Modeling Protein Secondary Structure by Products of Dependent Experts

by

Christian Anders Cumbaa

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2001

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

A phenomenon as complex as protein folding requires a complex model to approximate it. This thesis presents a bottom-up approach for building complex probabilistic models of protein secondary structure by incorporating the multiple information sources which we call *experts*.

Expert opinions are represented by probability distributions over the set of possible structures. Bayesian treatment of a group of experts results in a consensus opinion that combines the experts' probability distributions using the operators of normalized product, quotient and exponentiation. The expression of this consensus opinion simplifies to a product of the expert opinions with two assumptions: (1) balanced training of experts, i.e., uniform prior probability over all structures, and (2) conditional independence between expert opinions, given the structure.

This research also studies how Markov chains and hidden Markov models may be used to represent expert opinion. Closure properties are proven, and construction algorithms are given for product of hidden Markov models, and product, quotient and exponentiation of Markov chains. Algorithms for extracting single-structure predictions from these models are also given.

Current product-of-experts approaches in machine learning are top-down modeling strategies that assume expert independence, and require simultaneous training of all experts. This research describes a bottom-up modeling strategy that can incorporate conditionally dependent experts, and assumes separately trained experts.

Acknowledgements

Several persons helped me during the research and writing of this thesis. Forbes Burkowski supervised my research, provided advice, and read many drafts of this thesis. Both my official readers, Paul Kearney and Hugh Chipman, provided excellent comments and valuable criticism on the display draft of this thesis. Both my readers also independently influenced the direction of the research with some early comments on the work.

I wish to thank my colleagues Dana Wilkinson, Mike Hu, and especially Chris Pal for interesting and helpful discussions. I also wish to thank the members of the thesis@bang.dhs.org mailing list: Steve van Egmond (creator, administrator, and active participant), Bill Snow, and Danny D'Amours, for their comments and their support. Bill Snow also read and commented on an early draft of this thesis.

I am grateful for the support and encouragement given to me by Aysa N. September, Alison Cumbaa, and other members of my family.

Financial support was provided by Reba Martin, Bill and Carolyn Cumbaa, and by a grant from CITO.

Contents

List of Figures

Chapter 1

Introduction

Proteins are an important part of molecular biology. They are large biological molecules with complex structure. Protein molecules constitute much of the bulk of living organisms: enzymes, hormones, and structural material. The genetic makeup of an organism is composed of thousands of genes, most of which are *protein-coding*: they specify instructions for building proteins. To understand the life processes of an organism, it is necessary to first know the functions of the proteins produced by the organism. Proteins serve a great many functions in an organism, with some proteins serving multiple functions. The function of a protein molecule in a given environment is determined by its structure. Thus, to fully understand the function of a protein, it is necessary to know the structure of the molecule.

1.1 An Introduction to Protein Structure

The structure of a protein molecule is described in three levels of detail: primary, secondary, and tertiary.

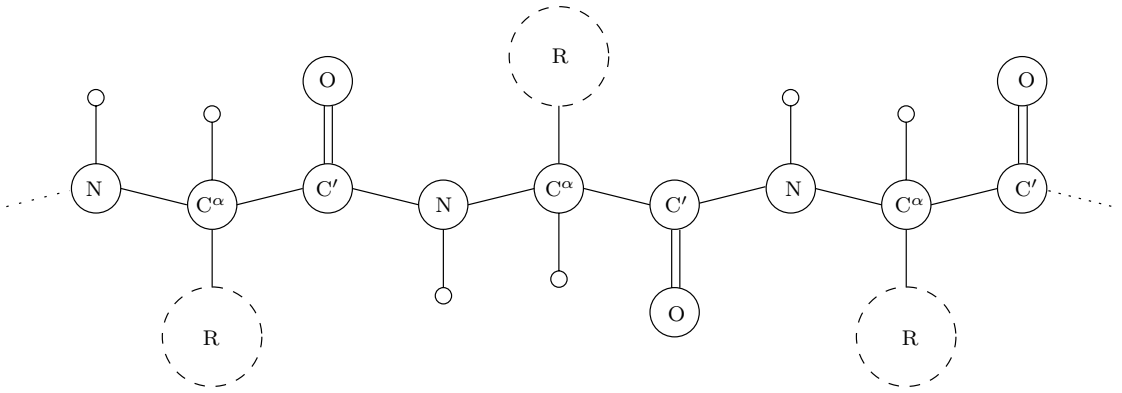

Figure 1.1: Detailed structure of the covalent bonds in a protein molecule. Three amino acid residues are shown. Boundaries of amino acid residues lie along C′–N bonds. Small circles represent hydrogen atoms. The exact structure of the R-group side-chains depends on the type of amino acid.

1.1.1 Primary structure

The primary structure of proteins specifies the structure of covalent bonds[1] in a protein molecule. A protein molecule is a chain of amino acid residues, joined head-to-tail by covalent bonds. Amino acid residues have one nitrogen atom at the head, a carbon atom in the middle (C^α), and a carbon atom (C') at the tail. Peptide bonds join amino acid residues together, linking the C' atom (tail) of one residue to the nitrogen atom (head) of the following residue. The Figure 1.1 illustrates the covalent structure of a protein molecule in detail. Twenty common types of amino acid residues exist. Each type is distinguished by the unique side chain of atoms (the R group) connected at the α carbon. Figure 1.2 lists the twenty amino acids and their abbreviations, and illustrates each R-group.

Assuming the N-C^α-C' chain backbone, primary structure of a protein may thus be summarized by its sequence of amino acid residues. The convention is to list the residues in head-to-tail order (N-terminal to C′-terminal). The ith residue of a protein is the ith residue of the sequence listed in conventional order. A protein with n residues is said to be of length n. A protein-coding gene in an organism specifies the primary structure of a protein: nucleotide triplets (codons) specify

[1] A covalent bond joins two atoms by the sharing of electrons. It is the strongest kind of chemical bond.

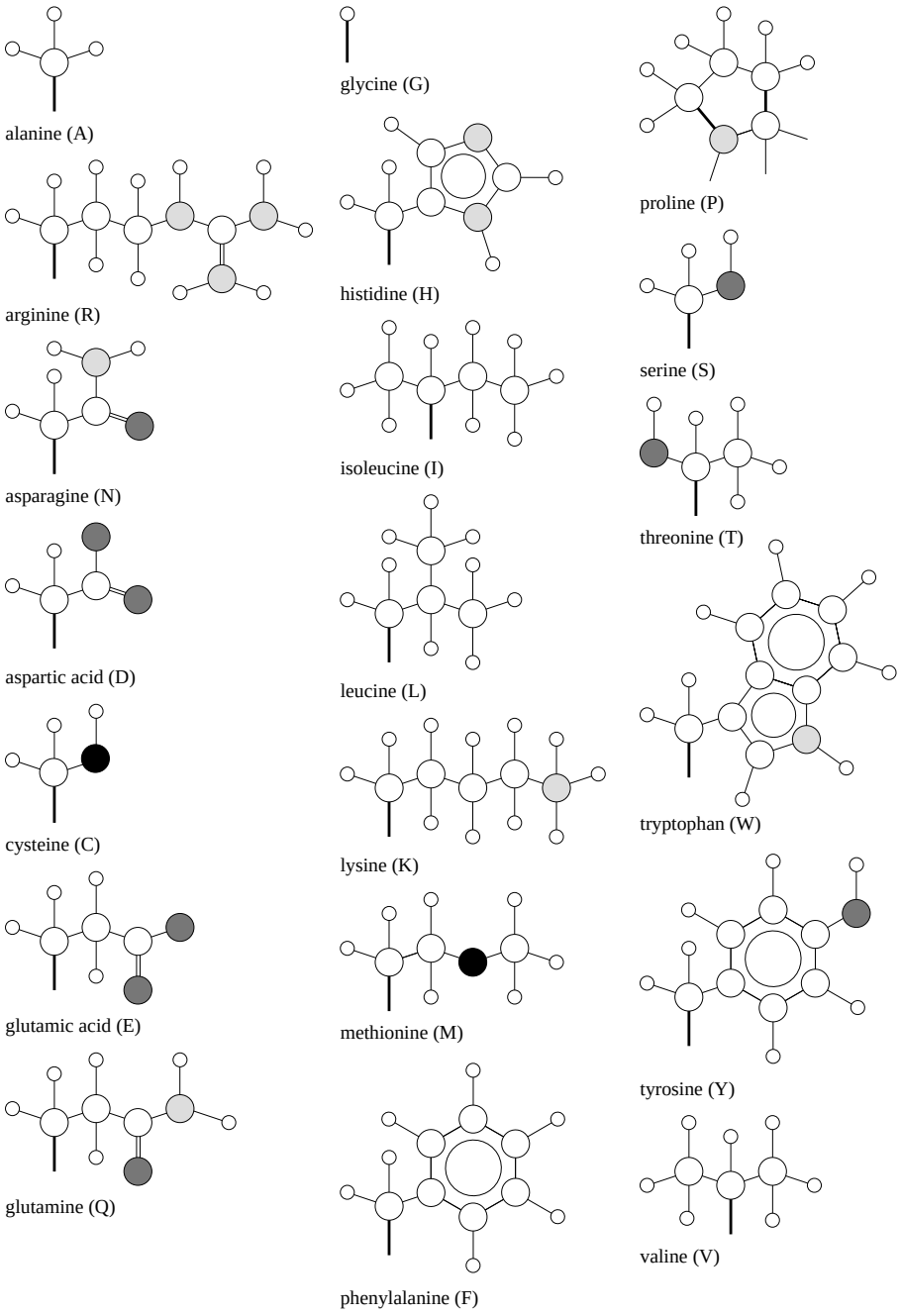

Figure 1.2: The names, abbreviations and R-groups of the twenty amino acids. Atoms are coloured white for carbon, light gray for nitrogen, dark gray for oxygen and black for sulphur. Small atoms are hydrogen. Bonds connecting R-groups to main-chain atoms are drawn in bold. Note that the proline side chain joins the main chain at both C^{α} and N atoms, which are shown.

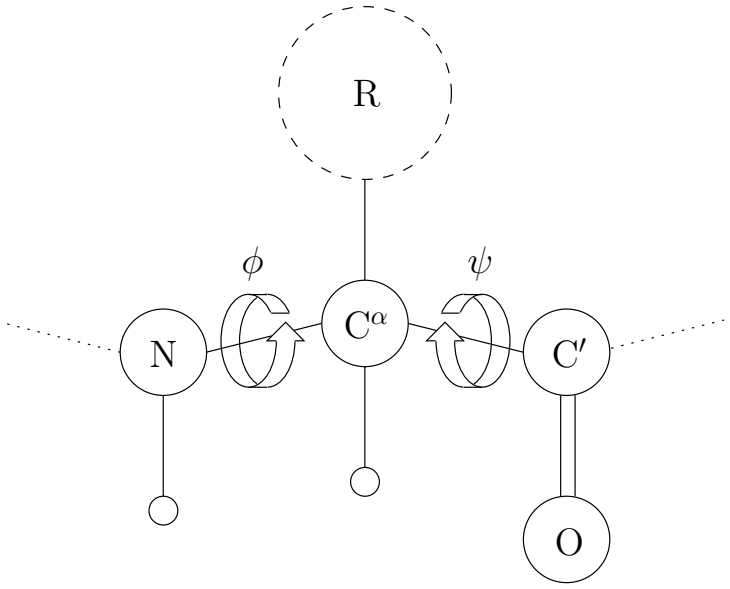

Figure 1.3: The ϕ, ψ angles of an amino acid residue. The rotational freedom of these bonds gives protein molecules flexibility.

individual amino acids; n consecutive codons spell out the sequence of n amino acid residues which constitute a length-n protein.

1.1.2 Secondary and tertiary structure

Protein molecules are not rigid; both N–C^{α} and C^{α}–C' bond angles can rotate in each residue (excepting proline residues, whose N–C^{α} bond angle is fixed). These bond angles are called *Ramachandran angles*, and by convention are denoted as ϕ for the N–C^{α} angle and ψ for the C^{α}–C' angle. Figure 1.3 illustrates the ϕ, ψ bond angles of a single residue. [1]

These ϕ and ψ angles give protein molecules great flexibility; by rotating segments of a protein molecule along ϕ and ψ angles, virtually any shape, or *conformation*, can be achieved. In a solvent environment, a protein will fold according to an energy cost which favours some conformations over others. This energy cost depends on the number and type of weak interactions (hydrogen bonds, ionic bonds, Van der Waals forces, dipole and hydrophobic) formed between pairs of atoms in the protein and between atoms of the protein and atoms of the solvent. Proteins are

typically designed by evolution to exist in a particular solvent environment, which we call their *native environment*. In their native environment, proteins will quickly fold into a single, stable conformation which is the global minimum Gibbs free energy over the space of conformations [2]. This conformation is called the *native fold* of the protein.

The *tertiary structure* of a protein refers to the three-dimensional shape of its native fold. The gross tertiary structure of a protein is captured by the specification of the structure of its *backbone* — the chain of N, C^{α}, and C' atoms stretching from the N-terminal to the C' terminal. The structure of the backbone may be specified by the n (ϕ, ψ) angle pairs, or alternatively by three-dimensional spatial coordinates of each atom in the chain. A full specification of the tertiary structure of a protein requires the details of side-chain conformations, which can again be specified either by bond angles or by atomic coordinates.

Between primary and tertiary structure is an intermediate level of structural description: *secondary structure*. Secondary structure assigns segments of contiguous residues of a protein to one of several conformational classes. These classes describe the participation of each amino acid residue in local conformations of the chain. These local conformations are generally based on patterns of hydrogen bonding between the main-chain nitrogen's hydrogen atom and main-chain oxygen atoms. Several different definitions of secondary structure exist. The most commonly-used definition is that of Kabsch and Sander [3]. They define eight classes of secondary structure:

α **helix:** A helical arrangement of residues, 3.6 residues per turn, formed by hydrogen bonds between the ith main-chain CO and the $i+4$th main-chain NH atoms. Kabsch and Sander denote α-helical residues with the label H. An example α helix is illustrated in Figure 1.4.

π **helix:** Similar to the α helix, with 4.4 residues per turn and hydrogen bonds between the atoms in the ith and $i+5$th residues. Kabsch and Sander denote π-helical residues with the label I.

3_{10} **helix:** Similar to the α helix, with hydrogen bonds between the atoms in the ith and $i+3$rd residues. Kabsch and Sander denote 3_{10}-helical residues with the label G.

Isolated β bridge: A pattern of main-chain hydrogen bonds between a pair of three adjacent

residues. Two kinds of bridges are possible between residues $i-1, i, i+1$ and $j-1, j, j+1$ of a protein:

Parallel bridges have hydrogen bonds connecting the CO $i-1$ to NH j and residues CO j to NH $i+1$, or the residues CO $j-1$ to NH i and CO i to NH $j+1$.

Antiparallel bridges have hydrogen bonds connecting the residues CO i to NH j and residues CO j to NH i, or the residues CO $i-1$ to NH $j+1$ and CO $j-1$ to NH $i+1$.

Kabsch and Sander denote the central residues in an isolated β strand with the label B.

Extended β strand: A set of two or more consecutive β bridges. A β sheet is formed by multiple, adjacent β strands arranged in a plane and joined side-by-side by bridges. β sheets resemble woven fabric: the backbones of the β strands are the warp, and hydrogen bonds between strands form the woof. Strands linked by parallel bridges form parallel β sheets; strands linked by antiparallel bridges form antiparallel β sheets. An example antiparallel β sheet is illustrated in Figure 1.5. Kabsch and Sander denote β-strand residues with the label E.

Turn: An n-turn at residue i in a protein occurs when a hydrogen bond links the CO atom of residue i with the NH atom of residue $i+n$. The value n may be any one of $\{3, 4, 5\}$. Thus n-turns are fragments of 3_{10}, α, and π helices. An example turn is illustrated in Figure 1.5. Kabsch and Sander denote isolated n turns with the label T.

Bend: A bend at residue i in a protein occurs when the angle between atoms C^{α}_{i-2}, C^{α}_{i}, and C^{α}_{i+2} exceeds 70°. Kabsch and Sander denote bends with the label S.

The eighth class of secondary structure, sometimes called *coil* or *loop*, is assigned to any conformation that does not fit into the above seven. Kabsch and Sander denote coils with a blank symbol.

