In such a case, we would be optimising not only the schedule of delivery but also the shape of the beam. Moreover in cases of chemotherapy delivery, it would be interesting to investigate the effect of updating the RL environment according to multiple governing models on the generation of chemotherapeutic delivery schedules. For example by employing an agent based model of chemotherapeutic delivery to update the state of the tumour site at one time scale immediately after treatment and then employing a coarse grained, ODE based model initialised by the state of the agent based model for the larger time scale between treatments.

In each of the chapters we employ stochastic models of cancer at various levels: tumour invasion, cell cycle phase identification, and tumour eradication. We also consider the effects of heterogeneity of cell types, heterogeneity of data, and heterogeneity of patients in these applications. By employing mathematical analysis and *in silico* simulations to describe the biological phenomena, we were able to investigate the mechanisms at play in the biology and identify additional areas for investigation. In Chapter 3, the stochastic process was completely model driven, in Chapter 4 the process was completely data driven, and in Chapter 5 the process was model driven (via the underlying tumour growth inhibition model) but data informed (via the relative measurements of bone marrow density).

References

[1] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017.

[2] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al. Benchmarking tinyml systems: Challenges and direction. arXiv preprint arXiv:2003.04821, 2020.

[3] Logan DR Beal, Daniel C Hill, R Abraham Martin, and John D Hedengren. Gekko optimization suite. Processes, 6(8):106, 2018.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, 2013.

[5] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research, 290(2):405–421, 2021.

[6] Julius Berner, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. The modern mathematics of deep learning. CoRR, abs/2105.04026, 2021.

[7] Jesse Berwald, Tomáš Gedeon, and John Sheppard. Using machine learning to predict catastrophes in dynamical systems. Journal of Computational and Applied Mathematics, 236(9):2235–2245, 2012.

[8] Thomas Blasi, Holger Hennig, Huw D Summers, Fabian J Theis, Joana Cerveira, James O Patterson, Derek Davies, Andrew Filby, Anne E Carpenter, and Paul Rees. Label-free cell cycle analysis for high-throughput imaging flow cytometry. Nature communications, 7(1):1–9, 2016.

[9] Tijana Borovski, E Melo Felipe De Sousa, Louis Vermeulen, and Jan Paul Medema. Cancer stem cell niche: the place to be. Cancer research, 71(3):634–639, 2011.

[10] Pratik Prabhanjan Brahma, Dapeng Wu, and Yiyuan She. Why deep learning works: A manifold disentanglement perspective. IEEE transactions on neural networks and learning systems, 27(10):1997–2008, 2015.

[11] Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[12] Michael Brown and Carl Wittwer. Flow cytometry: principles and clinical applications in hematology. Clinical chemistry, 46(8):1221–1229, 2000.

[13] Marina Carla Cabrera, Robert E Hollingsworth, and Elaine M Hurt. Cancer stem cell plasticity and tumor hierarchy. World journal of stem cells, 7(1):27, 2015.

[14] Anne E Carpenter, Thouis R Jones, Michael R Lamprecht, Colin Clarke, In Han Kang, Ola Friman, David A Guertin, Joo Han Chang, Robert A Lindquist, Jason Moffat, et al. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. Genome biology, 7(10):1–11, 2006.

[15] Christine L Chaffer, Ines Brueckmann, Christina Scheel, Alicia J Kaestli, Paul A Wiggins, Leonardo O Rodrigues, Mary Brooks, Ferenc Reinhardt, Ying Su, Kornelia Polyak, et al. Normal and neoplastic nonstem cells can spontaneously convert to a stem-like state. Proceedings of the National Academy of Sciences, 108(19):7950–7955, 2011.

[16] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Transactions on Neural Networks, 6(4):911–917, 1995.

[17] Keyang Cheng, Rabia Tahir, Lubamba Kasangu Eric, and Maozhen Li. An analysis of generative adversarial networks and variants for image synthesis on mnist dataset. Multimedia Tools and Applications, 79(19):13725–13752, 2020.

[18] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. Advances in Neural Information Processing Systems, 32, 2019.

[19] Francois Chollet. Deep learning with Python. Simon and Schuster, 2021.

[20] Yehuda Dar, Vidya Muthukumar, and Richard G Baraniuk. A farewell to the bias-variance tradeoff? an overview of the theory of overparameterized machine learning. arXiv preprint arXiv:2109.02355, 2021.

[21] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Tiezhen Wang, et al. Tensorflow lite micro: Embedded machine learning for tinyml systems. Proceedings of Machine Learning and Systems, 3:800–811, 2021.

[22] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In International Conference on Machine Learning, pages 1019–1028. PMLR, 2017.

[23] Brydon Eastman, Michelle Przedborski, and Mohammad Kohandel. Reinforcement learning derived chemotherapeutic schedules for robust patient-specific therapy. Scientific Reports, 11(1):17882, 2021.

[24] Brydon Eastman, Dominik Wodarz, and Mohammad Kohandel. The effects of phenotypic plasticity on the fixation probability of mutant cancer stem cells. Journal of Theoretical Biology, 503:110384, 2020.

[25] Leah Edelstein-Keshet. Mathematical models in biology. SIAM, 2005.

[26] Martin Eisen. Mathematical models in cell biology and cancer chemotherapy, volume 30. Springer Science & Business Media, 2013.

[27] Dalit Engelhardt. Dynamic control of stochastic evolution: A deep reinforcement learning approach to adaptively targeting emergent drug resistance. Journal of Machine Learning Research, 21(203):1–30, 2020.

[28] Wei Fan, Erheng Zhong, Jing Peng, Olivier Verscheure, Kun Zhang, Jiangtao Ren, Rong Yan, and Qiang Yang. Generalized and heuristic-free feature construction for improved accuracy. In Proceedings of the 2010 SIAM International Conference on Data Mining, pages 629–640. SIAM, 2010.

[29] Farzan Farnia and Asuman Ozdaglar. Gans may have no nash equilibria. arXiv preprint arXiv:2002.09124, 2020.

[30] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. 2018.

[31] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. Journal of the American Mathematical Society, 29(4):983–1049, 2016.

[32] Peter I Frazier. A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811, 2018.

[33] Philip Gerlee and Alexander RA Anderson. An evolutionary hybrid cellular automaton model of solid tumour growth. Journal of theoretical biology, 246(4):583–603, 2007.

[34] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. The journal of physical chemistry, 81(25):2340–2361, 1977.

[35] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. Annual Review of Physical Chemistry, 58(1):35–55, 2007. PMID: 17037977.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

[37] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[38] Piyush B Gupta, Christine M Fillmore, Guozhi Jiang, Sagi D Shapira, Kai Tao, Charlotte Kuperwasser, and Eric S Lander. Stochastic state transitions give rise to phenotypic equilibrium in populations of cancer cells. Cell, 146(4):633–644, 2011.

[39] Douglas Hanahan. Hallmarks of cancer: New dimensions. Cancer Discovery, 12(1):31–46, 2022.

[40] Douglas Hanahan and Robert A Weinberg. The hallmarks of cancer. cell, 100(1):57–70, 2000.

[41] Douglas Hanahan and Robert A Weinberg. Hallmarks of cancer: the next generation. cell, 144(5):646–674, 2011.

[42] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. The elements of statistical learning: data mining, inference, and prediction, volume 2. Springer, 2016.

[43] Simon S Haykin. Neural networks and learning machines, volume 3. Pearson education Upper Saddle River, 2009.

[44] F. Hayot and C. Jayaprakash. Nf-κb oscillations and cell-to-cell variability. Journal of Theoretical Biology, 240(4):583–591, 2006.

[45] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.

[46] J Hedengren, J Mojica, W Cole, and T Edgar. Apopt: Minlp solver for differential and algebraic systems with benchmark testing. In Proceedings of the INFORMS National Meeting, Phoenix, AZ, USA, volume 1417, page 47, 2012.

[47] John L Hodges. The significance probability of the smirnov two-sample test. Arkiv för Matematik, 3(5):469–486, 1958.

[48] DJ Huels and OJ Sansom. Stem vs non-stem cell origin of colorectal cancer. British journal of cancer, 113(1):1, 2015.

[49] James W Jacobberger, Phyllis S Frisa, R Michael Sramkoski, Tammy Stefan, Keith E Shults, and Deena V Soni. A new biomarker for mitotic cells. Cytometry Part A: The Journal of the International Society for Analytical Cytology, 73(1):5–15, 2008.

[50] Angela M Jarrett, Danial Faghihi, David A Hormuth Ii, Ernesto ABF Lima, John Virostko, George Biros, Debra Patt, and Thomas E Yankeelov. Optimal control theory for personalized therapeutic regimens in oncology: Background, history, challenges, and opportunities. Journal of clinical medicine, 9(5):1314, 2020.

[51] Alexandra Jilkine and Ryan N Gutenkunst. Effect of dedifferentiation on time to mutation acquisition in stem cell-driven cancers. PLoS computational biology, 10(3), 2014.

[52] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. CoRR, abs/1812.04948, 2018.

[53] Patrick Kidger and Terry J. Lyons. Universal approximation with deep narrow networks. CoRR, abs/1905.08539, 2019.

[54] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[55] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. CoRR, abs/1906.02691, 2019.

[56] Jack PC Kleijnen. An overview of the design and analysis of simulation experiments for sensitivity analysis. European Journal of Operational Research, 164(2):287–300, 2005.

[57] Leonid Koralov and Yakov G Sinai. Theory of probability and random processes. Springer Science & Business Media, 2007.

[58] Antonija Kreso and John E Dick. Evolution of the cancer stem cell model. Cell stem cell, 14(3):275–291, 2014.

[59] L Lai, N Suda, and V CMSIS-NN Chandra. Efficient neural network kernels for arm cortex-m cpus. CoRR abs/1801.06601, 2018.

[60] Na Lei, Dongsheng An, Yang Guo, Kehua Su, Shixia Liu, Zhongxuan Luo, Shing-Tung Yau, and Xianfeng Gu. A geometric understanding of deep learning. Engineering, 6(3):361–374, 2020.

[61] Roger J Lewis. An introduction to classification and regression tree (cart) analysis. In Annual meeting of the society for academic emergency medicine in San Francisco, California, volume 14. Citeseer, 2000.

[62] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[63] Yunqian Ma and Yun Fu. Manifold learning theory and applications, volume 434. CRC press Boca Raton, 2012.

[64] Ali Mahdipour-Shirayeh, Kamran Kaveh, Mohammad Kohandel, and Sivabal Sivaloganathan. Phenotypic heterogeneity in modeling cancer evolution. PloS one, 12(10):e0187000, 2017.

[65] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Natural Language Engineering, 16(1):100–103, 2010.

[66] Rafael Gomes Mantovani, Tomáš Horváth, Ricardo Cerri, Sylvio Barbon Junior, Joaquin Vanschoren, and André Carlos Ponce de Leon Ferreira de Carvalho. An empirical study on hyperparameter tuning of decision trees. arXiv preprint arXiv:1812.02207, 2018.

[67] Nemanja D Marjanovic, Robert A Weinberg, and Christine L Chaffer. Cell plasticity and heterogeneity in cancer. Clinical chemistry, 59(1):168–179, 2013.

[68] Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. Machine Learning, 49(1):59–98, 2002.

[69] Susan Christine Massey, Paula Whitmire, Tatum E Doyle, Joseph E Ippolito, Maciej M Mrugala, Leland S Hu, Peter Canoll, Alexander RA Anderson, Melissa A Wilson, Susan M Fitzpatrick, et al. Sex differences in health and disease: A review of biological sex differences relevant to cancer with a spotlight on glioma. Cancer letters, 498:178–187, 2021.

[70] Cameron Meaney, Gibin G Powathil, Ala Yaromina, Ludwig J Dubois, Philippe Lambin, and Mohammad Kohandel. Role of hypoxia-activated prodrugs in combination with radiation therapy: An in silico approach. Mathematical Biosciences and Engineering, 16(6):6257–6273, 2019.

[71] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[72] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[73] P. A. P. Moran. Random processes in genetics. Mathematical Proceedings of the Cambridge Philosophical Society, 54(1):60–71, 1958.

[74] Fatemeh Nargesian, Horst Samulowitz, Udayan Khurana, Elias B Khalil, and Deepak S Turaga. Learning feature engineering for classification. In Ijcai, pages 2529–2535, 2017.

[75] Peter C Nowell. The clonal evolution of tumor cell populations: Acquired genetic lability permits stepwise selection of variant sublines and underlies tumor progression. Science, 194(4260):23–28, 1976.

[76] Catherine Adell O'Brien, Antonija Kreso, and John E Dick. Cancer stem cells in solid tumors: an overview. In Seminars in radiation oncology, volume 19, pages 71–77. Elsevier, 2009.