The most commonly-occurring structures are α helices, β strands, and coils. A simplified secondary structure scheme reduces the classes to three:

1. Helices (G, H, I), denoted H.

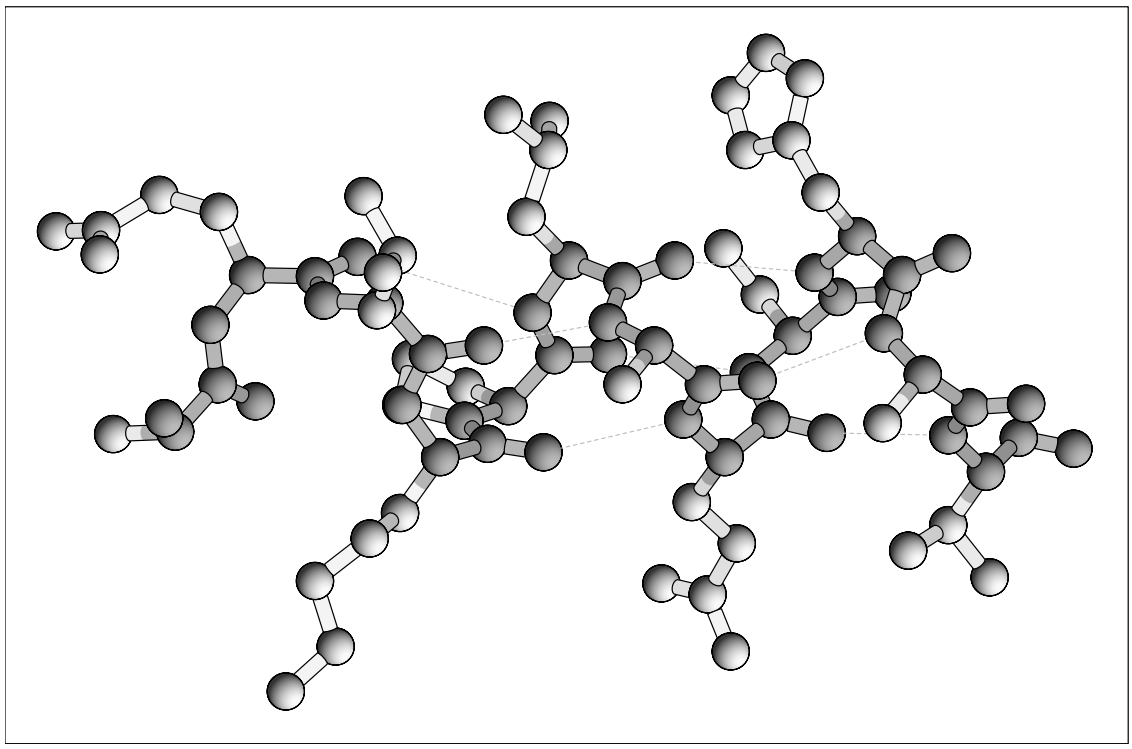

Figure 1.4: Example of an α helix, from sperm whale myoglobin (PDB ID 101M) [4]. The N-terminus is on the left. Backbone atoms are shaded to emphasize the helical structure. Hydrogen atoms are not shown. Dashed lines indicate hydrogen bonds.

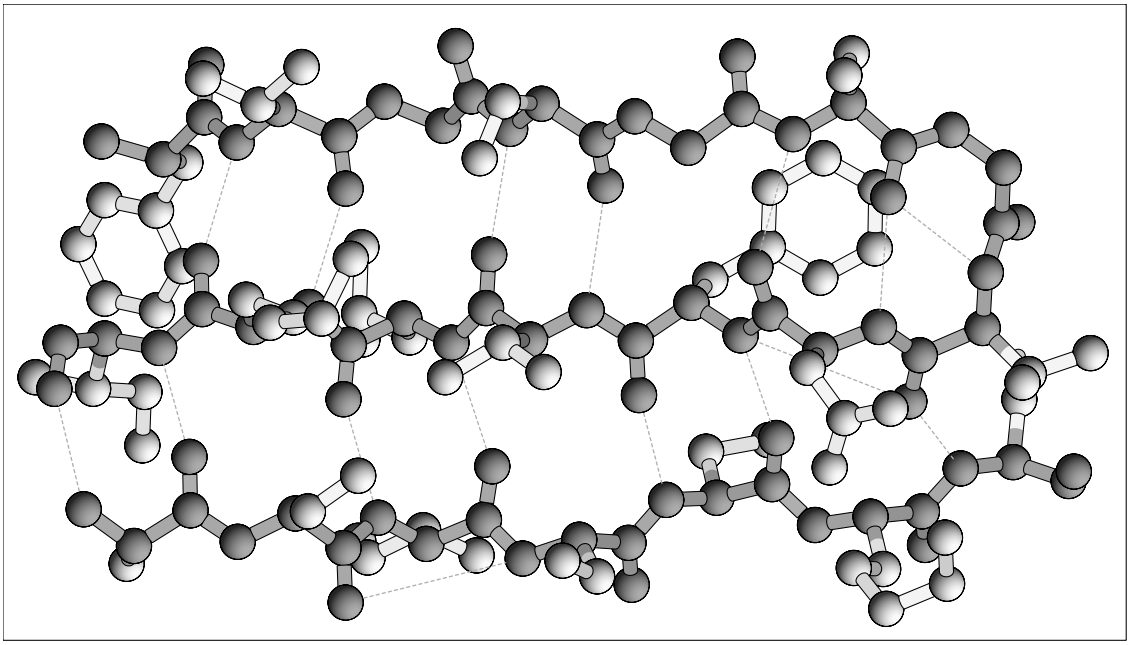

Figure 1.5: Example of an antiparallel β sheet, from the light chain of the anti-prion FAB 3F4 protein complex (PDB ID 1CR9) [5]. Three strands are shown. Note that the top and middle strands are joined by a 3-turn. Backbone atoms are shaded. Some side chains and hydrogen atoms are not shown. Dashed lines indicate hydrogen bonds.

2. Strands (B, E), denoted E.

3. Coil (S, T, coil), denoted C.

A *segment* of secondary structure is a maximal series of consecutive residues sharing the same secondary structure label. Thus the following secondary structure, taken from a fragment of promyelocytic leukemia zinc finger protein (PDB ID 1BUO) [6], has five segments:

$$\underbrace{\texttt{EEEEE}}_{1}\,\underbrace{\texttt{CC}}_{2}\,\underbrace{\texttt{EEEEE}}_{3}\,\underbrace{\texttt{C}}_{4}\,\underbrace{\texttt{HHHHHHH}}_{5}.$$

1.2 The Protein Folding Problem

The *Protein Folding Problem* is that of determining the tertiary structure of a protein, given only its amino acid sequence (primary structure). This problem arises often in biology, since genetic sequences are relatively easy to read, and the translation from codons to protein sequence is trivial. Biologists wishing to understand the function of a gene can therefore quickly obtain the amino acid sequence that the gene encodes. Knowing the tertiary structure of a protein can help determine its function.

The most direct method of determining the tertiary structure of a protein molecule is to observe it experimentally. Two technologies are used: X-ray diffraction and nuclear magnetic resonance (NMR). Both are complex and expensive, and each has unique drawbacks. X-ray diffraction is very time consuming. The main challenge is in creating a crystal of the target protein. This is not always possible, especially for proteins whose native environment is not an aqueous solution, e.g., trans-membrane proteins. NMR techniques are limited to small proteins.

The difficulty of direct observation of protein structure has created an interest in protein structure prediction, and has led to the development of various methods. The most direct prediction methods are theoretical, and attempt to find a structure having the global minimum potential energy. One way of finding this minimum energy state is by molecular force-field simulation of the physical process of folding. This approach involves modeling the interaction of each protein and solvent atom in a system of many thousands of atoms, with time steps measured in thousandths

of picoseconds, for a period of milliseconds to seconds. Current supercomputers are not powerful enough to simulate protein folding more than a small fraction of the time required. It is estimated that twenty years of steady increase in computer power remain before the smallest proteins can be folded in a molecular dynamics simulation [7]. A second theoretical approach tries to find the global minimum of the potential energy function analytically. The complexity of the energy function and the number of parameters make this approach infeasible.

Machine learning methods have had more success at predicting protein secondary structure. Some methods predict tertiary structure; others predict secondary structure. Apart from the inherent difficulty of the problem, three major factors hinder the success of these methods.

1. *Lack of sufficient data.* The main source of data is the Protein Data Bank [8] (PDB), which contains all known protein tertiary structures. As of 17 April 2001, the PDB holds 13027 experimentally-observed proteins. This is a small fraction of the set of all known protein sequences.

2. *Bias in the data.* The proteins in the PDB are not representative of the set of all proteins. Most protein families do not have a single member in the PDB, and many other families are over-represented. As of 17 April 2001, there are 571 entries in the PDB which are some form of hemoglobin, myoglobin, insulin or albumin.

3. *Natural selection against highly stable structures.* Biological organisms require their proteins to be easily disassembled so that their components may be recycled once they have served their function. As such, protein structures must be stable enough to function, but unstable enough to be disassembled when necessary. Nature therefore hides from us the prime examples of well-folding proteins.

1.3 Protein Secondary Structure Prediction

Protein secondary structure prediction is also an important problem. It is necessary to know the secondary structure before the tertiary structure can be predicted [1, p. 251]. Knowledge of secondary structure, even predicted secondary structure, can improve the accuracy of certain

tertiary structure prediction methods by 25% [9]. Another study reports a near-50% reduction of root-mean-squared-deviation errors in tertiary structure prediction when constraints derived from secondary structure predictions are incorporated [10]. Protein structure can also be used to detect distant protein homology.

Protein secondary structure prediction is different than most machine learning classification problems, in that no fixed-length records are involved, and the domain and range are huge (exponential in the length of the sequence). Predicting the structure of a length-n protein is not a single classification problem: a prediction can be incorrect and yet be close, by some measure, to the correct structure. Nor is it a series of n independent, single-residue classification problems: the structure of each single residue depends on the structure of its neighbours and the rest of the molecule.

The Critical Assessment of Techniques for Protein Structure Prediction (CASP) meeting is held biennially to assess the latest methods of protein secondary and tertiary structure prediction. CASP 4, the last conference, was held in 2000. A variety of secondary structure prediction methods, including the most successful of the CASP conferences and those of historical interest, will be described in Chapter 2.

1.3.1 Evaluation of secondary structure prediction

The practical impossibility of making a perfect prediction on a whole sequence necessitates the use of fine-grained accuracy measures for evaluating predictions. There are two common measures for evaluating the accuracy of protein structure predictions: Q_3 and Sov [11]. Both are similarity measures that map a (predicted, actual) pair of structures to a percentage score.

The Q_3 score compares both strings on a residue by residue basis; the Sov score compares both strings segment by segment. The simplest and most common is the Q_3 measure. The Q_3 measure indicates the percentage of correct assignments, residue by residue, of three states of secondary structure: helix, β strand, and coil.

For biologists, who are often interested in the tertiary structure and the biological function of proteins, the Q_3 score can be an inadequate measure of the accuracy of a prediction method. It is

often the number of secondary structure segments and their approximate lengths and positions in a protein that are important. Critics of Q_3 point to myoglobin, a protein with 70% of its residues wound into eight α helices. A prediction of a single α helix spanning the whole myoglobin sequence — an absurd prediction — would therefore have a Q_3 score of 70%.

The Sov [11] measure avoids the problems of Q_3. It measures accuracy by counting predicted and observed segments, and measuring their overlap. The Sov score is computed by summing over segments of secondary structure instead of individual residues. As with the residue-by-residue measures, Sov can measure the accuracy of predicting a single secondary structure class, or the accuracy of an entire prediction consisting of several classes of secondary structure.

1.4 Overview of Research

The central problem addressed by this thesis is the problem of probabilistically modeling protein secondary structure using a pool of experts. My research has focussed on several sub-problems:

- How can we model the multi-expert resolution problem?
- How can we characterize the classes of probability models that can efficiently represent expert opinions (probability distributions)?
- How do you make predictions and combine expert opinions when the opinions are represented by hidden Markov models and Markov chains?

Chapter 2 surveys the some of the work on protein secondary structure prediction, probabilistic modeling, and multi-expert resolution, and places the work of this thesis in those fields. Chapter 3 introduces an approach to modeling secondary structure by combining the opinions of multiple experts, expressed as probability distributions, into a single consensus probability distribution. Finally, Chapter 4 looks in detail at the classes of probability distributions generated by hidden Markov models and Markov chains, and shows how they are suited for building the kinds of models proposed in the previous chapter.

Chapter 2

Related Work

This thesis explores the intersection of three domains: protein secondary structure prediction, probabilistic sequence modeling, and multi-expert opinion resolution. This chapter will discuss research from these areas that relate to the work of this thesis, with an emphasis on protein secondary structure prediction methods.

2.1 Protein Secondary Structure Prediction

Most protein secondary structure prediction methods presented in the literature are machine learning methods. Their concern is with the acquisition and curation of training data, with training methods, and with the trained models that result. They are methods for converting training data into models. In contrast, the emphasis in this research is on combining existing models into a single, more complex and accurate model. This thesis describes a framework for protein secondary structure prediction rather than a particular method. More generally, it describes a framework for probabilistic modeling of any kind of sequence. Despite this difference, there is considerable room for comparison.

2.1.1 Overview of protein secondary structure prediction

There are two fundamental approaches to protein secondary structure prediction: homology modeling (fold recognition) and *ab initio* methods. Both are machine learning methods, typically making use of a database of proteins with experimentally determined structures. Homology modeling methods attempt to find a protein in their database that is homologous to the target protein, and use the structure of the homolog as the prediction for the target. Thus homology modeling methods are essentially nearest neighbour methods, where evolutionary distance or sequence similarity measures serve as the distance measure.

The challenge in homology modeling lies in identifying the homolog with known structure, and then aligning the target sequence with that of the homolog. For more distant homologs, the target sequence is compared and aligned directly with protein structures, and the technique is called fold recognition, or *protein threading*.

The approach works because proteins with homologous sequences tend to have high structural similarity. Homology modeling methods yield highly accurate predictions when a homolog can be found. However, Schneider and Sander report that less than 20% of all known proteins are homologous to at least one protein with known structure (cited in [12]).

Ab initio methods are a more diverse class of prediction methods, encompassing all methods that do not rely on knowledge about the structure of a homolog of the target protein to make their predictions. These methods may instead rely on physico-chemical knowledge, or information extracted from a database of solved protein structures. Most machine learning methods of secondary structure prediction fall under this category.

Local methods

Benner et al. [13] have noted that homology modeling methods may also be distinguished from *ab initio* methods by the underlying assumptions they make on the relative importance of local and non-local effects. Homology modeling methods compare entire sequences, placing less importance on local similarities or dissimilarities. Thus they assume that non-local effects dominate. Most *ab initio* methods make predictions in a piecewise manner, and consider only the local neighbourhood

when making predictions on part of the sequence. Thus they assume that local effects dominate. *Ab initio* methods that incorporate sequence family profile information (see Section 2.1.2 below) are an exception.

The framework described in this thesis can incorporate experts from any of the above categories: homology-modeling or *ab initio*, local or non-local. The origin of the experts' knowledge is inconsequential to the method. It is therefore possible to combine experts from many categories, creating a hybrid prediction method.

2.1.2 Important ideas in *ab initio* prediction

Some of the best methods incorporate several common ideas that solve a particular problem or improve secondary structure accuracy and quality.

Piecewise prediction

Traditional machine learning methods assume a domain of fixed-size records. These methods are not directly applicable to protein secondary structure prediction, because both the domain (amino acid sequences) and range (structure strings) of the problem are variable-length. Prediction methods that are based on these record-based machine learning methods must therefore reduce the problem to one or more fixed-size pieces.

Most prediction methods in this category use a *sliding-window* strategy to accomplish this reduction: the prediction problem on a target sequence of length n is divided into n classification problems, one per residue. The input to the classifier of the ith residue is $s_{i-k} \cdots s_{i+k}$, the window of $2k+1$ consecutive amino acids drawn from the target sequence s and centered on residue i. A special end-of-sequence symbol is added to Σ_{AA} so that windows may be centered on residues close to the endpoints of the target sequence. To predict the whole sequence, the input window is centered over the first residue, and the output of the classifier is recorded. The remaining residues are classified in turn by sliding the input window one residue to the right, until the last residue is classified and the prediction is complete. Methods that use the sliding window strategy include the classic Chou and Fasman method [14, 15], the Levin et al. [16] method, and the neural network

methods such as Qian and Sejnowski [17], PHD [18], and PSIPRED [19].

The sliding-window strategy is a clear example of the local effects assumption at work. The predicted structure of a single residue is determined solely by a local neighbourhood of amino acids surrounding it. Residues outside of the window cannot affect the prediction.

Sophisticated prediction methods that use a sliding window strategy will usually also incorporate a second prediction phase which repairs inconsistencies produced by the prediction obtained from the sliding window phase. The neural network methods previously cited are principal examples.

Balanced training

Machine learning models require a training set of proteins with known secondary structure. Choosing a prior distribution $P(X)$, the distribution of secondary structures in this training set, is a difficult issue. There is no precise definition of the distribution of proteins occurring in Nature. However, it is clear that the distribution of proteins in the Protein Data Bank is not representative of the distribution of proteins in Nature: entire protein families are missing from the PDB, and some individual proteins have multiple entries. Neither the distribution of proteins in Nature, however defined, nor the distribution of proteins in the PDB is the same as the distribution of proteins whose structures are being predicted by prediction tools today. This last distribution is impossible to determine.

Balanced training prevents imbalance in trained models of protein secondary structure by imposing a uniform probability distribution (prior) on the structure data. While a machine learning model of protein secondary structure trained on a database with non-uniform distribution of structures may score better on a test set with a similar distribution, balanced training ensures a better overall accuracy of the model when the distribution of the test set is unknown [20].

The theory developed in chapter 3 is simplified by the assumption of a uniform prior (balanced experts); see the corollary to Theorem 3.2.3. However, the same theory reveals how an expert opinion (probability distribution) trained on an unbalanced training set may be balanced retroactively; it also shows how to create a balanced model from a set of unbalanced experts.