[77] Fabian Otto. Model-free deep reinforcement learning—algorithms and applications. In Reinforcement Learning Algorithms: Analysis and Applications, pages 109–121. Springer, 2021.

[78] John Carl Panetta. A mathematical model of breast and ovarian cancer treated with paclitaxel. Mathematical biosciences, 146(2):89–113, 1997.

[79] John Carl Panetta and J Adam. A mathematical model of cycle-specific chemotherapy. Mathematical and computer modelling, 22(2):67–82, 1995.

[80] John Carl Panetta and K Renee Fister. Optimal control applied to cell-cycle-specific cancer chemotherapy. SIAM Journal on Applied Mathematics, 60(3):1059–1072, 2000.

[81] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, Computer Vision – ECCV 2020, pages 319–345, Cham, 2020. Springer International Publishing.

[82] Anna Philpott and Douglas J Winton. Lineage selection and plasticity in the intestinal crypt. Current opinion in cell biology, 31:39–45, 2014.

[83] Jan Poleszczuk and Heiko Enderling. A high-performance cellular automaton model of tumor growth with dynamically growing domains. Applied mathematics, 5(1):144, 2014.

[84] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. arXiv preprint arXiv:1711.10561, 2017.

[85] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. arXiv preprint arXiv:1711.10566, 2017.

[86] Anil V Rao. A survey of numerical methods for optimal control. Advances in the Astronautical Sciences, 135(1):497–528, 2009.

[87] Tannishtha Reya, Sean J Morrison, Michael F Clarke, and Irving L Weissman. Stem cells, cancer, and cancer stem cells. nature, 414(6859):105, 2001.

[88] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows, 2015.

[89] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. CoRR, abs/2112.10752, 2021.

[90] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. Advances in neural information processing systems, 30, 2017.

[91] Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. In International conference on machine learning, pages 4433–4441. PMLR, 2018.

[92] Robert E Schapire. Explaining adaboost. In Empirical inference, pages 37–52. Springer, 2013.

[93] F. W. Scholz and M. A. Stephens. K-sample anderson–darling tests. Journal of the American Statistical Association, 82(399):918–924, 1987.

[94] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 40(1):185–197, 2009.

[95] Jerry W Shay and Woodring E Wright. Hayflick, his limit, and cellular ageing. Nature reviews Molecular cell biology, 1(1):72–76, 2000.

[96] Sonia Singh and Priyanka Gupta. Comparative study id3, cart and c4. 5 decision tree algorithm: a survey. International Journal of Advanced Information Science and Technology (IJAIST), 27(27):97–103, 2014.

[97] Kathleen Sprouffske, C Athena Aktipis, Jerald P Radich, Martin Carroll, Aurora M Nedelcu, and Carlo C Maley. An evolutionary explanation for the presence of cancer nonstem cells in neoplasms. Evolutionary applications, 6(1):92–101, 2013.

[98] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. arXiv preprint arXiv:1505.00387, 2015.

[99] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[100] Purushothama Rao Tata, Hongmei Mou, Ana Pardo-Saganta, Rui Zhao, Mythili Prabhu, Brandon M Law, Vladimir Vinarsky, Josalyn L Cho, Sylvie Breton, Amar Sahay, et al. Dedifferentiation of committed epithelial cells into stem cells in vivo. Nature, 503(7475):218–223, 2013.

[101] Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58–68, 1995.

[102] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.

[103] Luca Venturi, Afonso S Bandeira, and Joan Bruna. Spurious valleys in one-hidden-layer neural network optimization landscapes. Journal of Machine Learning Research, 20:133, 2019.

[104] Jane E Visvader and Geoffrey J Lindeman. Cancer stem cells in solid tumours: accumulating evidence and unresolved questions. Nature reviews cancer, 8(10):755–768, 2008.

[105] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming, 106(1):25–57, 2006.

[106] Fiona M Watt and Brigid LM Hogan. Out of eden: stem cells and their niches. Science, 287(5457):1427–1430, 2000.

[107] Irving L Weissman. Stem cells. cell, 100(1):157–168, 2000.

[108] Frank Wilcoxon. Individual comparisons by ranking methods. Biometrics Bulletin, 1(6):80–83, 1945.

[109] Dominik Wodarz. Effect of cellular de-differentiation on the dynamics and evolution of tissue and tumor cells in mathematical models with feedback regulation. Journal of theoretical biology, 448:86–93, 2018.

[110] Krzysztof Wojcik and Jurek W Dobrucki. Interaction of a dna intercalator draq5, and a minor groove binder syto17, with chromatin in live cells—influence on chromatin organization and histone—dna interactions. Cytometry Part A: the journal of the International Society for Analytical Cytology, 73(6):555–562, 2008.

[111] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. arXiv preprint arXiv:2203.05482, 2022.

[112] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.

[113] Gregory Yauney and Pratik Shah. Reinforcement learning with action-derived rewards for chemotherapy and clinical trial dosing regimen selection. In Machine Learning for Healthcare Conference, pages 161–226, 2018.

[114] Stefano Zapperi and Caterina AM La Porta. Do cancer cells undergo phenotypic switching? the case for imperfect cancer stem cell markers. Scientific reports, 2(1):1–7, 2012.

[115] Assaf Zaritsky, Andrew R. Jamieson, Erik S. Welf, Andres Nevarez, Justin Cillay, Ugur Eskiocak, Brandi L. Cantarel, and Gaudenz Danuser. Interpretable deep learning of label-free live cell images uncovers functional hallmarks of highly-metastatic melanoma. bioRxiv, 2020.

[116] Cha Zhang and Yunqian Ma. Ensemble machine learning: methods and applications. Springer, 2012.

[117] Da Zhou, Yue Luo, David Dingli, and Arne Traulsen. The invasion of de-differentiating cancer cells into hierarchical tissues. PLoS computational biology, 15(7):e1007167, 2019.

[118] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.

APPENDICES

Appendix A

Numerics for Stochastic Simulations of Cancer Stem Cells

A.1 Numerics

The Moran simulations were run until fixation occurred or until 15,000 iterations were achieved – whichever came first. A total of 500,000 such simulations were run in 50 batches of 10,000. For each batch of 10,000 simulations, fixation probability was estimated by logging what percentage of the 10,000 iterations achieved fixation. The error of these probabilities was estimated as the standard error of the mean calculated between the 50 different batches. This process was completed once for every unique set of parameter values.

Similarly, the two Gillespie simulations (in Figures 3.2 and 3.3) were run until fixation of or until 10^8 iterations were achieved. For each parameter set, these simulations were run in 10 batches of 3000 simulations with the fixation probability being calculated for each individual batch of 3000 simulations. The error was calculated from these 10 different fixation probabilities as the standard error of the mean.

A.1.1 Figure Information

In Figure 3.2a the plot was generated by Moran simulations with the following parameters: $N_S = N_D = 10$, $u_1 = u_2 = 0.5$, $r_1 = r_2 = \tilde{r}_1 = \tilde{r}_2 = 1$, $\eta_1 = 0$, $d_1 = d_2 = \tilde{d}_1 = \tilde{d}_2 = 1$, and η_2 varying over 36 discrete values evenly placed between 0 and 1, inclusive.

Similarly, for Figure 3.2b was generated by Gillespie simulations (as in c[109]) with the following parameters: $r_1' = r_2' = 0.5$, $p_1' = p_2' = 0.8$, $h_{1,1} = h_{2,1} = h_{1,2} = h_{2,2} = h_{1,3} = h_{2,3} = 0.01$, $k_1 = k_2 = k_3 = 1$, $\alpha_1 = \alpha_2 = 0.5$, and $g_1' = g_2' = g$. The initial condition for these simulations was $(S_1, D_1, S_2, D_2) = (S_1^*, D_1^*, 1, 0)$ where S_1^* and D_1^* are defined as (within rounding to nearest integer) the equilibrium point of the deterministic differential equations in S_1 and D_1 (where S_2 and D_2 are assumed to be zero for the purpose of obtaining the equilibrium values) and vary depending on the value of g. The stochastic simulation was then repeated for each g value in $\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$ corresponding to initial conditions $(S_1^*, D_1^*) \in \{(114, 68), (132, 75), (101, 62), (166, 88), (183, 94), (200, 100)\}$. Note that while the Moran simulations in Figure c3.2a use finite, discrete population sizes, the simulations in Figure c3.2b make no such assumptions, allowing continuous, unbounded population sizes.

In Figure 3.3 the leftmost plot was generated by Moran simulations with the following parameters: $N_S = N_D = 10$, $u_1 = u_2 = 0.5$, $r_1 = r_2 = \tilde{r}_1 = \tilde{r}_2 = 1$, $d_1 = d_2 = \tilde{d}_1 = \tilde{d}_2 = 1$, and $\eta_1 = \eta_2 = \eta$ where η varies over 36 discrete values evenly placed between 0 and 1, inclusive.

Similarly, the rightmost plot was generated by Gillespie simulations with the following parameters:$r_1' = r_2' = 0.5$, $p_1' = p_2' = 0.8$, $h_{1,1} = h_{2,1} = h_{1,2} = h_{2,2} = h_{1,3} = h_{2,3} = 0.01$, $k_1 = k_2 = k_3 = 1$, $\alpha_1 = \alpha_2 = 0.5$, $g_1' = 0$, and g_2' taking values in $\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. As in Figure 3.2, the initial condition was taken to be $(S_1, D_1, S_2, D_2) = (S_1^*, D_1^*, 1, 0)$ with S_1^* and D_1^* defined as the equilibria points of the (S_1, D_1) system calculated similarly to the above.

In Figure 3.4 the data was generated with Moran simulations with the following parameters: $N_S = N_D = 10$, $u_1 = u_2 = 0.5$, $r_1 = r_2 = \tilde{r}_1 = \tilde{r}_2 = 1$, $d_1 = d_2 = \tilde{d}_1 = \tilde{d}_2 = 1$, and both η_1 and η_2 vary, independently, from 0 to 1 in 0.1 increments. A cubic spline was generated from the resulting data points (and their standard errors) and used to create the plots in Figures 3.5 and 3.6.

In Figure 3.7 the data was generated, for each fixed η_2 value, by averaging over all 36 η_1 values.

Figures 3.8 and 3.9 were generated similarly. In both figures Moran simulations were run with parameters $N_S = N_D = 10$, $d_1 = d_2 = \tilde{d}_1 = \tilde{d}_2 = 1$, and η_1 and η_2 independently varying over 36 discrete values evenly placed between 0 and 1, inclusive. In Figure 3.8 $r_2 = \tilde{r}_2 = r$ where r takes on values in $\{0.25, 0.50, 0.75, 1, 2, 3, 4, 5\}$ while $u_1 = u_2 = 0.5$. Similarly, in Figure 3.9 u_2 was allowed to vary between 0.1 and 0.9 inclusive in increments of 0.1, while $r_2 = \tilde{r}_2 = 1$ and $u_1 = 0.5$ were kept constant. The averages were calculated from the resulting data points as in Figure 3.7.

Figures 3.11 and 3.12 were generated with Moran simulations run with $N_S = N_D = 10$ where $\eta_1 = \eta_2 = 1$ and $\tilde{r}_1$ and $\tilde{r}_2$ were varied over 36 discrete values evenly placed between 0 and 1. For Figure 3.11, the values of $\tilde{r}_1$ and $\tilde{r}_2$ were varied independently, for Figure 3.12 they were varied together. The data were generated by averaging the results of 100 simulations 5000 times. The remaining parameters were set as $r_1 = r_2 = 1$, $u_1 = u_2 = 0.5$.

Appendix B

Additional Details to Chapter 5

In Figure B.1 we present a visualisation of the virtual patients used for testing of all three algorithms and the training virtual patients used for training of the NTNOC algorithm. These figures demonstrate the parameter values chosen via Latin hypercube sampling for the four non-zero bone marrow parameters from Table 5.1 required to parameterise (5.2).

In Figure B.2 we present the trajectories of the bone-marrow and the control obtained via the objective functional in (5.3).

In Figure B.3 we produce a plot similar to Figure 5.3 with the proliferative fraction of both the healthy bone-marrow cells and the proliferative fraction of the malignant breast-cancer cells. For this plot, the bone marrow compartments and breast cancer compartments are parameterised with the nominal parameters from Table 5.1. Notably, in all three scenarios the schedule is able to drive the proliferative fraction of malignant cells down while maintaining the proliferative fraction of the healthy cells. In Figure B.3 $\rho_p^{\mathrm{bm}}(t)$ corresponds to the proliferative fraction of the bone-marrow cells and $\rho_p^{\mathrm{bc}}(t)$ corresponds to the proliferative fraction of the breast-cancer cells.