Sequence family profiles

Prediction accuracy is improved by incorporating sequence family profile information, such as that obtained from a multiple sequence alignment. Rost and Sander observed an increase of six percentage points in the Q_3 accuracy of their neural network method when including a profile of the amino acid frequencies occurring in the column of the multiple sequence alignment of each residue of the target sequence [20]. Further improvements lead to the PHD method [18], which supplements the profile with global sequence information. Two types of information are present in multiple sequence alignments that can help predictions: a profile of the frequency of amino acids that occur in each column, and a profile of the insertion/deletion points of the consensus sequence.

Protein secondary structure prediction methods that incorporate multiple sequence alignment information into their predictions include PHDsec [18], PSIPRED [19], and PREDATOR [21].

Neural network methods

Many successful protein secondary structure prediction methods use neural networks. Qian and Sejnowski [17] were the first to use neural networks for *ab initio* protein secondary structure prediction; their work was based on a speech generation method co-developed by one of the authors.

Neural networks assume a fixed-size input and output; thus, they may only be used for piecewise prediction. Qian and Sejnowski developed a two-stage, sliding-window method to achieve piecewise prediction. The first stage employed a sliding-window strategy with window size 13 or 17. The output layer classified the central residue in the window, and consisted of three units, one per secondary structure class.

The accuracy of the raw prediction of the first stage was improved in the second stage by cascading the first-stage outputs into the inputs of a second stage neural network. A sliding-window technique was again used. Outputs of the first-stage network taken from 13 consecutive predictions were fed into a second network as inputs. This second network then produced a final structural classification of the central residue.

The two-stage, cascaded-inputs strategy of Qian and Sejnowski shares some elements with the modeling technique developed in this thesis.

- The neural network of the first stage may be interpreted as a protein secondary structure expert. The difference is that this expert is a function $e : \Sigma_{AA}^1 3 \rightarrow \mathbb{R}^3$. Its input is a window of amino acids $s_{i-6} \cdots s_{i+6}$ rather than a variable-length sequence s. Its output is a triple of reals, which may be loosely interpreted as a probability distribution over the possible structures of the ith residue, though the three values are not constrained to sum to one. The output makes no prediction on the structure of adjacent residues or other regions of the target. This is equivalent to assigning the background probability distribution to the rest of the target.

- The neural network of the first stage is used to produce n expert opinions.

- These opinions were then combined in the second stage. The key difference between the method of this thesis and the Qian and Sejnowski method is that they employ a neural network to combine expert opinions; conversely, the opinions are combined by taking their product. Their method of combining expert opinions is learned from data, whereas the method of combination in this thesis was determined analytically (see Chapter 3).

Other neural network methods that followed, e.g., [18], employed the same strategies introduced by Qian and Sejnowski (sliding window, cascading inputs).

2.1.3 Disadvantages of single-structure prediction

Protein folding is a complex, but essentially deterministic process. If we use $\mathcal{S}$ to represent the set of possible amino acid sequences, and $\mathcal{X}$ to represent the set of possible protein secondary structures, the folding process is represented by a function $f : \mathcal{S} \rightarrow \mathcal{X}$ mapping amino acid sequences in $\mathcal{S}$ to structures in $\mathcal{X}$.

Typical protein secondary structure prediction methods reflect this determinism by providing a single predicted structure, representing their best guess of the true secondary structure of the target molecule, given the amino acid sequence. Yet this single guess rarely matches the observed

structure exactly. (In the CASP 3 competition, where multiple teams competed to predict the secondary structure of multiple proteins, the mean Q_3 score of predictions per protein ranged from 37.5% to 95.5% [22].) Thus, all predictions of protein structure contain uncertainty. The single structure produced by a prediction method is chosen by the method on some basis, either to maximize the probability of a perfect prediction, or to maximize the expected similarity of the prediction with the correct structure. These goals are not identical, and are discussed further in Section 3.1.1. The main disadvantage, then, of a single-sequence prediction is that the goal of the predictor may not be the same goal as the user of the prediction method. Thus it is better to reveal the uncertainty of a prediction than to keep it hidden. A complete description of the uncertainty of a prediction is given by a probability distribution $P(X)$ over possible structures $x \in \mathcal{X}$.

Some sophisticated prediction methods do include a coarse measure of uncertainty in their predictions, such as the reliability index of Rost and Sander's PHD method [23]. Their reliability index gives an integer score from 0 to 9 for each residue, indicating the degree of certainty that the predicted structure at that residue is the correct structure. Residue-by-residue reliability indices of this form fall short of a complete description of the uncertainty of the prediction in three respects:

1. The reliability score is coarse-grained.

2. The relative probabilities of the non-predicted structure classes for each residue are not given.

3. The conditional probabilities are unspecified.

Prediction methods that use probabilistic models calculate the full probability distribution. The hidden Markov model prediction methods of Asai et al. [24] and Bystroff et al. (HMMSTR) [25] are examples. Probabilistic modeling methods such as Asai et al. and HMMSTR are especially important to this thesis, because they can serve as experts; see Theorem 4.2.4.

2.2 Probabilistic Sequence Modeling

The field of probabilistic sequence modeling spans several domains, from speech recognition to biological sequence analysis. Some sequence data is continuous, time-series data, such as acoustic signals used in speech recognition. Other sequence data, principally biological sequence data, is discrete, spatially-oriented sequence data. Examples include genetic sequences, amino acid sequences, and protein secondary structure.

The principal tool of probabilistic sequence modeling is the hidden Markov model (HMM). Hidden Markov models are powerful tools for probabilistic analysis of biological sequences. Their use spread to the field of bioinformatics after successful application in the field of speech recognition.

HMMs are used to solve many of the principal problems in bioinformatics:

1. *Gene finding.*
2. *Phylogenetic tree reconstruction:* tree HMMs.
3. *Sequence alignments:* profile HMMs (multiple sequence alignments), e.g., [26], pair HMMs (pairwise alignments).
4. *Sequence similarity scoring:* profile HMMs.
5. *Protein secondary structure prediction,* e.g., Asai et al. [24], HMMSTR [25]

Hidden Markov models are stochastic sequence generators. They generate strings x according to a probability distribution $P(x)$. To generate a sequence, a hidden Markov model moves through a hidden sequence of internal states that affect the generated sequence. The joint probability of generating a sequence x while traversing a hidden state path π is denoted $P(x, \pi)$. The power of HMMs lies in their ability to ascribe meaning to these hidden states. Predictions are made by inferring the hidden state sequence that generated the string, $P(\pi \mid x)$. Probabilistic inference on the posterior $P(\pi \mid x)$ is called *decoding.*

The common factor in all the applications listed above is the use of HMMs as experts in each problem domain. Predictions (expert opinions) are obtained from these HMMs via decoding.

Chapter 4 studies the use of HMMs as representations of expert *opinions*, not experts. Thus decoding is not an issue in this research, though it is briefly addressed in Section 4.2.4. The connection between HMMs used as experts and HMMs used as expert opinions is made by Theorem 4.2.4.

2.3 The Multi-expert Resolution Problem

This thesis addresses the multi-expert resolution problem: faced with k expert opinions, expressed as probability distributions $P_1, P_2, \ldots, P_k$ over a space of events $\mathcal{X}$, what is the consensus opinion P_*? This problem of combining expert opinions has been studied by many. Their work is surveyed by Genest and Zidek [27], among others. Genest and Zidek identify two main approaches to the problem: linear opinion pools (mixtures of experts) and logarithmic opinion pools (products of experts).

A linear opinion pool, otherwise known as a mixture or sum of experts, combines the expert opinions linearly:

$$P_*(x) \propto \sum_{i=1}^{k} w_i P_i(x),$$

where the w_i values are weights representing the subjective quality of each expert opinion P_i. P_* is thus a weighted mean. The implicit assumption when using a linear opinion pool is that one expert is the correct expert; when normalized so that $\sum w_i = 1$, the weights w_i represent the probability that expert opinion P_i is the correct opinion. A second assumption is that the expert opinions are not *calibrated* (see Definition 3.2.1). The machine learning field refers to methods for generating linear opinion pools as *ensemble methods*.

A logarithmic opinion pool, otherwise known as an independent opinion pool [28] or a product of experts, combines the expert opinions by taking their product:

$$P_*(x) \propto \prod_{i=1}^{k} P_i(x).$$

A generalization of this formula introduces exponential weights to each expert:

$$P_*(x) \propto \prod_{i=1}^{k} (P_i(x))^{w_i} .$$

Morris [29, 30, 31] has shown that this multiplicative method of expert opinion combination is appropriate when the expert opinions are conditionally independent on x, and when the working model is that of a Bayesian decision maker with a prior opinion $P(x)$ who consults these k experts, and updates his or her belief in the outcome x based on these opinions. This model is discussed further in Section 3.2.3, where the product of experts approach is used to combine expert opinions on protein structure.

2.3.1 Modeling complex processes with products of experts

Section 3.3.1 describes a strategy for modeling complex processes using products of experts. Hinton has developed a similar strategy, with applications to handwriting recognition and image recognition [32]. Hinton's work differs from this research on several points:

- *Top-down versus bottom-up.* Hinton assumes the complex process is the result of the product of several experts. His technique divides the process in top-down fashion into multiple sub-processes, each represented by an expert. The entire model, consisting of the product of the sub-models, is trained at once on the training data. Parameters of the sub-models are estimated using Gibbs sampling [33].

  This research creates complex models in a bottom-up fashion by assembling simpler, separately-trained models and then computing their product.

- *Independence assumptions.* Hinton assumes his experts are conditionally independent. This research allows for conditionally dependent models.

- *Emphasis on decoding.* Hinton makes predictions by decoding his models, i.e., inferring from his models the hidden state π that produced the observed data x, using a posterior probability distribution $P(\Pi = \pi \mid X = x)$. Thus Hinton is taking products of experts,

whereas this research studies products of expert opinions.

2.3.2 Product of hidden Markov models

Chapter 4 studies the use of hidden Markov models to represent expert opinion. Theorem 4.2.2 shows that the product of hidden Markov models is also a hidden Markov model. HMMs created by the construction in the proof of that theorem are distributed-state models that resemble Zoubin and Gharamani's Factorial HMMs [34] and Brand's Coupled HMMs [35, 36].

Chapter 3

Probabilistic Modeling with Experts

3.1 Probabilistic Treatment of Protein Structure

This chapter discusses a method for dealing with experts, considered here as generators of probability distributions that represent our uncertainty about the secondary structure of a protein. Nature provides a protein folding function, $f : \mathcal{S} \rightarrow \mathcal{X}$, that maps protein primary structure ($\mathcal{S} = \Sigma^*_{AA}$, strings over the amino acid alphabet) to protein secondary structure ($\mathcal{X} = \Sigma^*_{SS}$, strings over the alphabet of secondary structure classes). Because secondary structure is a labeling of each residue in an amino acid sequence, f maps length-n amino acid sequences ($\mathcal{S}_n = \Sigma^n_{AA}$) to equal-length secondary structure strings ($\mathcal{X}_n = \Sigma^n_{SS}$). The function f is unknown, and is at the heart of our uncertainty. We may represent this uncertainty as a formal probability space $(\Omega, \mathcal{F}, P)$, where

Ω is the sample space. It is the set of all possible values of f, i.e., all mappings

$$\{\omega : \mathcal{S} \rightarrow \mathcal{X} \mid \forall n \in \mathbb{N}_{>0}, s \in \mathcal{S}_n \implies \omega(s) \in \mathcal{X}_n\}.$$

$\mathcal{F}$ is the set of events. Elementary events are subsets of Ω of the form $\{\omega \in \Omega \mid \omega(s) = x\}$, for every $x \in \mathcal{X}_n$ and for some fixed target sequence s of length n. In other words, we are only concerned with the structure of the target, and do not distinguish between elements of Ω that assign the same structure to the target.

P is a probability measure (distribution) on the measurable space $(\Omega, \mathcal{F})$.

We introduce the random variable $X : \Omega \to \mathcal{X}_n$ to represent the structure of the target protein. X is a mapping from the sample space to length-n structures, defined as $X(\omega) = \omega(s)$. The event $X = x$, for any $x \in \mathcal{X}$, is the event that the target protein has structure x. It is the same as the elementary event $\{\omega \in \Omega \mid \omega(s) = x\} \in \mathcal{F}$. Thus $P(X = x)$ is the probability that the target protein has structure x.

We also introduce the random variable $X_i : \Omega \to \Sigma_{SS}$, for $i \in \{1, \ldots, n\}$, to represent the structure of the ith residue of the target. If $\omega(s) = x$, then $X_i(\omega) = x_i$.

A probability distribution $P(X)$ over structures $x \in \mathcal{X}$ partitions $\mathcal{X}$ into two disjoint subsets: possible and impossible structures. This is equivalent to a definition of a formal language. A language $L \subseteq \mathcal{X}$ is defined by its characteristic function $\chi_L : \mathcal{X} \to \{0, 1\}$:

$$\chi_L(x) = \begin{cases} 1 & \text{if } x \in L, \\ 0 & \text{otherwise.} \end{cases}$$

A probability distribution $P(X)$ is a mapping $\mathcal{X} \to [0, 1]$, and thus serves as a generalized characteristic function.

Definition 3.1.1. *The* language of probability distribution $P(X)$, *denoted* $L(P)$, *is the set of strings in* $\mathcal{X}$ *with positive probability:*

$$L(P) = \{x \in \mathcal{X} \mid P(X = x) > 0\}.$$

3.1.1 Using a probability distribution for prediction

A probability distribution $P(X)$ over $\mathcal{X}$ or $\mathcal{X}_n$ represents a vast amount of information. There are several interesting questions we may answer with the help of the distribution. The most fundamental question is what probability is assigned to a particular structure: $P(X = x)$. A second question, asked when a single prediction is required, asks what is the best or most likely structure. "Best" will have different meanings, depending on the goals of the prediction. A third question asks for representative structures from the distribution, i.e., randomly choose a structure (or k structures) from the distribution $P(X)$. A fourth question asks for the expected value $E[g(X)]$ of some function g. The function g could, for example, count the number of helix segments in the structure, or calculate the percentage of the structure in coil conformation.

How a probability distribution is used to make a single prediction, i.e., to choose the "best" structure, depends on the purpose of the prediction. $P(X = x)$ is the probability that any single prediction x is the correct structure. Thus the structure with maximal probability,

$$x^* = \operatorname*{argmax}_{x \in \mathcal{X}_n} P(X = x),$$

is the structure to choose if a single structure is desired and the predictor wishes to maximize the chance of making a correct prediction.

This is not always the case, because a simple correct/incorrect judgment of a prediction is often too severe. Intuitively, two predictions may be incorrect, but one may resemble the actual structure more closely than the other. This concept of similarity between structures has been formalized; two common distance metrics for comparing strings are the edit distance and Hamming distance metrics. For protein secondary structures, the Sov [11] and Q_3 measures are used. (Q_3 is equivalent to Hamming distance.) Often, the predictor wishes to minimize the expected distance between the prediction and the actual structure. Thus the predictor will choose the structure $\hat{x}$, where

$$\hat{x} = \operatorname*{argmin}_{x \in \mathcal{X}} \sum_{x' \in \mathcal{X}_n} P(X = x') d(x, x')$$

and $d(x, x')$ is the distance measure.

Of the two standard measures of prediction accuracy, Q_3 is the simplest, and is the easiest measure for which to compute the structure which maximizes expected similarity. Unlike Sov, Q_3 considers the structure of each residue separately: $Q_3(x, x') = \frac{1}{n}\sum_{i=1}^{n}[x_i = x'_i]$. Thus the predicted structure of each residue may be chosen independently so as to maximize its expected contribution to the Q_3 score:

$$\begin{aligned}\hat{x}_i &= \operatorname*{argmax}_{\sigma \in \Sigma_{SS}} \frac{1}{n} \sum_{x' \in \mathcal{X}_n} P(X = x')[\sigma = x'_i] \\ &= \operatorname*{argmax}_{\sigma \in \Sigma_{SS}} P(X_i = \sigma).\end{aligned}$$

The structures x^* and $\hat{x}$ may very well not be the same. Indeed, $\hat{x}$ may even be an impossible structure, i.e., $P(X = \hat{x}) = 0$. By the same token, x^*_i may not be the most likely structure of the ith residue, for any i. This distinction between x^* and $\hat{x}$ has parallel in the English language. If $\mathcal{X}$ were the set of words made of the 26 letters of the Roman alphabet, and if $P(X)$ were the distribution of those words in English text, then we would have $x^* = \textit{the}$, yet $\hat{x}_i \neq h$. In other words, the word *the* occurs more frequently than any other word in English text, yet "h" is not the most commonly occurring second letter in English words.[1]

The remainder of this chapter describes a framework for modeling protein secondary structure in terms of probability distributions.

3.2 Experts

Here we introduce the concept of *experts*. An expert is a representation of incomplete knowledge about the folding function f; it is a function mapping sequences to probability distributions over the structure space. If e is an expert, the value $p = e(s)$ is a probability distribution, where $p(x)$ gives the expert's subjective assessment of the probability that s has structure x, given e's

[1] If we consider the assembled texts of the four Thomas Hardy novels *Jude the Obscure*, *The Mayor of Casterbridge*, *Return of the Native*, and *Tess of the d'Urbervilles, A Pure Woman* to be a representative sample of the English language, the most commonly occurring second letter of words is *o*.

knowledge of f. It is an *expert opinion*, a measure of the expert's uncertainty about the structure of s.[2]

We will assume that expert opinions are reasonable, in the sense that expressions of certainty must be correct. In other words, if an expert opinion assigns a zero probability to structure x, i.e., $p(x) = 0$, then we must have $f(s) \neq x$. This condition prevents the situation where two expert opinions cannot be reconciled because the intersection of their languages is null.

Morris's concept of calibration [30] is a stronger condition:

Definition 3.2.1 (Calibration). *An expert opinion $p = e(s)$ is* calibrated *if*

$$P(X = x \mid e(s)) = p(X = x),$$

i.e., given no other information, our subjective belief in the structure X is the same as expressed by the expert.

3.2.1 Sources of expert information

Probabilistic information about protein secondary structure can be obtained from many sources. Any information about protein structure in general, or specifically about the structure of the target protein may be represented by an expert and used for prediction. We may distinguish between two types of expert: those that know the length of the target protein and those that do not. Experts knowing the length of the target amino acid sequence s also know the length of the secondary structure string. Thus their opinions are distributions over the space $\mathcal{X}_{|s|} = \Sigma_{SS}^{|s|}$. Experts ignorant of the length of the target yield opinions with distributions over the whole space $\mathcal{X} = \Sigma_{SS}^{*}$.

[2] Although $p = e(s)$ represents an expert's uncertainty about the structure of s, the expert may not be confident of this assessment: the expert may be uncertain about its uncertainty. This double-uncertainty would exist in a machine-learning model with a small training set, with confidence intervals associated with each $p(X = x)$ value.

Sequence length

An expert knowing nothing more than the length (n) of the target sequence will assign uniform probability to all structures in $\mathcal{X}_n$, and zero probability to all structures in $\mathcal{X} \setminus \mathcal{X}_n$.

Amino-acid-sequence patterns

Certain amino-acid-sequence patterns have been discovered whose occurrence in an amino-acid sequence correlates strongly with specific protein structure [37, 25]. If enough occurrences of these patterns are found in a protein-structure database, a pattern-based expert may infer the distribution of secondary structures for proteins matching the pattern. Experts based on amino-acid-sequence patterns are *rational experts* (see Section 3.2.2).

Protein family profiles

Evolution tends to preserve helices and beta strands; thus evolutionary events such as insertion or deletion of residues in proteins tend to occur in the coil regions. Sophisticated protein secondary structure prediction methods (e.g., [18]) exploit this information by determining the evolutionary family of the target protein and then computing the optimum alignment of the target protein with the family. Regions in the alignment where insertions and deletions are common indicate regions where the coil conformation occurs with high probability.

A family profile expert exploits this information, assigning higher probabilities to structures with coil segments in the insertion/deletion regions and helix and strand segments in the unbroken regions.

Structure grammars

An expert modeling secondary structure as a formal language may employ a grammar to disqualify impossible structures. Simple grammars can enforce relatively low-level constraints, such as the minimum lengths of helices and strands. More sophisticated grammars can enforce higher-level aspects of protein structure, such as the requirement that beta strands be part of a beta sheet.

The following grammar generates secondary structure strings, and imposes two constraints: first, that helix, sheet, and coil segments have minimum lengths 3, 1, and 1, respectively; second, that strings must start and end with a coil segment.

$$\begin{aligned}
\mathbf{S} &\rightarrow C\mathbf{S}'C \\
\mathbf{S}' &\rightarrow HHH\mathbf{A}\mathbf{S}' \mid \mathbf{B}\mathbf{S}' \mid \mathbf{C}\mathbf{S}' \mid \Lambda \\
\mathbf{A} &\rightarrow H\mathbf{A} \mid \Lambda \\
\mathbf{B} &\rightarrow E\mathbf{B} \mid \Lambda \\
\mathbf{C} &\rightarrow C\mathbf{C} \mid \Lambda.
\end{aligned}$$

If we call that grammar G, the language specified by it is denoted by $L(G)$. A grammar-based structure expert would assign $p(x) = 0$ if $x \notin L(G)$, and $p(x) > 0$ otherwise.

Physico-chemical models

An expert may form an opinion on the protein structure based on a physical model of the folding protein. Although an accurate atomic-scale simulation of the folding process is not feasible with current computing resources, a simplified simulation experiment may indicate areas in the amino acid chain with strong preference for or aversion to certain secondary structures.

Physical observation

An expert based on incomplete observation of the target protein's structure may be able to exclude some structures or favour others. A low-resolution X-ray crystallography experiment can reveal the structure of some regions of a protein but not others. NMR studies of proteins can also be a source of structural constraints. An expert based on these incomplete observations would give a uniform probability to those hypothetical structures matching the observed details of the actual structure, and give zero probability to all other structures.

Circular dichroism experiments can quickly determine the proportions of helices, β strands and coils that constitute a target protein's secondary structure [38]. An expert based on a circular

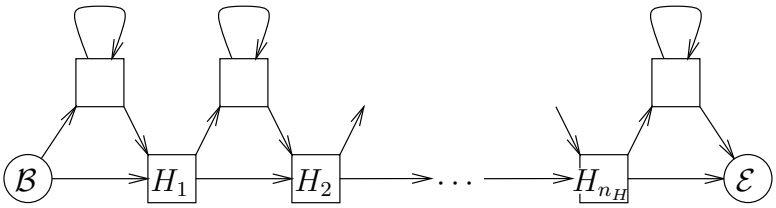

Figure 3.1: An expert opinion asserting that the target has n_H helix residues, implemented by a probabilistic automaton. The automaton uses the states H_i to count the number of helix residues in the structure. (See Chapter 4.)

dichroism experiment revealing secondary structure proportions c_H, c_E and c_C would give the highest probability to those structures having secondary structure content in those proportions, and very low probability to hypothetical structures with significantly different proportions. Figure 3.1 shows how an automaton such as a hidden Markov model (see Section 4.2) could represent an expert opinion asserting that the structure must have exactly n_H helix residues.

3.2.2 Rational experts

Earlier, we discussed amino acid sequence patterns as a source of information for protein secondary structure experts. A sequence pattern π defines a partitioning of the sequence space into two subsets, or languages: L and its complement, $\bar{L}$. All sequences in the sequence space matching π are members of L; all others are members of $\bar{L}$. More generally, we can partition the sequence space into an arbitrary number of disjoint subsets $L_1, L_2, \ldots, L_k$. We will now focus on a definition for experts whose opinions are based on how they partition the sequence space. We call these experts *rational* experts.

The incomplete information about f is represented in a rational expert as follows. A rational expert partitions $\mathcal{S}_n$ into some finite number k of disjoint languages (subsets): $\mathcal{S}_n = L_1 \cup L_2 \cup \cdots \cup L_k$. For each subset L_i, the expert keeps a probability distribution $p_i(x)$, which is the distribution of secondary structures of the sequences in that subset. In other words, $p_i(x)$ is the probability that $f(s) = x$ given that $s \in L_i$. The result of query $e(s)$ on expert e is simply the probability distribution p_i kept by e, where $s \in L_i$.

Rational experts are ideal versions of certain kinds of machine-learning models, such as decision

trees and k-nearest neighbours, that partition their domain into disjoint subsets, and that treat all elements within a subset identically. Rational experts are ideal because their training data for each opinion $e(s \in L_i)$ is complete, i.e., all of L_i, as opposed to a small sample from L_i. Studying the properties of rational experts may yield insights which may also be applicable to these machine-learning models.

Theorem 3.2.1. *The opinions of rational experts are calibrated.*

Proof. Let $p = e(s)$ be the opinion of a rational expert e. If we are given opinion p and no other information about the structure of s, we know only that there is some subset $L_s \subseteq \mathcal{S}$ such that $s \in L_s$, and for every $x \in \mathcal{X}_n$, there is some number c_x of sequences in L_s with structure x such that $c_x / |L_s| = p(x)$. Thus the probability that $f(s) = x$ is same as the probability of drawing a protein at random from L_s and observing structure x:

$$P(X = x \mid e(s)) = \frac{c_x}{|L_s|} = p(x).$$

□

For example, if we had $\Sigma_{AA} = \{a, b\}$, then $\mathcal{S}_3$ would contain eight sequences:

$$\mathcal{S}_3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}.$$

Let $\Sigma_{SS} = \{H, E, C\}$ and f be defined as follows.

s	$f(s)$
aaa	HHH
aab	HHH
aba	EEC
abb	EEC
baa	EEE
bab	CCC
bba	CCC
bbb	HHH

Finally, let e be a rational expert that partitions $\mathcal{S}_3$ into two subsets: $L_1 = \{aaa, aab, aba, abb, baa\}$ and $L_2 = \{bab, bba, bbb\}$. Then the probability distributions kept by e are obtained from the definitions of f, L_1, and L_2:

x	$p_1(x)$	$p_2(x)$
HHH	2/5	1/3
EEC	2/5	0
EEE	1/5	0
CCC	0	2/3

If we choose a particular sequence from $\mathcal{S}_3$, say aaa, the difference between folding function f and expert e becomes clear. $f(aaa) = HHH$ is a deterministic assignment of structure to sequence. In contrast, $e(aaa) = \{p(HHH) = 2/5, p(EEC) = 2/5, p(EEE) = 1/5\}$ is a probabilistic assignment of structure to sequence.

Note that an expert cannot distinguish between sequences which fall into the same subset of the partition of $\mathcal{S}_n$. The individual values of $f(s)$, for each $s \in L_i$ are blurred together into a single probability distribution. It is this blurring which makes an expert's knowledge about f incomplete.

Two further examples of experts are the boundary cases. The first boundary case is the all-generalizing expert, $e_\circ$. Its partition of $\mathcal{S}_n$ is the null partition, which does not divide $\mathcal{S}_n$. (So $k = 1$ and $L_1 = \mathcal{S}_n$.) For any $s \in \mathcal{S}_n$, $p_\circ = e_\circ(s)$ yields the probability distribution of structures across all of $\mathcal{S}_n$. Thus it does not distinguish any sequences at all, in the sense that its output is independent of s. Using the above definition of $f(s)$, we would have

x	$p_\circ(x)$
HHH	3/8
EEC	1/4
EEE	1/8
CCC	1/4.

The second boundary case is the all-specializing expert, $e_\bullet$. It partitions $\mathcal{S}_n$ such that every element is in its own subset; each subset L_i is a singleton set $\{s_i\}$. So $p_\bullet = e_\bullet(s)$ yields the probability distribution where $p_\bullet(x) = 1$ for $x = f(s)$, and 0 elsewhere. If we had $f(aaa) = HHH$, and $p_\bullet = e_\bullet(aaa)$, then we would have $p_\bullet(HHH) = 1$. Thus $e_\bullet$ represents complete knowledge of f, and is the only expert whose opinions have zero uncertainty.

3.2.3 Combining expert opinions

Experts are used to predict protein secondary structure. When an expert e_A is queried for its opinion on target sequence s, the result is $p_A = e_A(s)$, a probability distribution. Considering only the opinion given by e_A, and assuming that $e_A(s)$ is a calibrated opinion, the probability that $f(s) = x$, for any structure x, is $p_A(x)$. In other words, $P(f(s) = x \mid e_A) = p_A(x)$. Prediction then proceeds as required, according to Section 3.1.1.

When more than one expert's opinion is considered, the situation becomes more complicated. If, in addition to querying e_A for its opinion on s, we also query expert e_B, then we have two expert opinions to work with: $e_A(s)$ and $e_B(s)$. What then is $P(f(x) \mid e_A, e_B)$? This is the problem of combining expert opinions.

The problem of combining expert opinions has been studied in the statistical literature. Vari-

ations of this problem, and approaches to their solutions are reviewed by Genest and Zidek [27].

We will apply Bayesian methods to illuminate this problem. Following Morris [29], we regard the voicing of an expert opinion as an event. This allows us to introduce the notions of the probability of a particular opinion, and the conditional dependence or independence of a set of expert opinions, given a structure.

We begin by distinguishing between conditionally dependent and independent expert opinions.

Before querying e_A or e_B on the structure, all we know is $P(X)$, the prior probability distribution, equivalent to the background distribution of structures of $\mathcal{S}_n$. (Note that $P(X) = e_\circ(s)$, the result of querying the all-generalizing rational expert on the structure of s.) In this state, we can assign a probability distribution $P(p_A)$ over the set of opinions that may be voiced by expert e_A. This is the prior probability of opinion $e_A(s)$.

If we are then given the actual structure of s, our evaluation of the probability of hearing opinion p_A from e_A may change. We write this posterior probability as $P(p_A \mid X = x)$. (Intuitively, if we trust our expert e_A, and we know that $f(s) = x$, for some specific structure x, we will expect $e_A(s)$ to give high probability to x.) Similarly, the posterior probability of an opinion p_B voiced by e_B is $P(p_A \mid X = x)$. The notion of conditional independence of $e_A(s)$ and $e_B(s)$ is determined by the value of the joint posterior probability, $P(p_A, p_B \mid X = x)$:

Definition 3.2.2. *Let $e_1, e_2, \ldots, e_k$ be a set of k experts. The set of expert opinions $p_1 = e_1(s), p_2 = e_2(s), \ldots, p_k = e_k(s)$ are conditionally independent, given $f(s) = x$, if*

$$P(p_1, p_2, \ldots, p_k \mid X = x) = \prod_{i=1}^{k} P(p_i \mid X = x).$$

Observation 3.2.2. *Let e_C be an expert with constant opinion p_C (i.e., $\forall s, e_C(s) = p_C$; examples include the all-generalizing expert $e_\circ$, and structure-grammar-based experts). Let and e_1 be any other expert. Then the expert opinions $e_C(s)$ and $e_1(s)$ are conditionally independent, given any structure x.*

This is true because the event $e_C(s) = p_C$ is the certain event, i.e., $P(e_C(s) = p_C) = 1$.

The following theorem is based on Morris's Bayesian analysis of expert opinions [29].

Theorem 3.2.3 (Product of Conditionally Independent Expert Opinions). *Let s be an amino acid sequence with a prior probability distribution $P(X = x)$ on the structure of s. Let $e_1(s), e_2(s), \ldots, e_k(s)$ be the opinions of k experts on the structure of s. Then if this set of expert opinions is conditionally independent, given $X = x$, the posterior probability $P(X = x \mid e_1(s), \ldots, e_k(s))$ of the structure of s is a product of the expert opinions:*

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \propto \frac{\prod_{i=1}^{k} P(x \mid e_i(s))}{(P(X = x))^{k-1}}. \tag{3.1}$$

Proof. According to Bayes's[3] Theorem,

$$\begin{aligned} P(X = x \mid e_1(s), \ldots, e_k(s)) &= \frac{P(e_1(s), \ldots, e_k(s) \mid X = x)P(X = x)}{P(e_1(s), \ldots, e_k(s))} \\ &\propto P(e_1(s), \ldots, e_k(s) \mid X = x)P(X = x). \end{aligned}$$

Now we develop the $P(e_1(s), \ldots, e_k(s) \mid X = x)$ term of the right-hand side:

$$\begin{aligned} P(e_1(s), \ldots, e_k(s) \mid X = x) &= \prod_{i=1}^{k} P(e_i(s) \mid e_{i+1}(s), \ldots, e_k(s), X = x), \\ &= \prod_{i=1}^{k} P(e_i(s) \mid X = x) \end{aligned}$$

which is due the independence requirement.

Using Bayes's Theorem again,

$$P(e_i(s) \mid X = x) = \frac{P(X = x \mid e_i(s))P(e_i(s))}{P(X = x)}.$$

Thus

$$P(e_1(s), \ldots, e_k(s) \mid X = x) = \prod_{i=1}^{k} \frac{P(X = x \mid e_i(s))P(e_i(s))}{P(X = x)}.$$

[3]Though often written *Bayes'* in the literature, correct English usage requires the extra *s*. Strunk and White: "Form the possessive singular of nouns by adding *'s*. Follow this rule whatever the final consonant." [39, Rule 1, p. 1]

Finally, we substitute this result into our original equation, obtaining

$$\begin{aligned} P(X = x \mid e_1(s), \ldots, e_k(s)) &\propto \left(\prod_{i=1}^{k} \frac{P(X = x \mid e_i(s)) P(e_i(s))}{P(X = x)} \right) P(X = x) \\ &\propto \frac{\prod_{i=1}^{k} P(X = x \mid e_i(s))}{(P(X = x))^{k-1}}, \end{aligned}$$

which is the desired result. □

Corollary 3.2.4. *If the prior distribution $P(X = x)$ is uniform, the posterior probability is proportional to the product of the expert opinions.*

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \propto \prod_{i=1}^{k} P(X = x \mid e_i(s)) \tag{3.2}$$

The distribution described by Equation (3.1) can be written

$$P(X \mid e_1(s), \ldots, e_k(s)) = \left(\bigotimes_{i=1}^{k} \big(P(X \mid e_i(s)) \oslash P(X) \big) \right) \otimes P(X), \tag{3.3}$$

where $\otimes$ is the normalized product operator and $\oslash$ is the normalized quotient operator, both defined below.