Figure B.4 demonstrates various dosing schedules on testing patients. We plot the schedules obtained via employing the reinforcement learning agent trained on the nominal parameter set, the mean optimal controller derived treatment, the NTNOC treatment, and the optimal treatment (which was treated as unknown) acquired by solving the optimal control problem on each particular testing patient (achieved by treating the model parameters as known and employing the APOPT algorithm as implemented in GEKKO [46, 3]).

In Figure B.5 we present plots of chemotherapy schedules along with the trajectories of

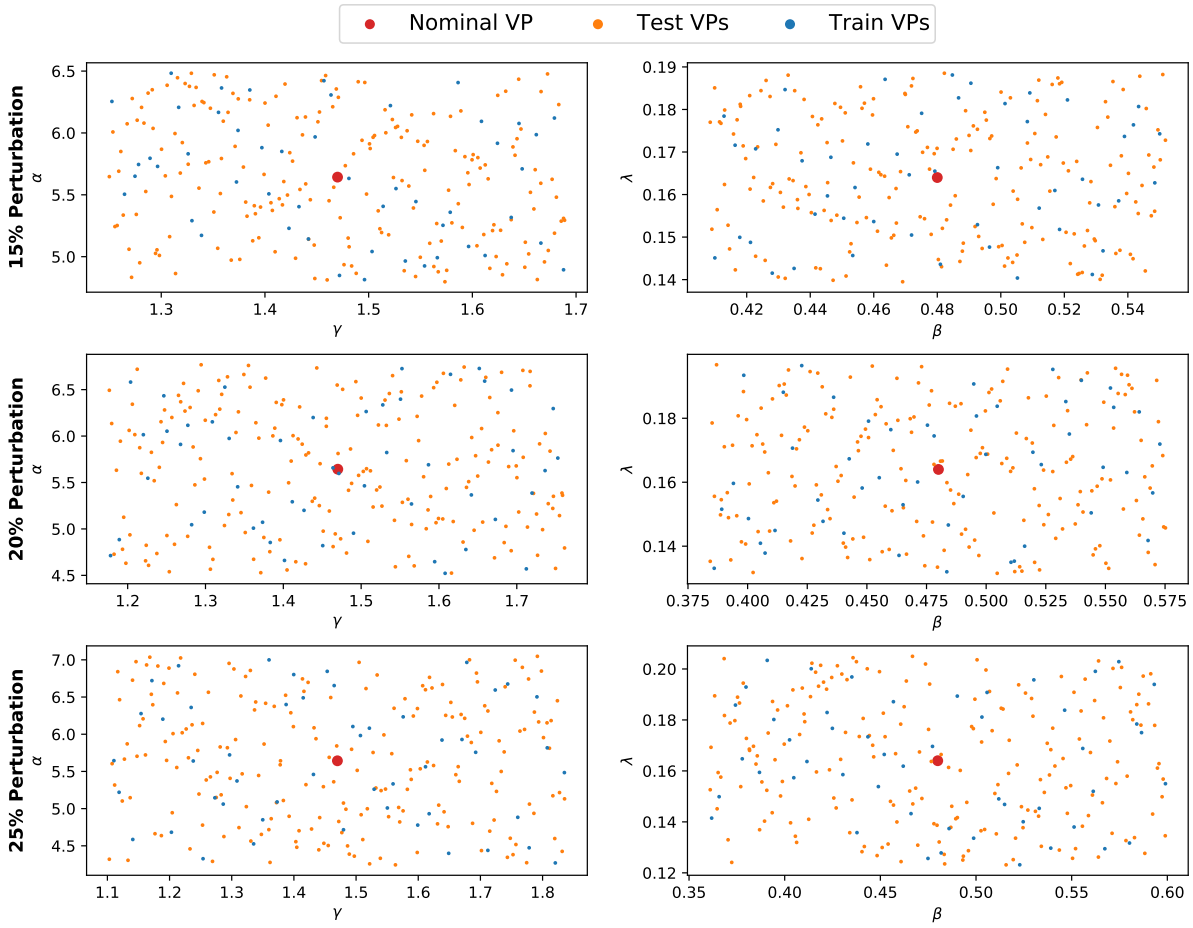


Figure B.1: Visualisation of the 4 non-zero virtual patient parameters for various perturbation strengths. The orange dots indicate the 200 virtual patients used in testing, the blue dots the 1000virtual patients used in training, and the red dot indicates the location of the nominal virtual patient.

the effected bone marrow cells. These plots are generated for the same virtual patients as in the first column of Figure B.4.

The results presented in Figures 5.7 – 5.10 all depend upon the particular sample of testing patients (the values of θ_i^k for all k and i). In order to investigate how these results change with differing batches of testing virtual patients, we now present the results of Figure 5.7 and Figure 5.9 for five additional batches of 200 testing virtual patients. Explicitly, we independently sampled five batches of 200 testing virtual patients in addition

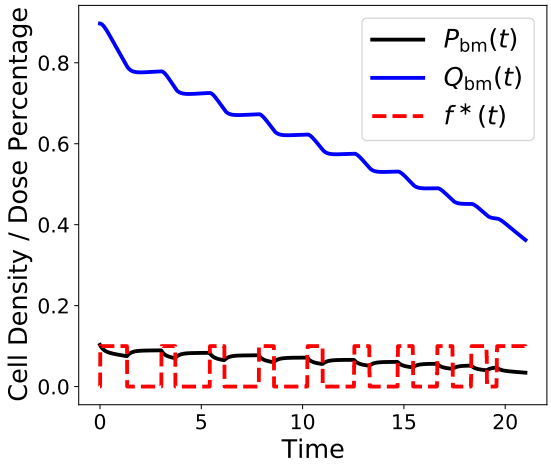
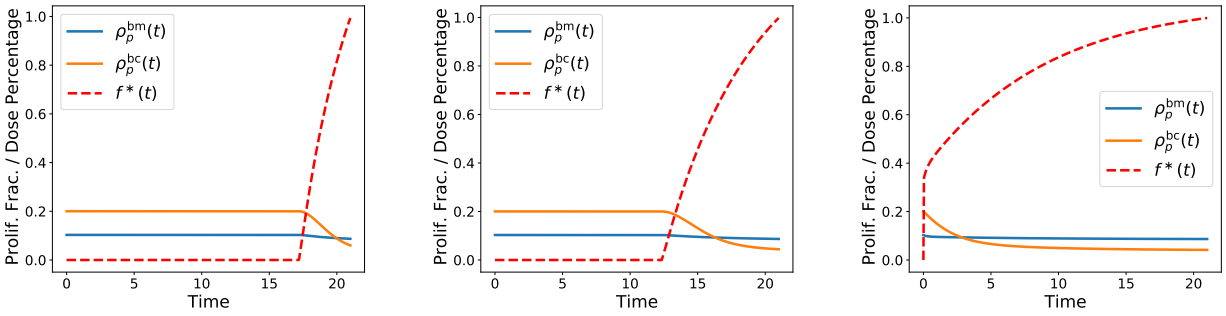


Figure B.2: A plot of the (5.1) parameterised for bone-marrow under the optimal control achieved by maximizing the objective functional in (5.3) with $b = 2$. Solved numerically using IPOPT in GEKKO [105, 3].