Definition 3.2.3 (Normalized Product). *The* normalized product *of k probability distributions $p_1(X), p_2(X), \ldots, p_k(X)$ is denoted $\bigotimes_{i=1}^{k} p_i(X)$ (or $p_1(X) \otimes p_2(X)$ when $k = 2$), and is the probability distribution $p_*(X)$ proportional to the product of the first k distributions:*

$$p_*(X = x) \propto \prod_{i=1}^{k} p_i(X = x) \tag{3.4}$$

$$= \frac{\prod_{i=1}^{k} p_i(X = x)}{\sum_{x' \in \mathcal{X}} \prod_{i=1}^{k} p_i(X = x)}. \tag{3.5}$$

Definition 3.2.4 (Normalized Quotient). *The* normalized quotient *of probability distributions $p_1(X)$ and $p_2(X)$ is denoted $p_1(X) \oslash p_2(X)$, and is the probability distribution $p_*(X)$ defined as*

follows.

$$p_*(X=x) \propto \begin{cases} 0 & \text{if } p_2(X=x)=0, \\ p_1(X=x)/p_2(X=x) & \text{otherwise} \end{cases}$$
$$= \begin{cases} 0 & \text{if } p_2(X=x)=0, \\ \frac{p_1(X=x)/p_2(X=x)}{\sum_{\{x' \in \mathcal{X} \mid p_1(X=x)>0\}} p_1(X=x')/p_2(X=x')} & \text{otherwise} \end{cases}.$$

The normalized product of distributions $p_1(X), \ldots, p_k(X)$ is undefined if the intersection of their languages is null. The normalized quotient of $p_1(X)$ and $p_2(X)$ is undefined if the language of p_1 is not a subset of the language of p_2.

The result of Equation (3.3) is well-defined: the terms of the normalized product are calibrated expert opinions, thus the intersection of their languages is non-null; the divisor of the normalized quotient is the prior distribution, whose language must be a superset of that of any other expert.

Note that the language of the normalized product of a set of probability distributions is the intersection of the languages of the probability distributions. We will sometimes refer to the normalized product operator as the *conjunction* operator, due to its correspondence to the AND operator in formal logic.

Conditionally dependent experts

In the case where our expert opinions are not conditionally independent, Theorem 3.2.3 does not apply, and the analysis becomes more difficult [29, 28]. We may, however, derive a similar result for a certain case of expert dependence. In general, for the conditionally dependent case, we have

$$P(e_1(s), e_2(s), \ldots, e_k(s) \mid X=x) \neq \prod_{i=1}^{k} P(e_i(s) \mid X=x).$$

We now consider the case where there exists an exponent γ such that

$$P(e_1(s), e_2(s), \ldots, e_k(s) \mid X=x) \propto \left(\prod_{i=1}^{k} P(e_i(s) \mid X=x) \right)^{\gamma}.$$

Now, as in the proof of Theorem 3.2.3, we apply Bayes's rule to obtain

$$\begin{aligned} P(X = x \mid e_1(s), \ldots, e_k(s)) &= \frac{P(e_1(s), \ldots, e_k(s) \mid X = x) P(X = x)}{P(e_1(s), \ldots, e_k(s))} \\ &= \frac{\left(\prod_{i=1}^{k} P(e_i(s) \mid X = x) \right)^{\gamma} P(X = x)}{P(e_1(s), \ldots, e_k(s))}, \end{aligned}$$

which, after a second application of Bayes's rule,

$$\begin{aligned} &= \frac{\left(\frac{\prod_{i=1}^{k} P(X=x \mid e_i(s)) P(e_i(s))}{P(X=x)} \right)^{\gamma} P(X = x)}{P(e_1(s), \ldots, e_k(s))} \\ &\propto \left(\prod_{i=1}^{k} P(X = x \mid e_i(s)^{\gamma} \right) P(X = x)^{1-k\gamma}. \end{aligned}$$

Note that this result is the same as Equation (3.1), except for the use of the exponent γ. When $\gamma = 1$, the expert opinions are conditionally independent, and the results are identical. We have thus derived a generalization of Theorem 3.2.3:

Theorem 3.2.5 (Product of Expert Opinions). *Let s be an amino acid sequence with a prior probability distribution $P(X = x)$ on the structure of s. Let $e_1(s), e_2(s), \ldots, e_k(s)$ be the opinions of k experts on the structure of s. If there exists an exponent γ, a measure of the conditional independence of these expert opinions, such that*

$$P(e_1(s), e_2(s), \ldots, e_k(s) \mid X = x) = \left(\prod_{i=1}^{k} P(e_i(s) \mid X = x) \right)^{\gamma},$$

then the posterior probability $P(X = x \mid e_1(s), \ldots, e_k(s))$ of the structure of s is a product the prior distribution and powers of the expert opinions:

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \propto \left(\prod_{i=1}^{k} P(X = x \mid e_i(s)^{\gamma} \right) P(X = x)^{1-k\gamma}. \tag{3.6}$$

A similar corollary applies here as in the conditional-independence case:

Corollary 3.2.6. *If the prior distribution $P(X = x)$ is uniform, the posterior probability is*

proportional to the product of the powers of the expert opinions.

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \propto \prod_{i=1}^{k} P(X = x \mid e_i(s))^{\gamma}. \tag{3.7}$$

The prior distribution also vanishes from the posterior distribution in the degenerate case of maximum conditional dependence:

Corollary 3.2.7. *If the expert opinions are conditionally dependent to the point where $\gamma = 1/k$, the posterior probability is proportional to the geometric mean of the expert opinions:*

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \propto \left(\prod_{i=1}^{k} P(X = x \mid e_i(s)) \right)^{1/k}.$$

Essentially, the exponent γ measures the degree of conditional dependence between expert opinions, with $\gamma = 1$ in the conditionally independent case, and $\gamma = 1/k$ in the degenerate case. (Note that γ may not be restricted to the interval $[1/k, 1]$.)

The distribution expressed in Equation (3.6) can be written

$$P(X \mid e_1(s), \ldots, e_k(s)) = \left(\bigotimes_{i=1}^{k} (P(X \mid e_i(s)) \oslash P(X)) \uparrow \gamma \right) \otimes P(X), \tag{3.8}$$

with $\otimes$ and $\oslash$ as defined previously, and where $\uparrow$ is the normalized exponentiation operator, defined below.

Definition 3.2.5 (Normalized Exponentiation). *The* normalized exponentiation *of probability distribution $p(X)$, with exponent $\gamma \geq 0$, is denoted $p(X) \uparrow \gamma$. It is defined to be*

$$\begin{aligned} p(X = x) \uparrow \gamma &\propto (p(X = x))^{\gamma} \\ &= \frac{(p(X = x))^{\gamma}}{\sum x' \in \mathcal{X} \, (p(X = x'))^{\gamma}}. \end{aligned}$$

Observation 3.2.8. *For any probability distribution $p(X)$ and real number $\gamma > 0$, the language of $p(X)$ is equal to the language of $(p(X) \uparrow \gamma)$.*

Information-theoretic justification

Kapur and Kesavan [40] offer some information-theoretic insight into Corollary 3.2.7 (the case where $\gamma = 1/k$ — high conditional dependence between expert opinions). Simply stated, they show that the probability distribution most similar to each of a set of expert opinions is the distribution obtained from the normalized geometric mean of the expert opinions.

The measure of similarity used in Kapur and Kesavan's argument is $D(p : p')$, the Kullback-Leibler measure. It is a standard information-theoretic measure of directed divergence, or cross-entropy. Informally, it measures the distance or difference from probability distribution $p(X)$ to $p'(X)$. It is defined as:

$$D(p : p') = \sum_{x \in \mathcal{X}} p(x) \ln \frac{p(x)}{p'(x)}.$$

The Kullback-Leibler measure has the property that $D(p : p') = 0$ if and only if $p = p'$. It is an asymmetric measure, and thus not a true distance metric.

Kapur and Kesavan's argument is as follows. Let $p_1, p_2, \ldots, p_k$ be opinions on the structure of s obtained from our k experts, i.e., probability distributions over $\mathcal{X}_n$. For any probability distribution p, the mean distance from p to our expert opinions is

$$\sum_{i=1}^{k} \frac{1}{k} D(p : p_i).$$

More generally, a weighted average of the distances from p to our expert opinions is

$$\sum_{i=1}^{k} \lambda_i D(p : p_i), \tag{3.9}$$

where each λ_i is a weight and $\sum_{i=1}^{k} \lambda_i = 1$.

The probability distribution p_* which minimizes the average divergence from itself to the

expert opinions is

$$p_*(X) = \operatorname*{argmin}_{p(X)} \sum_{i=1}^{k} \lambda_i D(p : p_i),$$

which reduces to

$$p_*(x) = \frac{\prod_{i=1}^{k} (p_i(X = x))^{\lambda_i}}{\sum_{x' \in \mathcal{X}} \prod_{i=1}^{k} (p_i(X = x'))^{\lambda_i}},$$

which confirms Corollary 3.2.7. Intuitively, in the extreme case of conditional dependence where we have k copies of the same expert and all opinions $p_i(X)$ are identical, the geometric mean is also identical: $p_*(X) = p_i(X)$.

Note that the directed divergence measure used here is $D(p : p_i)$, as opposed to $D(p_i : p)$. This direction ($p \to p_i$) is chosen according to Kullback's minimum cross-entropy principle, which, for cross-entropy minimization problems, requires the divergence be measured from the variable probability distribution to the fixed distribution [41]. (If the direction of the divergence measure in Equation (3.9) were reversed, i.e., $D(p_i : p)$, the resulting minimal-KL-distance probability distribution p_* would simply be the weighted arithmetic mean of the expert opinions: $p_*(x) = \sum_{i=1}^{k} \lambda_i p_i(x)$.)

3.2.4 Balanced experts

Balanced training is a means of improving the overall accuracy of protein secondary structure models [20]. If the prior distribution of protein structure, $P(X)$, is uniform, experts trained on data from this distribution are automatically balanced. The corollary to Theorem 3.2.3 shows how balanced training simplifies the calculation of the combined expert opinion.

When the prior probability distribution is non-uniform, it appears in Equation (3.1). We offer this interpretation of its appearance: if we re-write the equation as

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \propto P(X = x) \prod_{i=1}^{m} \frac{P(x \mid e_i(s))}{P(X = x)},$$

we see that the quotient $P(x \mid e_i(s))/P(X = x)$ removes the bias of the prior before the expert is combined with the others. The initial $P(X = x)$ term then restores the bias. The interpretation, then, is that the quotient $P(x \mid e_i(s))/P(X = x)$ balances the expert. Thus we have a means of balancing an expert opinion after it has been trained on biased data.

The same method of removing the bias from an expert also applies to the final probability distribution $P(X = x \mid e_1(s), \ldots, e_k(s))$, itself an expert opinion: dividing the distribution by the prior balances the distribution:

$$P(X = x \mid e_1(s), \ldots, e_k(s)) \oslash P(X = x) = \otimes_{i=1}^{m} \left(P(x \mid e_i(s)) \oslash P(X = x) \right).$$

3.3 Summary

3.3.1 Product of experts as a modeling technique

The theory introduced in this chapter suggests a strategy for creating complex models of protein secondary structure using a bottom-up, modular modeling approach:

1. Choose a prior probability distribution $P(X)$ representing the background distribution of protein structures. Carefully consider balanced training issues (Section 3.2.4).

2. Collect as much information as possible about the target protein. Section 3.2.1 lists some possible sources. Represent each unit of information as an expert opinion.

3. Assess the conditional (in)dependence of each expert opinion.

4. Combine the assembled expert opinions using Equations (3.1), (3.6), or (3.2). If the expert opinions are conditionally dependent, but the exponent γ is not known, a conservative estimate is $\gamma = 1/k$.

   This step and the previous step may be repeated in an iterative fashion with a few experts at a time, so that the consensus opinion is obtained from a hierarchical assembly of the expert opinions.

5. Make a prediction: choose an appropriate structure from $\mathcal{X}_n$ based on the resulting probability distribution $P_*(X)$ and the goals of the prediction (Section 3.1.1).

3.3.2 Choosing probabilistic models to represent expert opinion

Probability distributions can be represented in different ways. For distributions over a large sample space such as $\mathcal{X}_n$, it is not feasible to represent the distribution as a table of $(x, P(x))$ pairs. Concise representations of probability distributions are obtained with probabilistic models such as hidden Markov models and Markov chains.

In order to usefully represent expert opinions, a class $\mathcal{C}$ of probabilistic models must satisfy several conditions.

- Equations (3.3) and (3.8) require that $\mathcal{C}$ be closed under normalized product.
- If the expert opinions are conditionally dependent, Equation (3.8) requires that $\mathcal{C}$ be closed under normalized exponentiation.
- If the prior probability distribution is non-uniform, Equations (3.3) and (3.8) require that $\mathcal{C}$ be closed under normalized quotient.

Additionally, depending on the goals of the prediction, algorithms may be required to compute the following values from members of $\mathcal{C}$.

- $p(X = x)$
- $x^* = \text{argmax}_{x \in \mathcal{X}_n} p(X = x)$
- $p(X_i = \sigma)$,
- random samples from $p(X)$

Although Chapter 4 will discuss the closure of product, quotient and exponentiation for specific classes of probability distributions, one closure property applies in general:

Observation 3.3.1. *Any class of probability distributions that is closed under conjunction is also closed under normalized integer exponentiation, for positive integers.*

Proof. By induction. The base case is $p^1 = p$. For any $i > 1$, $p^i = p \otimes p^{i-1}$. □

In the next chapter, we will study the effectiveness of Markov chains and hidden Markov models as representations of expert opinions.

3.4 Future Work

Two unsolved problems are apparent in this chapter, which correspond to steps 3 and 4 of the strategy summarized in Section 3.3.1. The first is the problem of assessing conditional dependence or independence between expert opinions. The second is the problem, for the conditionally dependent case, of determining whether an exponent γ exists, and if so, estimating its value. The second problem encompasses the first problem, in that $\gamma = 1$ for the independent case.

Reasoning about these two problems should be simpler if we restrict ourselves to a particular class of experts, e.g., rational experts. Consider the case where we have two rational experts, e_1 and e_2, that partition the sequence space such that the target s falls in subsets L_1 and L_2 of $\mathcal{S}_n$, respectively. The conditional independence of $e_1(s)$ and $e_2(s)$ depends on the magnitudes $|L_1|$, $|L_2|$, $|L_1 \cap L_2|$, and $|\mathcal{S}_n|$.

Chapter 4

Using Markov Models to Represent Expert Opinion

The previous chapter developed a framework for probabilistic modeling of protein secondary structure by combining the opinions of experts. These opinions, represented by probability distributions, are combined using the operations of normalized product, quotient and exponentiation. The resulting probability distribution is used to make predictions by obtaining the probabilities $P(X = x)$, $P(X_i = a)$, and $\text{argmax}_{x \in \mathcal{X}} P(X = x)$ from the underlying probability distribution $P(X)$.

When the sample space is very large, it becomes infeasible to explicitly represent probability distributions in a naive fashion such as a table. Probabilistic models are a compact way of representing probability distributions over large sample spaces. This chapter investigates the consequences of modeling sequences by products of experts when we limit our probability distributions to those that can be represented by certain classes of probabilistic models. In particular, two related classes of probabilistic models are studied: Markov chains and hidden Markov models. To be suitable for the products of experts modeling technique, these model classes must be closed under the specified operations, and efficient algorithms must exist to extract predictions from them. As we shall demonstrate, both classes are closed under normalized product. Markov chains

are also closed under normalized quotient and exponentiation. The closure of HMMs under normalized quotient and exponentiation is conjectured. Methods are described for creating conjoined and exponentiated models. In addition, algorithms for computing the required probabilities that avoid explicit enumeration of the sequence space are given.

The probabilistic models in this chapter are all *generative models*. Generative models are abstract machines that operate by a stochastic process. They are not functions that compute particular probabilities in the distribution. The way they specify probability distributions is indirect: generative models can be *executed*. A single execution of a generative model produces a single string. The string is said to be *emitted*, or *generated* by the model. This string is created by a stochastic mechanism, arranged so that every string x is generated with probability $P(X = x)$. The generated string is sometimes referred to as the *observed* string, in contrast to the hidden inner workings of the model that produced the string.

4.1 Markov Chains

A Markov chain is a generative model. It is a probabilistic state machine with a special begin state and a special end state. Execution of a Markov chain proceeds as a series of random state transitions, beginning in the begin state and ending in the end state. At any step in its execution, the Markov chain is in a particular state. The state that the Markov chain will be in at the next step is chosen randomly, according to a probability that is conditional on the current state. The string generated by the execution of a Markov chain is the sequence of states that it visited between the begin and end states.

Since the strings generated by a Markov chain are composed of symbols representing states of the machine, we will sometimes refer to the state set of a Markov chain as its *alphabet*. Therefore we speak interchangeably of states $q \in Q$ and output symbols $\sigma \in \Sigma$. Conceptually, the Markov chain *emits* the name of the current state after every state transition.

Definition 4.1.1 (Markov chains). *Formally, a Markov chain is specified by a quadruple* $M = (Q, \mathcal{B}, \mathcal{E}, A)$*, where* Q *is a finite set of states (the alphabet),* $\mathcal{B}$ *and* $\mathcal{E}$ *are the begin and end states,*

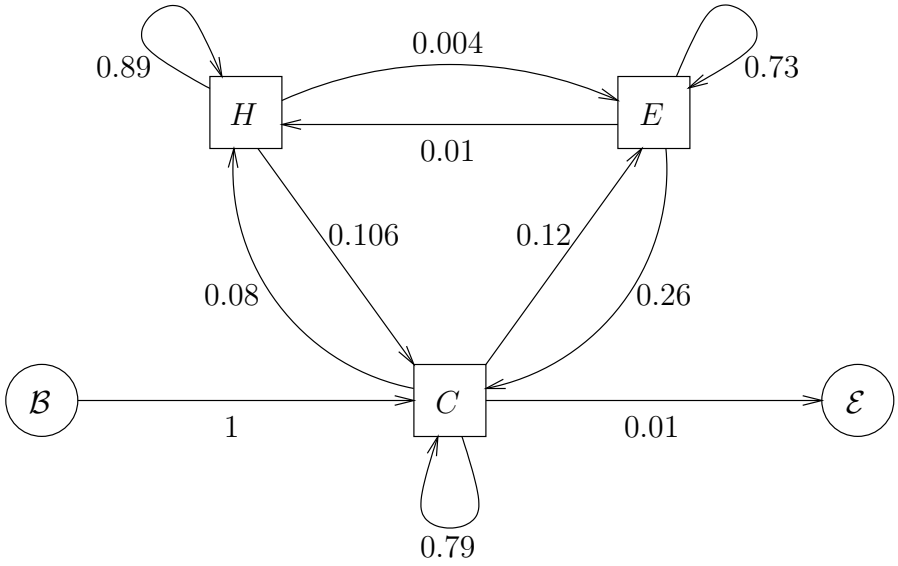

Figure 4.1: Example of a Markov chain.

and A is the transition probability table. Elements a_{qr} of A indicate the probability of making a transition from state q to state r, for $q \in Q \cup \{\mathcal{B}\}$, $r \in Q \cup \{\mathcal{E}\}$.

Markov chains always begin their execution at time t_0 in state $\mathcal{B}$. During execution, at time t_i and in state q, a Markov chain will randomly choose the state it will be in at time t_{i+1} according to the probability a_{qr}. Execution ends as soon as the Markov chain makes a transition to state $\mathcal{E}$. The sequence generated by the execution of a Markov chain is the sequence of states $\pi = \pi_1\pi_2 \ldots \pi_n$, also called the *state path* where π_i is the state of the Markov chain at time t_i, $\mathcal{B}$ is the state of the Markov chain at time t_0 and $\mathcal{E}$ is the state of the Markov chain at time t_{n+1}. Thus the probability of Markov chain generating string x of length n is

$$P(x) = a_{\mathcal{B}x_1} \left(\prod_{i=1}^{n-1} a_{x_i x_{i+1}} \right) a_{x_n \mathcal{E}}, \tag{4.1}$$

which, when we let $x_0 = \mathcal{B}$ and $x_{n+1} = \mathcal{E}$, simplifies to $\prod_{i=0}^{n} a_{x_i x_{i+1}}$.

An example Markov chain is illustrated in Figure 4.1. Call this Markov chain M. Then $M = (Q, \mathcal{B}, \mathcal{E}, A)$, where the state set $Q = \{H, E, C\}$ and the transition probability matrix is as

follows:

$$A = \begin{array}{c} \\ \mathcal{B} \\ H \\ E \\ C \end{array} \begin{array}{c} \begin{array}{cccc} H & E & C & \mathcal{E} \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0.89 & 0.004 & 0.106 & 0 \\ 0.01 & 0.73 & 0.26 & 0 \\ 0.08 & 0.12 & 0.79 & 0.01 \end{pmatrix} \end{array}$$

This Markov chain generates strings over the alphabet $\{H, E, C\}$. The zeroes in the transition probability matrix force all strings generated by M to have length at least one, and to begin and end with a C. A regular expression[1] describing the language of M (the set of strings that M can generate with non-zero probability) is thus $C\left((H + E)^* C\right)^*$.

We can use Equation (4.1) above to calculate the probability with which M generates strings. For example, the probability of generating the string $CEEC$ is

$$\begin{aligned} P(X = CEEC \mid M) &= a_{\mathcal{B}C} \cdot a_{CE} \cdot a_{EE} \cdot a_{EC} \cdot a_{C\mathcal{E}} \\ &= 1 \times 0.12 \times 0.73 \times 0.26 \times 0.01 \\ &= 0.00022776. \end{aligned}$$

4.1.1 Probabilistic queries

Computing the most probable sequence

The problem of computing a string with maximal probability of being generated by M, $x^* = \text{argmax}_{x \in Q^*} P(X = x \mid M)$, is essentially the shortest path problem from graph theory. We may regard a Markov chain as a weighted, directed graph, whose vertex set is Q, and whose edge set is all state transitions with non-zero probability. Weights on edges (q, r) are the negative logarithms of the transition probabilities, $-\ln a_{qr}$. Then x^* is the sequence of states contained in the shortest path from $\mathcal{B}$ to $\mathcal{E}$.

[1] Using the notation + for union and $*$ for Kleene closure.

Dijkstra's algorithm solves the shortest path problem, and thus may be used to compute x^*. The running time of the simplest implementation of this algorithm is $O(|Q|^2)$.

The algorithm is given below. Set $S \subseteq Q \cup \{\mathcal{B}, \mathcal{E}\}$ is the set of examined states, $prob(q)$ is the maximal probability of walking a path from $\mathcal{B}$ to q, and $prev(q)$ is the preceding state in that path.

Initialization

for each $q \in Q \cup \{\mathcal{E}\}$:

$prob(q) \leftarrow 0$

$prev(q) \leftarrow \emptyset$

$prob(\mathcal{B}) \leftarrow 1$

$S \leftarrow \emptyset$

$\bar{S} \leftarrow Q \cup \{\mathcal{B}, \mathcal{E}\}$

Iteration

while $\mathcal{E} \notin S$:

$q \leftarrow \text{argmax}_{q' \in \bar{S}}\, prob(q')$

$S \leftarrow S \cup \{q\}$

$\bar{S} \leftarrow \bar{S} \setminus \{q\}$

for each $r \in \{r' \in \bar{S} \mid a_{qr} > 0\}$:

if $prob(q) \cdot a_{qr} > prob(r)$ then

$prev(r) \leftarrow q$

$prob(r) \leftarrow prob(q) \cdot a_{qr}$

Termination

$q \leftarrow \mathcal{E}$

$x^* \leftarrow \Lambda$

until $prev(q) = \mathcal{B}$:

$x^* \leftarrow x^* q$

Computing the probability of the structure of a single residue

Section 3.1.1 in the previous chapter explained that using the value $\hat{x}$ as a prediction maximizes the expected Q_3 score. The elements $\hat{x}_i$ are defined as

$$\hat{x}_i = \underset{\sigma \in \Sigma_{SS}}{\operatorname{argmax}} P(X_i = \sigma),$$

which means that we require the ability to compute the probability of the structure of a single residue, $P(X_i = \sigma \mid M)$.

We introduce the random variable Π to represent the state path walked by a Markov chain during an execution. We also introduce the random variable Π_i to represent the ith element of Π. Since the generated string of a Markov chain is the state path that it walks, the random variables Π and X are identical, as are Π_i and X_i. Thus $P(X_i = \sigma \mid M) = P(\Pi_i = q \mid M)$.

The probabilities $P(\Pi_i = q \mid M)$ for all $q \in Q$ and for all $i \in [1..k]$ (k arbitrary) may be computed in k steps with the following dynamic programming algorithm:

Initialization

$P(\Pi_0 = \mathcal{B} \mid M) \leftarrow 1$

for each $q \in Q$, $q \neq \mathcal{B}$:

$P(\Pi_0 = q \mid M) \leftarrow 0$

Iteration

for $i \leftarrow 1, \ldots, k$:

for each $r \in Q$:

$P(\Pi_i = r \mid M) \leftarrow \sum_{q \in Q} P(\Pi_{i-1} = q \mid M) \cdot a_{qr}$

The same probabilities may be computed by taking successive powers of probability matrix A. Values a_{qr} of $A = A_1$ indicate the probability of being in state r when the previous state was q. Values a'_{qr} of $A' = A^i$ indicate the probability of being in state r when the ith last state was q.

Thus $a'_{\mathcal{B}r}$ is the probability that $\Pi_i = r$. (We assume here that A is a full $|Q \cup \{\mathcal{B}, \mathcal{E}\}| \times |Q \cup \{\mathcal{B}, \mathcal{E}\}|$ matrix.)

4.1.2 Operations on Markov chains

Product of Markov chains

Let $M_1 = (Q, \mathcal{B}, \mathcal{E}, A_1)$ and $M_2 = (Q, \mathcal{B}, \mathcal{E}, A_2)$ be two Markov chains generating strings with probability distributions $P(X \mid M_1)$ and $P(X \mid M_2)$. We are interested in creating a third Markov chain, $M_* = (Q, \mathcal{B}, \mathcal{E}, A_*)$, such that it generates strings with a probability distribution proportional to the product of the first two. For ease of discussion, we speak in terms of proportionality. The actual probability of M_* generating a string x is equal to the product of $P(X = x \mid M_1)$ and $P(X = x \mid M_2)$, times a normalizing constant such that the resulting probabilities sum to one. In other words, we want

$$\begin{aligned} P(X = x \mid M_*) &\propto P(X = x \mid M_1) \cdot P(X = x \mid M_2) \\ &= \frac{P(X = x \mid M_1) \cdot P(X = x \mid M_2)}{\sum_{x' \in Q^*} P(X = x' \mid M_1) \cdot P(X = x' \mid M_2)}. \end{aligned}$$

This is the same probability distribution as is obtained by a generative model containing M_1 and M_2, and that generates strings by repeatedly running M_1 and M_2 in parallel until both machines emit the same word x, at which point the model also emits x. It is the distribution over words x given that M_1 and M_2 have both generated the same word.

This distribution can be described by a Markov chain. In other words, there exists a transition probability matrix A_* such that $M_* = (Q, \mathcal{B}, \mathcal{E}, A_*)$ generates strings according to that distribution. Consider a situation where M_1 and M_2 are executing synchronously, and both are in some same state $q \in Q \cup \{\mathcal{B}, \mathcal{E}\}$. We ignore any previous states. Then we define $D(q)$ to be the probability that M_1 and M_2 will generate the same suffix, i.e., M_1 and M_2 will continue to make transitions to identical states until both simultaneously end in state $\mathcal{E}$. So $D(\mathcal{E}) = 1$ and $D(\mathcal{B}) = \sum_{x \in Q^*} P(X \mid M_1) P(X \mid M_2)$, and more generally, for $q \in Q \cup \{\mathcal{B}\}$, $D(q) = \sum_{r \in Q} a^1_{qr} a^2_{qr} D(r)$. (We use the notation a^1_{qr} to indicate elements of A_1, and a^2_{qr} to

indicate elements of A_2.) A_* can now be defined in terms of A_1, A_2, and the D values.

Theorem 4.1.1. *Let $M_1 = (Q, \mathcal{B}, \mathcal{E}, A_1)$ and $M_2 = (Q, \mathcal{B}, \mathcal{E}, A_2)$ be Markov chains generating strings according to the probability distributions $P(X \mid M_1)$ and $P(X \mid M_2)$. Let A_* be a transition probability matrix whose elements are defined thus:*

$$a^*_{qr} = \frac{a^1_{qr} a^2_{qr} D(r)}{\sum_{r' \in Q \cup \{\mathcal{E}\}} a^1_{qr'} a^2_{qr'} D(r')}.$$

Then the Markov chain $M_* = (Q, \mathcal{B}, \mathcal{E}, A_*)$ generates strings according to the probability distribution which is the normalized product of $P(X \mid M_1)$ and $P(X \mid M_2)$.

Proof. Consider any string $x \in Q^*$. Let its length be n. Then

$$\begin{aligned}
P(X = x \mid Q_*) &= a^*_{\mathcal{B}x_1} \left(\prod_{i=1}^{n-1} a^*_{x_i x_{i+1}} \right) a^*_{x_n \mathcal{E}} \\
&= \frac{a^1_{\mathcal{B}x_1} a^2_{\mathcal{B}x_1} D(x_1)}{\sum_{r \in Q \cup \{\mathcal{E}\}} a^1_{\mathcal{B}r} a^2_{\mathcal{B}r} D(r)} \left(\prod_{i=1}^{n-1} \frac{a^1_{x_i x_{i+1}} a^2_{x_i x_{i+1}} D(x_{i+1})}{\sum_{r \in Q \cup \{\mathcal{E}\}} a^1_{x_i r} a^2_{x_i r} D(r)} \right) \\
&\quad \cdot \frac{a^1_{x_n \mathcal{E}} a^2_{x_n \mathcal{E}} D(\mathcal{E})}{\sum_{r \in Q \cup \{\mathcal{E}\}} a^1_{x_n r} a^2_{x_n r} D(r)} \\
&= \frac{P(X = x \mid M_1) P(X = x \mid M_2) \prod_{i=1}^{n} D(x_i)}{D(\mathcal{B}) \prod_{i=1}^{n} D(x_i)} \\
&= \frac{P(X = x \mid M_1) P(X = x \mid M_2)}{\sum_{x' \in Q^*} P(X = x' \mid M_1) P(X = x' \mid M_2)},
\end{aligned}$$

which is the desired result. □

The recursive definitions of the D values form a system of $|Q| + 1$ unknowns in $|Q| + 1$ linear equations, and can therefore be solved by Gaussian elimination.

Exponentiation of a Markov chain

Let $M = (Q, \mathcal{B}, \mathcal{E}, A)$ be a Markov chain which generates strings according to a probability distribution $P(X \mid M)$. We are interested in creating a Markov chain $M_* = (Q, \mathcal{B}, \mathcal{E}, A_*)$ that generates strings x with probability proportional to some power $(P(X = x \mid M))^\gamma$. We can construct M_* using a similar technique to that used above for products.

As above, we define a set of values $D(q)$ in terms of a system of linear equations. Let $D(\mathcal{E}) = 1$, and $D(q) = \sum_{r \in Q \cup \{\mathcal{E}\}} a_{qr}^{\gamma} D(r)$ for $q \in Q \cup \{\mathcal{B}\}$. Now we define A_* in terms of A and the D values:

$$a_{qr}^* = \frac{a_{qr}^{\gamma} D(r)}{\sum_{r' \in Q \cup \{\mathcal{E}\}} a_{qr'}^{\gamma} D(r')}.$$

Consider any string $x \in Q^*$. Let its length be n. Then

$$\begin{aligned}
P(X = x \mid Q_*) &= a_{\mathcal{B}x_1}^* \left(\prod_{i=1}^{n-1} a_{x_i x_{i+1}}^* \right) a_{x_n \mathcal{E}}^* \\
&= \frac{a_{\mathcal{B}x_1}^{\gamma} D(x_1)}{\sum_{r \in Q \cup \{\mathcal{E}\}} a_{\mathcal{B}r}^{\gamma} D(r)} \left(\prod_{i=1}^{n-1} \frac{a_{x_i x_{i+1}}^{\gamma} D(x_{i+1})}{\sum_{r \in Q \cup \{\mathcal{E}\}} a_{x_i r}^{\gamma} D(r)} \right) \frac{a_{x_n \mathcal{E}}^{\gamma} D(\mathcal{E})}{\sum_{r \in Q \cup \{\mathcal{E}\}} a_{x_n r}^{\gamma} D(r)} \\
&= \frac{(P(X = x \mid M))^{\gamma} \prod_{i=1}^{n} D(x_i)}{D(\mathcal{B}) \prod_{i=1}^{n} D(x_i)} \\
&= \frac{(P(X = x \mid M))^{\gamma}}{\sum_{x' \in Q^*} (P(X = x' \mid M))^{\gamma}},
\end{aligned}$$

which is the desired result.

Quotient of Markov chains

The quotient operator for Markov chains is required if the prior is represented by a Markov chain. The operator can be implemented in terms of exponentiation, where the exponent is -1. Section 4.1.2 gives a construction algorithm for powers of Markov chains. This construction, for exponent $\gamma = -1$, is well-defined for any Markov chain whose transition probabilities satisfy the constraint $a_{qr} \neq 0$, for all $q \in Q \cup \{\mathcal{B}\}$ and $r \in Q \cup \{\mathcal{E}\}$. In other words, quotient is well-defined for those Markov chains which generate every string with non-zero probability. For such Markov chains, $M_1 \oslash M_2 = M_1 \otimes M_2^{-1}$.

4.1.3 Generating fixed-length strings with Markov chains

Ordinary Markov chains are more suitable for representing probability distributions over $\mathcal{X}$ rather than $\mathcal{X}_n$. In order to represent a probability distribution over $\mathcal{X}_n$, a generative model must assign

a zero probability to strings of length other than n. For Markov chains, this is only possible if $n \leq |Q|$, and if the model assigns a zero probability to any word containing duplicate symbols: otherwise, there must exist a loop in the state-transition topology, which implies that some strings of arbitrary length are generated with non-zero probability.