(a) $b = 0.5$ (b) $b = 1$ (c) $b = 2$

Figure B.3: A plot of the proliferative fraction $\rho_p(t) = P(t)/(P(t)+Q(t))$ for both the healthy bone-marrow cells and the malignant breast-cancer cells under the optimal chemotherapeutic control $f^*(t)$ (dashed red) for different values of b. The objective functional used to achieve this optimal control is given via (5.2). Small values of b correspond to weighting preservation of the bone marrow as more important and larger values of b correspond to weighting total drug delivery as more important.

to the batch used in the main manuscript text using the same method as described in Section 5.2.3. For ease of notation, let $B_0^k = \{\theta_i^k \mid 1 \leq i \leq 200\}$ denote the original batch of virtual patients used in the manuscript text at perturbation levels $k \in \{0.15, 0.20, 0.25\}$. Let $B_1^k, B_2^k, \ldots, B_5^k$ represent the 5 additional batches of testing patients sampled independently.

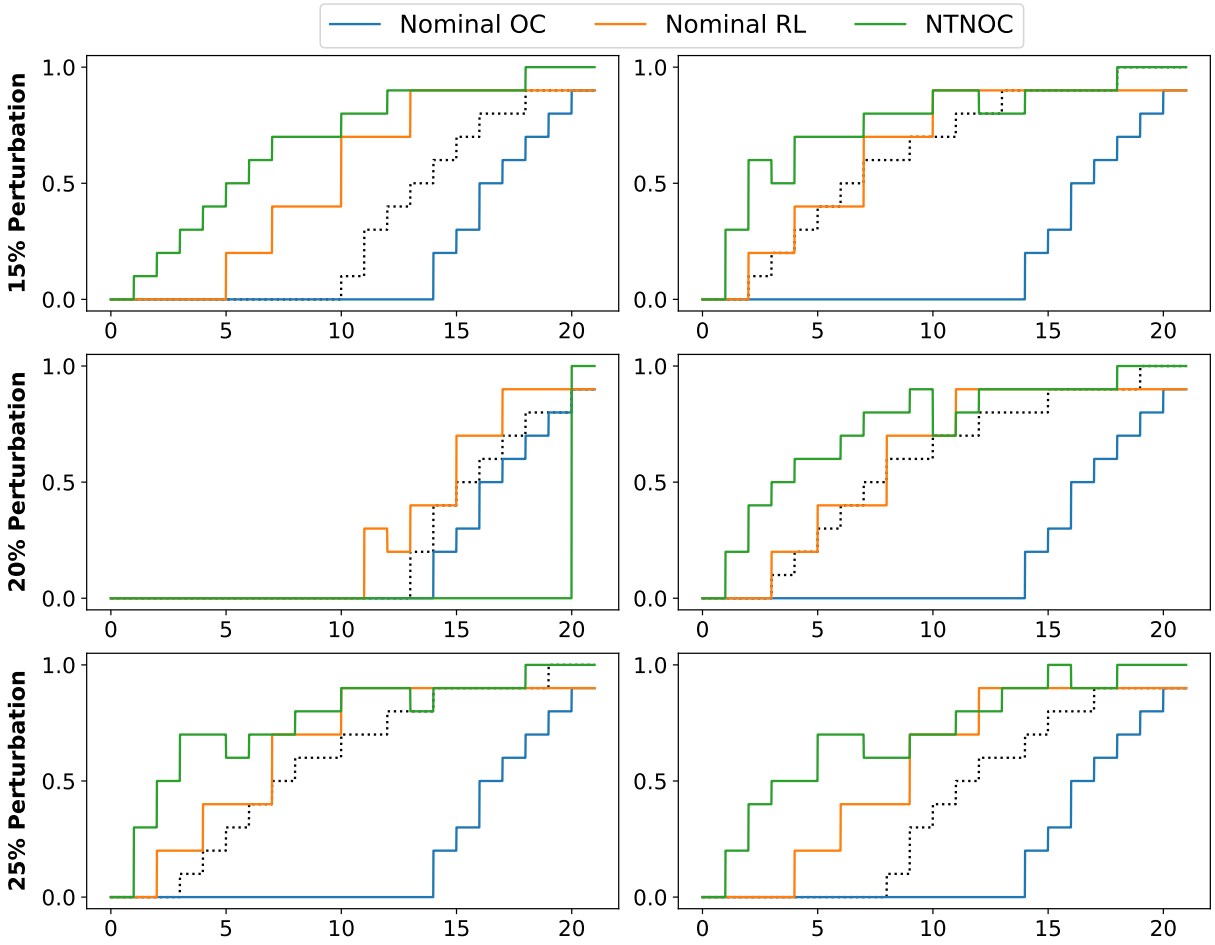


Figure B.4: Various chemotherapy dosing schedules obtained on various virtual patients by the three blind methods. In dotted black lines, we also present the theoretically optimal schedule for each patient as a comparison. While the nominal OC schedules remain static (blue lines), the RL and NTNOC schedules adapt to each patient (orange and green lines). In fact, the tendency for the RL schedules (orange lines) to be "closer" (visually) to the theoretical maximal value (dotted lines) visually can demonstrate the tendency for the RL agent to be more robust to perturbations in the parameter values.

In Figure B.6a we present Figure 5.7b as a probability distribution. Next, in Figures B.6b we present the same histogram as displayed as displayed in Figure B.6a except $B_1^{0.20}$ was used as the testing set in place of $B_0^{0.20}$. Histograms created with batches $B_2^{0.20}$ to $B_5^{0.20}$ are presented in Figures B.6c–B.6f. We note the similarities in shape between the histograms

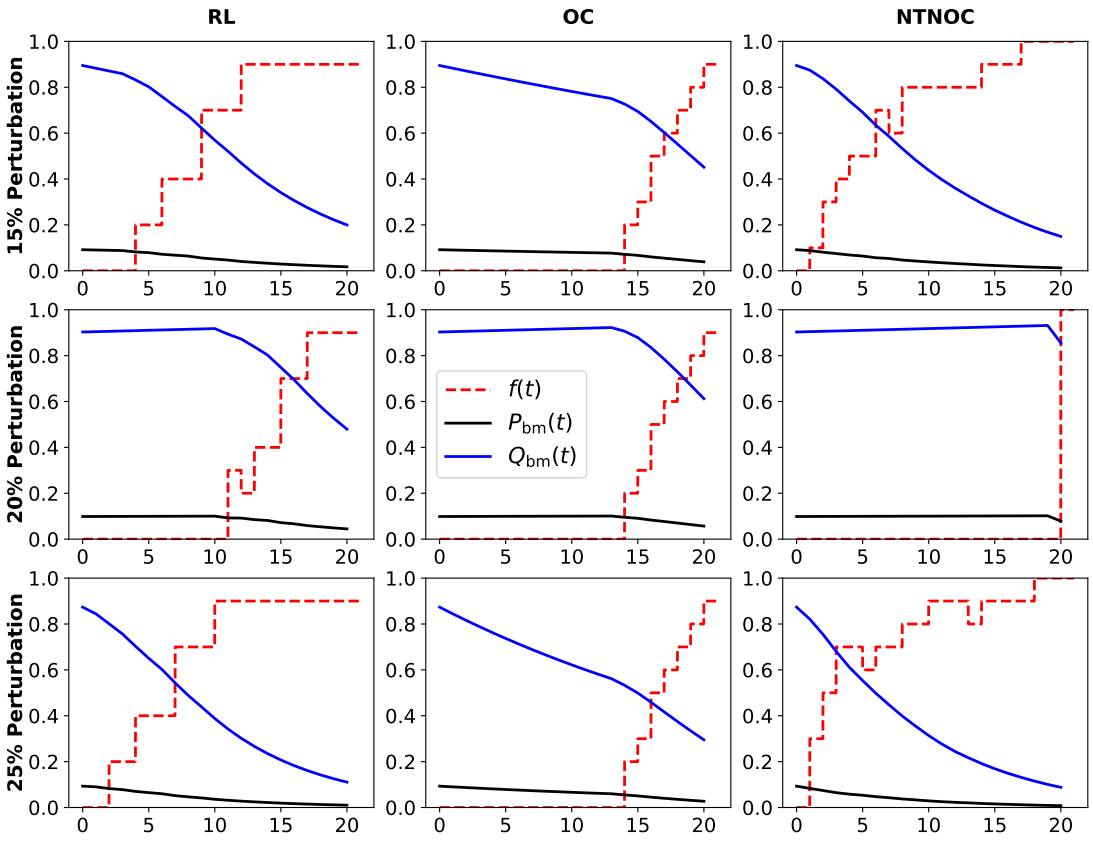
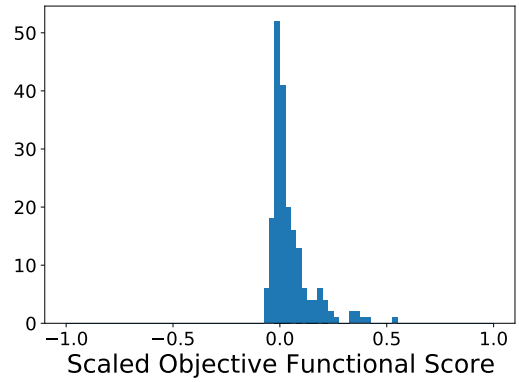


Figure B.5: Plots of the proliferative and quiescent bone marrow cells in various virtual patients under chemotherapeutic schedules acquired by the three different learning agents considered in this work.

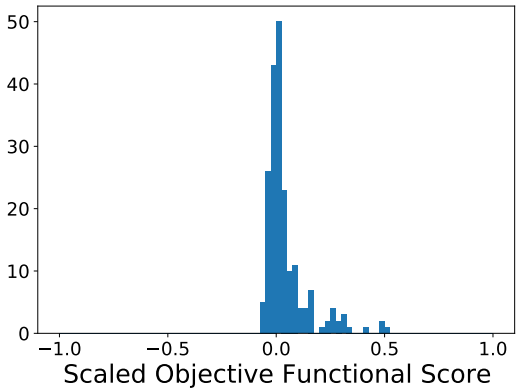
presented. In fact, we performed two statistical tests to demonstrate the similarity of the histograms produced by batch $B_0^{0.20}$ with those produced by batches $B_1^{0.20}$ to $B_5^{0.20}$. Explicitly, we considered a two sample Kolmogorov-Smirnov test and a two sample Anderson-Darling test [47, 93]. Both tests have the null hypothesis that the two samples are from the same distribution. In all five cases we are unable to reject the null hypothesis ($p > 0.25$).

Similarly, in Figure B.7 we present a related visualisation. In Figures B.7a–B.7c we present the results of comparing the reinforcement learner derived treatments from the nominal optimal controller derived treatments on all 6 batches B_0^k to B_5^k at all perturbation strengths. The data for batch B_0^k are presented as solid lines while the data for batches B_1^k to B_5^k are presented as semi-opaque filled histograms stacked on top of one another.

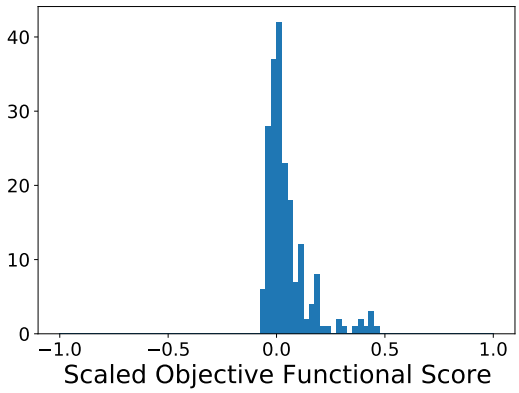
This allows one to easily compare the relative shapes of the distribution of these data. In Figures B.7d–B.7f we present similar data for comparing the reinforcement learner derived treatments with those from the NTNOC derived treatments in the same manner. In all 6 cases, we note that the histograms appear similar to one another via the naked eye. We again applied a two sample Kolmogorov-Smirnov test and a two-sample Anderson-Darling test to formally compare the samples. For all 6 experiments we compared the five new testing batches B_1^k to B_5^k with the reference testing batch B_0^k under the null hypothesis that the two samples are from the same distribution. In all 30 of these such cases we were unable to reject the null hypothesis with either test ($p > 0.25$).