Inhomogeneous Markov chains avoid these restrictions. Inhomogeneous Markov chains are Markov chains with k transition probability matrices $A_1, \ldots, A_k$ instead of one. While walking a state path π, an inhomogeneous Markov chain chooses state π_i with probability $a^{i \bmod k}_{\pi_{i-1}\pi_i}$. In other words, the first state is chosen according to the probabilities in matrix A_1, the second state is chosen according to the probabilities in matrix A_2, and the ith state is chosen according to the probabilities in matrix $A_{i \bmod k}$. Thus inhomogeneous Markov chains are useful for modeling periodic regularities in a sequence, such as in nucleotide (genetic) sequences, where $k = 3$ [42].

To model length-n sequences, we choose $k = n$, and set the probabilities of A_1 to $a^1_{qr} = 0$ and $a^1_{q\mathcal{E}} = 1$, for all $q \in Q$ (but not $\mathcal{B}$) and all $r \in Q$. This forces the Markov chain to stop executing after the nth symbol is generated.

4.2 Hidden Markov Models

A hidden Markov model (HMM) is a generative model that adds an extra layer of randomness to a Markov chain. A HMM is composed of two components: a Markov chain and a set of emission probability distributions. Unlike a Markov chain, the output of a HMM is not a sequence of states. A HMM has an output alphabet that is separate from its state set. Execution of a HMM proceeds as in the simpler Markov chain, except that every time a state transition occurs, a symbol from the output alphabet is emitted. This symbol is chosen randomly, according to a distribution conditional on the current state. Several states may emit the same symbol from the output alphabet, and so the state sequence of a HMM's execution cannot be directly determined from the output of that execution. Thus a HMM hides the execution details of its Markov chain from the observer.

Definition 4.2.1 (Hidden Markov Models). *Formally, a HMM M is denoted by 6-tuple*

$$M = (Q, \Sigma, \mathcal{B}, \mathcal{E}, A, E),$$

where $(Q, \mathcal{B}, \mathcal{E}, A)$ are the components of a Markov chain, Σ is the output alphabet, and E is the emission probability matrix. Elements $e_q(\sigma)$ of E give the probability of emitting symbol $\sigma \in \Sigma$ while in state $q \in Q$.

To generate a string $x = x_1 x_2 \ldots x_L$, M walks a path $\pi = \pi_0 \pi_1 \ldots \pi_{L+1}$ through its state set, from the distinguished begin state $\pi_0 = \mathcal{B}$ to the distinguished end state $\pi_{L+1} = \mathcal{E}$. From state π_i, the next state π_{i+1} in the state path is chosen with probability $a_{\pi_i \pi_{i+1}}$. At each state π_i except $\mathcal{B}$ and $\mathcal{E}$, M emits symbol x_i from its alphabet, chosen with probability $e_{\pi_i}(x_i)$. Because M does not emit symbols in states $\mathcal{B}$ and $\mathcal{E}$, they are called *silent states*.

When the state path (of length n) walked by a HMM M is known, the probability that it generates a string x of the same length is simply the product of the emission probabilities of each state:

$$P(X = x | \Pi = \pi, M) = \prod_{i=1}^{n} e_{\pi_i}(x_i).$$

(As for Markov chains, the random variable Π represents the state path walked by the HMM. Note, however, that unlike Markov chains, the random variable X is distinct from Π.)

When the state path walked by the M is unknown, the probability of M generating x is the probability that it generates x by any path. Thus the overall probability of generating x is the sum of the joint probabilities of emitting x and walking path π, for all possible π:

$$\begin{aligned} P(X = x \mid M) &= \sum_{\pi \in Q^n} P(x, \pi \mid M) \\ &= \sum_{\pi \in Q^n} P(\pi \mid M) P(x \mid \pi, M) \\ &= \sum_{\pi \in Q^n} a_{\mathcal{B}\pi_1} \left(\prod_{i=1}^{n-1} a_{\pi_i \pi_{i+1}} \right) a_{\pi_n \mathcal{E}} \prod_{i=1}^{n} e_{\pi_i}(x_i). \end{aligned} \tag{4.2}$$

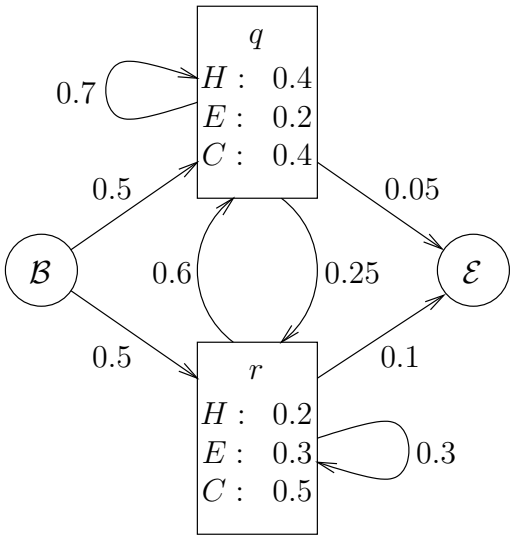

Figure 4.2: Example hidden Markov Model. Emission probabilities are indicated in each non-silent state.

An example HMM is shown in Figure 4.2. Call this HMM M. Then $M = (Q, \Sigma, \mathcal{B}, \mathcal{E}, A, E)$, where the state set is $Q = \{q_1, q_2\}$, the alphabet is $\Sigma = \{H, E, C\}$, and the transition and emission probability matrices are as follows:

$$A = \begin{array}{c} \\ \mathcal{B} \\ q \\ r \end{array} \begin{array}{c} \begin{array}{ccc} q & r & \mathcal{E} \end{array} \\ \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.7 & 0.25 & 0.05 \\ 0.6 & 0.3 & 0.1 \end{pmatrix} \end{array} \qquad E = \begin{array}{c} \\ H \\ E \\ C \end{array} \begin{array}{c} \begin{array}{cc} q & r \end{array} \\ \begin{pmatrix} 0.4 & 0.2 \\ 0.2 & 0.3 \\ 0.4 & 0.5 \end{pmatrix} \end{array}$$

Th zero $a_{\mathcal{BE}}$ transition probability forces the state paths walked by M to have length at least one (M cannot walk the null path). Thus M cannot generate the empty string. All other state paths are possible. In addition, since all emission probabilities are non-zero, all strings other than the empty string are possible. A regular expression describing the language of M is therefore $(H + E + C)(H + E + C)^*$. The absence of zeroes in the emission probability matrix implies that any state can emit any symbol from the alphabet. Thus M may generate any string of length n while walking any length-n state path.

We can use Equation (4.2) above to calculate the probability with which M generates strings. For example, the probability of generating the string HHH is

$$\begin{aligned} P(X = HHH \mid M) &= \sum_{\pi \in \{qqq, qqr, \ldots, rrr\}} a_{\mathcal{B}\pi_1} \left(\prod_{i=1}^{2} a_{\pi_i \pi_{i+1}} \right) a_{\pi_3 \mathcal{E}} \left(\prod_{i=1}^{3} e_{\pi_i}(x_i) \right) \\ &= (0.01225 \times 0.064) + (0.00875 \times 0.032) + (0.00375 \times 0.032) \\ &\quad + (0.00375 \times 0.016) + (0.0105 \times 0.032) + (0.0075 \times 0.016) \\ &\quad + (0.0045 \times 0.016) + (0.0045 \times 0.008) \\ &= 0.001808. \end{aligned}$$

The value of $P(X = x \mid M)$ may be calculated without explicit enumeration of all possible paths. This calculation is performed efficiently by the *forward* algorithm, a standard dynamic programming algorithm for HMMs:

Initialization

for each $q \in Q$:

$f_1(q) \leftarrow a_{\mathcal{B}q} e_q(x_1)$

Iteration

for $i \leftarrow 2, \ldots, n$:

for each $r \in Q$:

$f_i(r) \leftarrow \sum_{q \in Q} f_{i-1}(q) a_{qr} e_r(x_i)$

Termination

$f_{n+1}(\mathcal{E}) \leftarrow \sum_{q \in Q} f_n(q) a_{q\mathcal{E}}$

$P(X = x \mid M) = f_{n+1}(\mathcal{E})$

To illustrate the working of this algorithm, we will now use this algorithm to calculate $P(X = HHH \mid M)$ on our example HMM M. The first step is the initialization step, which computes

the following values:

$$f_1(q) = a_{\mathcal{B}q}e_q(H) = 0.5 \times 0.4 = 0.2$$
$$f_1(r) = a_{\mathcal{B}r}e_r(H) = 0.5 \times 0.2 = 0.1.$$

Next is the iteration step. For iteration $i = 2$, we compute

$$f_2(q) = f_1(q)a_{qq}e_q(H) + f_1(r)a_{rq}e_q(H) = (0.2 \times 0.7 \times 0.4) + (0.1 \times 0.6 \times 0.4) = 0.08$$
$$f_2(r) = f_1(q)a_{qr}e_r(H) + f_1(r)a_{rr}e_r(H) = (0.2 \times 0.25 \times 0.2) + (0.1 \times 0.3 \times 0.2) = 0.016.$$

For iteration $i = 3$, we compute

$$f_3(q) = f_2(q)a_{qq}e_q(H) + f_2(r)a_{rq}e_q(H) = (0.08 \times 0.7 \times 0.4) + (0.016 \times 0.6 \times 0.4) = 0.02624$$
$$f_3(r) = f_2(q)a_{qr}e_r(H) + f_2(r)a_{rr}e_r(H) = (0.08 \times 0.25 \times 0.2) + (0.016 \times 0.3 \times 0.2) = 0.00496.$$

This ends the iteration step. Last is the termination step. We compute

$$f_4(\mathcal{E}) = f_3(q)a_{q\mathcal{E}} + f_3(r)a_{r\mathcal{E}} = (0.02624 \times 0.05) + (0.00496 \times 0.1) = 0.001808.$$

The final answer is $P(X = x \mid M) = f_4(\mathcal{E}) = 0.001808$, which is the same result we obtained in the direct calculation of $P(X = x \mid M)$.

4.2.1 Non-terminal silent states

The canonical form of HMMs allows for exactly two silent states: the begin and end states. However, many formulations of HMMs in the literature allow for silent states other than the begin and end states. These nonterminal silent states do not increase the power of HMMs, since for every HMM with extra silent states, there exists an equivalent HMM with none. Consider a HMM $M = (Q, \Sigma, \mathcal{B}, \mathcal{E}, A, E)$, with nonterminal silent state set $S \subset Q$. M may be modified to

produce an equivalent HMM M' without nonterminal silent states:

$$M' = (Q - S, \Sigma, \mathcal{B}, \mathcal{E}, A', E),$$

where

$$a'_{qr} = \sum_{\pi \in \{q\}S^*\{r\}} \prod_{i=0}^{|\pi|-1} a_{\pi_i \pi_{i+1}}.$$

In other words, each transition probability a_{qr} is replaced by the sum of the probabilities of all paths from state q to state r having only silent states in between. Figure 4.3(a) shows an example HMM with a silent state. Figure 4.3(b) shows the equivalent HMM after silent states have been removed.

4.2.2 Probabilistic queries

Computing the most probable sequence

Computing the most probable generated string x^* generated by a HMM is more difficult than the most probable sequence for Markov chains, since every string can be generated by multiple (or indeed all) state paths through the HMM.

A related problem is the Most Probable Labeling problem for HMMs. The Most Probable Labeling problem is a posterior decoding problem (these are discussed further in Section 4.2.4). We are given a HMM M, a string x, and a labeling function $g : Q \to \Sigma_g$ which maps states in M to labels in Σ_g. The labeling of an entire state path π is simply the concatenation of the labels of each state in the path: $g(\pi) = g(\pi_1)g(\pi_2)\cdots$. The Most Probable Labeling problem asks, given that HMM M generated string x, what is the most probable labeling $g(\pi)$ of the unknown state path π that M walked while generating x?

The two problems are related as follows:

Theorem 4.2.1. *The Most Probable Generated String problem, for fixed string length, and the Most Probable Labeling problem are equivalent. That is, each problem reduces to the other.*

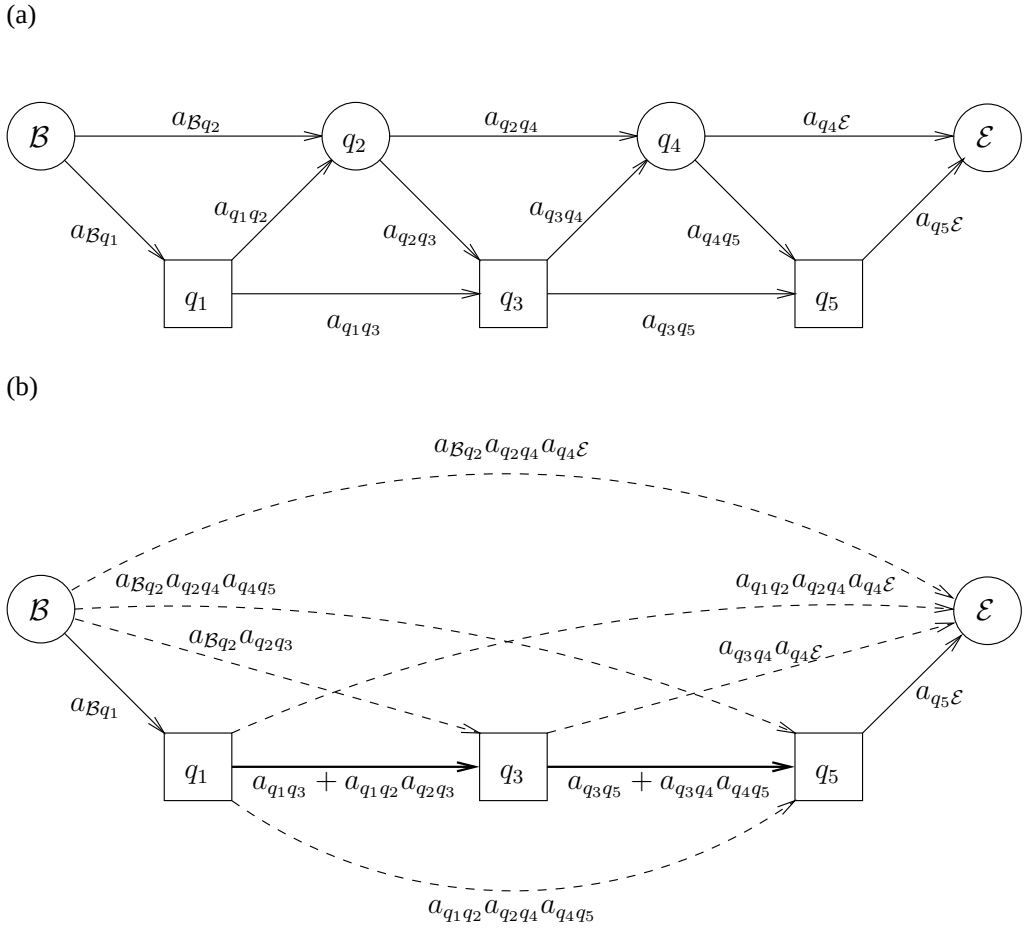

Figure 4.3: Eliminating silent states from a HMM. An example HMM is shown (a) with non-terminal silent states q_2 and q_4, and (b) after elimination of these states. Transition probabilities in (b) are written in terms of the original probabilities in (a). Note the increased probability of the transitions drawn with thick lines, and the new transitions drawn with dashed lines.

Proof. We will first reduce the Most Probable Generated String problem to the Most Probable Labeling problem. Let $M = (Q, \Sigma, \mathcal{B}, \mathcal{E}, A, E)$ be a HMM. First, create HMM $M' = (Q', \Sigma, \mathcal{B}, \mathcal{E}, A', E')$ from M, where

$$Q' = Q \times \Sigma$$

$$a'_{(q,\sigma_1)(r,\sigma_2)} = a_{qr} e_r(\sigma_2)$$

$$e'_{(q,\sigma)}(\sigma) = 1, \quad e'_{(q,\sigma)}(\sigma' \neq \sigma) = 0.$$

Then M' generates strings with the same probability as M. (The only difference is that for every state $q \in Q$ in M, there are $|\Sigma|$ corresponding states in M' — one per symbol in the alphabet. M' decides which symbol it will next emit before making the transition to the next state; M does not.)

Next, create HMM M'' from M' by reducing the alphabet to a single symbol: $\{\varsigma\}$. Let each non-silent state (q, σ) in M'' emit symbol ς with probability one. Now create a labeling function $g : Q' \to \Sigma$, defined as $g((q, \sigma)) = \sigma$.

Finally, solve the Most Probable Labeling problem for HMM M'', string ς^n, and labeling function g. The most probable labeling $y^* \in \Sigma^n$ of M'' is the most probable string x^* of length n generated by M. This completes the first reduction.

We will now reduce the Most Probable Labeling problem to the Most Probable Generated String problem. This reduction requires the construction in the proof of Theorem 4.2.4, presented below in Section 4.2.4. Let M be a HMM, x be a string, and g be a labeling function. Let the posterior probability of M walking a path π, given a generated string x, be $P(\Pi = \pi \mid X = x, M)$.

First, use the construction in the proof of Theorem 4.2.4 to create a HMM M' from M such that it *generates* state paths from M with probability $P(\Pi = \pi \mid X = x, M)$.