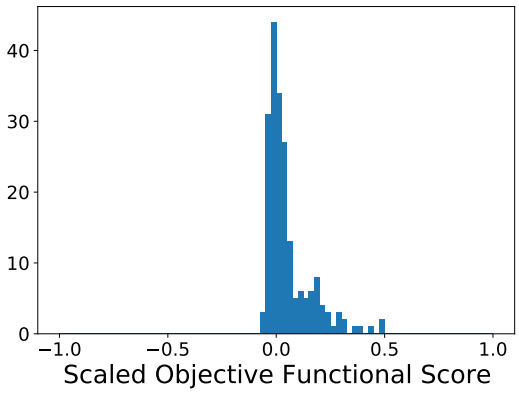
(a) Testing virtual patient batch $B_0^{0.20}$.



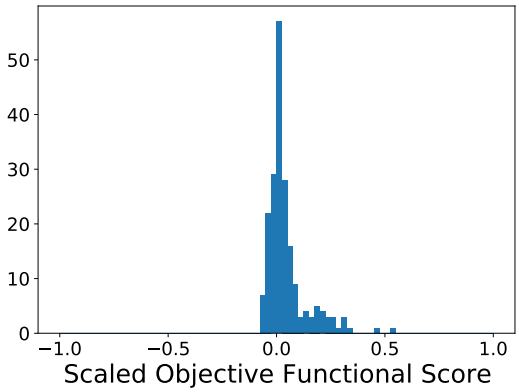
(b) Testing virtual patient batch $B_1^{0.20}$.



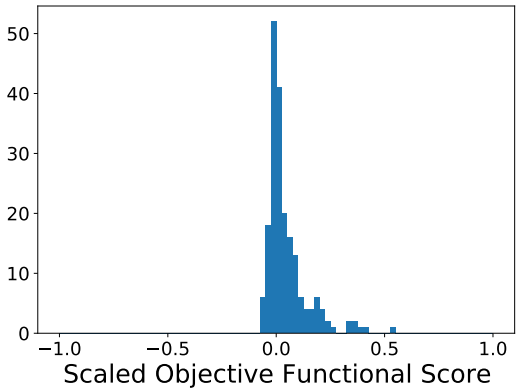
(c) Testing virtual patient batch $B_2^{0.20}$.



(d) Testing virtual patient batch $B_3^{0.20}$.

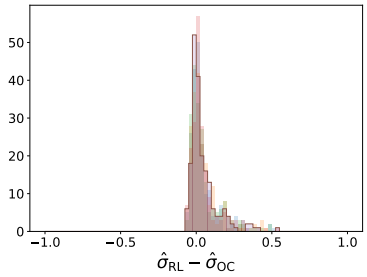


(e) Testing virtual patient batch $B_4^{0.20}$.

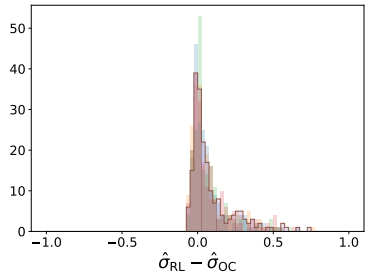


(f) Testing virtual patient batch $B_5^{0.20}$.

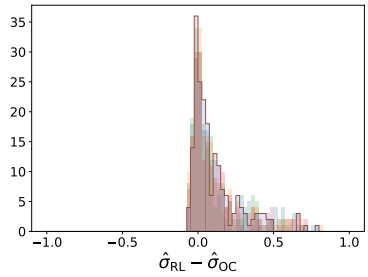
Figure B.6: Histograms of $\hat{\sigma}_{\mathrm{RL}}(\theta_i^{0.20};\xi_0) - \hat{\sigma}_{\mathrm{OC}}(\theta_i^{0.20};\xi_0)$ where the testing patients θ_i^{20} are from batches $B_0^{0.20}$ to $B_5^{0.20}$.



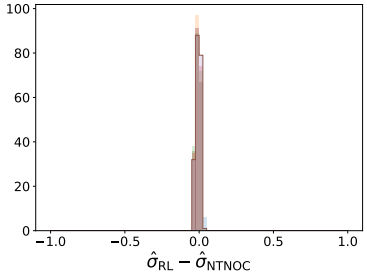
(a) 15% perturbation strength.



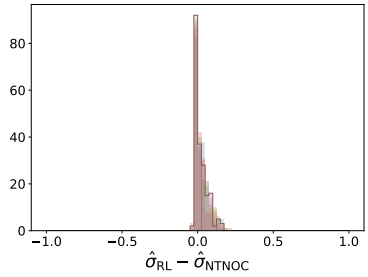
(b) 20% perturbation strength.



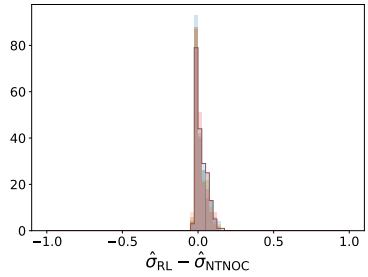
(c) 25% perturbation strength.



(d) 15% perturbation strength.



(e) 20% perturbation strength.



(f) 25% perturbation strength.

Figure B.7: A plot comparing the histograms of the difference in scaled objective functional scores. Figures B.7a – B.7c show histograms of $\hat{\sigma}_{\mathrm{RL}} - \hat{\sigma}_{\mathrm{OC}}$ for various batches while Figures B.7d – B.7f show histograms of $\hat{\sigma}_{\mathrm{RL}} - \hat{\sigma}_{\mathrm{NTNOC}}$ for various batches. In all plots the histogram of batch B_0^k is represented by the solid outlines and the histograms of batches B_1^k to B_5^k are represented by the semi-opaque filled bars. Neither the Kolmogorov-Smirnov 2-sample test nor the Anderson-Darling 2-sample test could distinguish B_0^k from any of B_1^k to B_5^k.