Next, from M', create a HMM M'' that emits label symbols $g(q)$ instead of state names q. Now solve the Most Probable Generated String problem for HMM M''. The most probable string x^* generated by M'' is the most probable labeling y^* of state paths walked by M. This completes the second reduction. □

Krogh asserts that "there is no exact algorithm for finding the most probable labeling" of a HMM, and offers an algorithm that approximates the solution [43]. Since there is no exact algorithm for finding the most probable labeling of a HMM, there cannot be an exact algorithm for finding the most probable generated sequence of a HMM.

Consequently, HMMs are not suitable expert opinions for modeling protein secondary structure when the goal of prediction is to maximize the chance of making a perfect prediction, unless an approximation is acceptable.

Computing the probability of the structure of a single residue

To determine the probability $P(X_i = \sigma \mid M)$, we first need to calculate $P(\Pi_i = q \mid M)$, for all states $q \in Q$, since any state may potentially emit the symbol σ at position i in the generated word. Thus, the first step is to use the algorithm described in Section 4.1.1 to compute $P(\Pi_i \mid M)$. Then it remains to calculate

$$P(X_i = \sigma \mid M) = \sum_{q \in Q} P(\Pi_i = q \mid M) \cdot e_q(\sigma).$$

4.2.3 Operations on hidden Markov models

Product of hidden Markov models

Let $M_1 = (Q_1, \Sigma, \mathcal{B}_1, \mathcal{E}_1, A_1, E_1)$ and $M_2 = (Q_2, \Sigma, \mathcal{B}_2, \mathcal{E}_2, A_2, E_2)$ be two HMMs generating strings with probability distributions $P(X = x \mid M_1)$ and $P(X = x \mid M_2)$. We are interested in creating a third HMM, $M_* = (Q_*, \Sigma, \mathcal{B}_*, \mathcal{E}_*, A_*, E_*)$, such that it generates strings with a probability distribution proportional to the product of the first two, i.e., the normalized product, or conjunction, of $P(X \mid M_1)$ and $P(X \mid M_2)$. We will call a HMM that generates strings according to this distribution a *conjoined* HMM.

As when we studied the products of Markov chains, it is helpful to keep in mind the idea of two machines running in parallel, synchronously.

To model the simultaneous generation of the same sequence by both M_1 and M_2, a conjoined HMM keeps track of the current state of both machines. If Q_1 and Q_2 are the state sets of M_1 and

M_2, the state set of the conjoined HMM is set of all pairs ($q_1 \in Q_1, q_2 \in Q_2$), i.e. the Cartesian product $Q_1 \times Q_2$. Since M_1 and M_2 are constrained to generate the same sequence, the start state of the conjoined HMM is the pair $(\mathcal{B}_1, \mathcal{B}_2)$ and the end state is the pair $(\mathcal{E}_1, \mathcal{E}_2)$. Any states visited in between the start and end state by the conjoined HMM are pairs of non-terminal, non-silent states from Q_1 and Q_2.

Let (q_1, q_2) be such a state. When the conjoined HMM emits a symbol $\sigma \in \Sigma$ in that state, it models the simultaneous emission of σ by M_1 in state q_1 and M_2 in state q_2. The probability that both M_1 in state q_1 and M_2 in state q_2 emit the same symbol $\sigma \in \Sigma$ is simply the product of their emission probabilities: $e^1_{q_1}(\sigma)e^2_{q_2}(\sigma)$. However, since both machines are constrained to generate the same string, we condition the emission of a on the event that both machines simultaneously emit *any* same symbol. Thus we write

$$e^*_{(q_1,q_2)}(\sigma) = \frac{e^1_{q_1}(\sigma)e^2_{q_2}(\sigma)}{d(q_1, q_2)},$$

where $d(q_1, q_2) = \sum_{\sigma' \in \Sigma} e^1_{q_1}(\sigma')e^2_{q_2}(\sigma')$, the probability that M_1 in state q_1 and M_2 in state q_2 emit the same symbol.

Definition 4.2.2. *Consider the situation where M_1 and M_2 are executing synchronously, and are currently in states $q_1 \in Q_1$ and $q_2 \in Q_2$, respectively. We ignore any emissions made by M_1 or M_2 prior to their current states. Then $D(q_1, q_2)$ is the probability that M_1 and M_2 will both generate the same suffix and will both reach their end states at the same time step, i.e., both will emit the same symbol after every state transition until they both make transitions to their end states.*

Note that $D(\mathcal{E}_1, \mathcal{E}_2) = 1$, since in that case M_1 and M_2 will each generate Λ (the empty word) with probability 1. For all other pairs, $D(q_1, q_2)$ can be expressed recursively:

$$D(q_1, q_2) = d(q_1, q_2) \sum_{\substack{r_1 \in Q_1 \cup \{\mathcal{B}_1\} \\ r_2 \in Q_2 \cup \{\mathcal{B}_2\}}} a^1_{q_1 r_1} a^2_{q_2 r_2} D(r_1, r_2),$$

where we extend the definition of d to include $d(\mathcal{B}_1, \mathcal{B}_2) = 1$. Then the D values form a system

of $|Q_1| \cdot |Q_2|$ unknowns in as many linear equations in the general case, solvable by Gaussian elimination. In the case where the state transition graphs of M_1 and M_2 are acyclic, the $D(q_1, q_2)$ values can be solved more efficiently in breadth-first-search order, beginning with $(\mathcal{E}_1, \mathcal{E}_2)$ and tracing backwards along the transition arcs.

Now we can formally define HMM conjunction.

Definition 4.2.3. *Let $M_1 = (Q_1, \Sigma, \mathcal{B}_1, \mathcal{E}_1, A_1, E_1)$ and $M_2 = (Q_2, \Sigma, \mathcal{B}_2, \mathcal{E}_2, A_2, E_2)$ be two hidden Markov models. Then the conjoined HMM, $M_* = M_1 \otimes M_2$ is defined as*

$$M_* = (Q_*, \Sigma, \mathcal{B}_*, \mathcal{E}_*, A_*, E_*),$$

where $Q_* = Q_1 \times Q_2$, $\mathcal{B}_* = (\mathcal{B}_1, \mathcal{B}_2)$, $\mathcal{E}_* = (\mathcal{E}_1, \mathcal{E}_2)$, and the elements of A_* and E_* are

$$\begin{aligned} a^*_{(q_1,q_2)(r_1,r_2)} &= \frac{a^1_{q_1 r_1} a^2_{q_2 r_2} D(r_1, r_2)}{\sum_{(r'_1, r'_2) \in Q_*} a^1_{q_1 r'_1} a^2_{q_2 r'_2} D(r'_1, r'_2)} \\ &= \begin{cases} a^1_{\mathcal{B}_1 r_1} a^2_{\mathcal{B}_2 r_2} \frac{D(r_1,r_2)}{\sum_{x'} P_1(x') P_2(x')} & \text{when } (q_1, q_2) = (\mathcal{B}_1, \mathcal{B}_2), \\ d(q_1, q_2) a^1_{q_1 r_1} a^2_{q_2 r_2} \frac{D(r_1,r_2)}{D(q_1,q_2)} & \text{otherwise.} \end{cases} \\ e^*_{(q_1,q_2)}(b) &= \frac{e^1_{q_1}(b) e^2_{q_2}(b)}{d(q_1, q_2)}. \end{aligned}$$

Theorem 4.2.2. *Let M_1 and M_2 be hidden Markov models generating strings over Σ with probability distributions $P_1(x) = P(X = x \mid M_1)$ and $P_2(x) = P(X = x \mid M_2)$, respectively. Let $M_* = M_1 \otimes M_2$ with probability distribution $P_*(x) = P(X = x \mid M_*)$. Then $P_*(x) \propto P_1(x) \cdot P_2(x) = \frac{P_1(x) P_2(x)}{\sum_{x' \in \Sigma^*} P_1(x') \cdot P_2(x')}$.*

Proof. Let $L = |x|$.

$$\begin{aligned}
P_*(x) &= \sum_{\pi} P_*(x, \pi) \\
&= \sum_{\pi \in \{\mathcal{B}_*\} Q_*^L \{\mathcal{E}_*\}} a^*_{\mathcal{B}_* \pi_1} \prod_{i=1}^{L} e^*_{\pi_i}(x_i) a^*_{\pi_i \pi_{i+1}} \\
&= \sum_{\substack{\rho \in \{\mathcal{B}_1\} Q_1^L \{\mathcal{E}_1\} \\ \tau \in \{\mathcal{B}_2\} Q_2^L \{\mathcal{E}_2\}}} \frac{a^1_{\mathcal{B}_1 \rho_1} a^2_{\mathcal{B}_2 \tau_1} D(\rho_i, \tau_i)}{\sum_{x'} P_1(x') P_2(x')} \prod_{i=1}^{L} \frac{e^1_{\rho_i}(x_i) e^2_{\tau_i}(x_i)}{d(\rho_i, \tau_i)} a^1_{\rho_i \rho_{i+1}} a^2_{\tau_i \tau_{i+1}} d(\rho_i, \tau_i) \frac{D(\rho_{i+1}, \tau_{i+1})}{D(\rho_i, \tau_i)} \\
&= \sum_{\substack{\rho \in \{\mathcal{B}_1\} Q_1^L \{\mathcal{E}_1\} \\ \tau \in \{\mathcal{B}_2\} Q_2^L \{\mathcal{E}_2\}}} \frac{P_1(x, \rho) P_2(x, \tau) D(\mathcal{E}_1, \mathcal{E}_2)}{\sum_{x'} P_1(x') P_2(x')} \\
&= \frac{P_1(x) P_2(x)}{\sum_{x'} P_1(x') P_2(x')}
\end{aligned}$$

□

Theorem 4.2.3. *Let $p_1(X)$ and $p_2(X)$ be probability distributions generated by two HMMs, M_1 and M_2, whose state sets have size m and n, respectively. Then the worst-case minimum size of a HMM M_* generating the probability distribution $p_*(X) = p_1(X) \otimes p_2(X)$ is exactly mn states.*

Proof. Let N be the worst-case minimum size of M_*. If we use the construction algorithm for conjoined HMMs to create M_*, it will have exactly mn states; thus the $N \leq mn$. We need to show there exist cases where $N = mn$. The following is such a case.

Let m, n be relatively prime. Let M_1 be a HMM with m states that generates exactly those strings whose length is a multiple of m (see Figure 4.4).

Let M_2 be a HMM with n states that generates exactly those strings whose length is a multiple of n (see Figure 4.5).

Then M_* is a HMM with mn states that generates exactly those strings whose length is a multiple of mn (see Figure 4.6).

Further, M_* has the fewest number of states of any HMM that generates p_*. This is true because the shortest path from $\mathcal{B}$ to $\mathcal{E}$ in any HMM is the length of the shortest string that the HMM can generate. Any HMM M' with fewer than mn states can generate a string with length

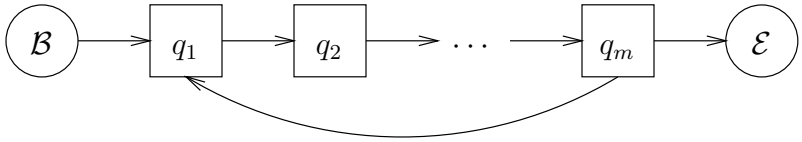

Figure 4.4: A HMM generating strings whose lengths are multiples of m.

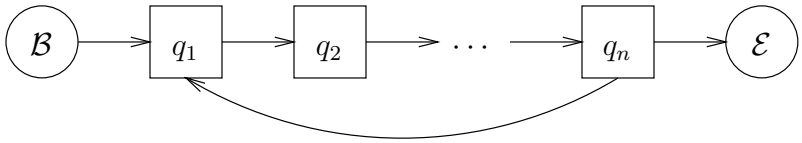

Figure 4.5: A HMM generating strings whose lengths are multiples of n.

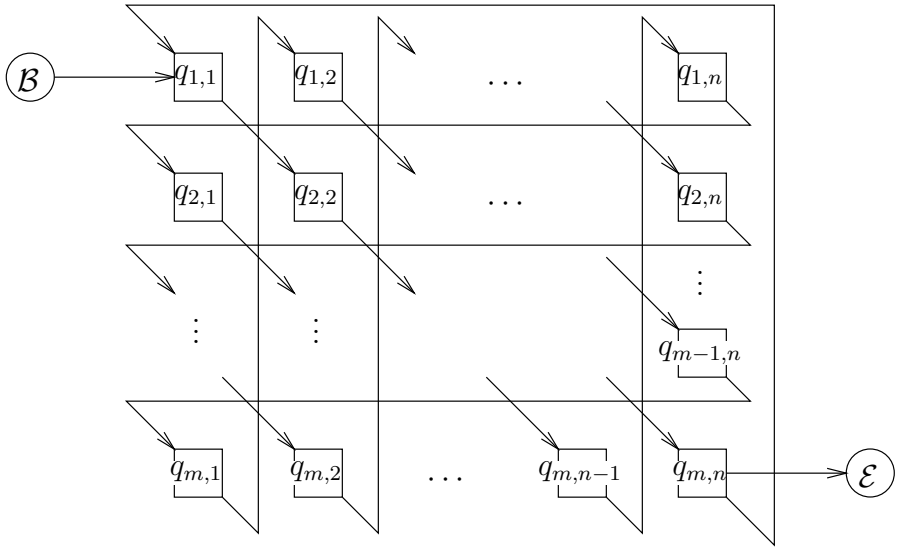

Figure 4.6: A HMM generating strings whose lengths are multiples of mn.

$< mn$; thus $L(M') \neq L(M_*)$.

Therefore there exist cases where $N = mn$, and so the worst-case minimum size of M_* is mn. □

Exponentiation of a hidden Markov model

It is unknown whether HMMs are closed under exponentiation. We conjecture that for any HMM M and exponent $\gamma > 0$, there exists an HMM M' with the same topology as M such that $P(X = x \mid M') = P(X = x \mid M) \uparrow \gamma$. In other words, we conjecture that the class of probability distributions generated by HMMs is closed under positive, normalized exponentiation, and further, that exponentiation of a HMM does not increase the state set size.

In absence of a method for computing the exponentiation of a hidden Markov model, HMMs are not suitable for representing expert opinions which are conditionally dependent on other expert opinions (represented as HMMs or otherwise).

Quotient of a hidden Markov model

It is unknown whether HMMs are closed under quotient.

In absence of a method for computing the quotient of a hidden Markov model, HMMs are not suitable for representing the prior distribution $P(X)$ (see Equation(3.8)). A Markov chain may be used instead.

4.2.4 Posterior probabilities of hidden Markov models

Posterior decoding

During the execution of a hidden Markov model, it generates two sequences: the state path π, generated by a Markov process, and the observed or emitted sequence x, based on the state path. So the outcome of the execution of a hidden Markov model is a joint event, described by two random variables, Π representing the state path and X representing the emitted sequence. The probability distribution over both is the joint probability distribution $p(\Pi = \pi, X = x)$. However,

the state path is ordinarily kept hidden (hence the name *hidden* Markov model). Thus, ordinarily, we speak only of the marginal probability distribution $p(X = x)$, and call x the generated sequence.

However, when the observed sequence is known and we wish to obtain information about the state path, we are dealing with the posterior distribution $p(\Pi \mid X = x)$. Answering probabilistic queries concerning the posterior distribution is called *decoding* in the speech recognition literature, and is the central use of HMMs in computational biology [42].

When HMMs are used for posterior decoding, the states of the HMM are meaningful. In speech recognition, the states represent phonemes of words; the symbols in the emitted sequence represent the perceived sounds. The perceived sounds are decoded in order to determine their meaning — the phonemes of the words the speaker intended to convey. Likewise, in the protein secondary structure prediction experiments of Asai et al. [24], the states represent units of secondary structure; the symbols in the emitted sequence represent the amino acids of the protein. The amino acid sequence is decoded in order to determine its meaning, in a sense — the secondary structure into which the sequence folds.

Posterior decoding can take several different forms. The simplest form of posterior decoding is determining the most likely state path walked by the HMM in order to generate x. A second posterior decoding problem is that of determining the most likely state at a particular point π_i in the state path walked by the HMM in order to generate x.

A variation of these two decoding problems is when it is not the states themselves which are important, but rather a label $G(q)$ applied to states q of the HMM. For example, in the Asai et al. experiments [24], the various states were labeled to indicate whether they represent helix, sheet, turn or coil structures. A global labeling $G(\pi)$ of a state path $\pi = \pi_1\pi_2 \ldots \pi_n$ is the concatenation of the labels of each of the states in the path: $G(\pi) = G(\pi_1)G(\pi_2) \cdots G(\pi_n)$. When states are labeled, we are dealing with the posterior probability distribution over global labels of state paths, $p\left(G(\Pi) \mid x\right)$. The equivalent problem to that of determining the most likely state path is the problem of determining the most likely global labeling of state paths.

Dynamic programming algorithms exist that solve most of these posterior decoding problems. One exception is the Most Probable Labeling problem, which, as mentioned in Section 4.2.2, has

no exact solution. Most commonly used is the Viterbi algorithm, which computes the most likely state path [42].

Obtaining expert opinions from posterior probabilities

In this research, we use HMMs strictly as representations of (prior) probability distributions, i.e., as representations of expert opinions. As such, we are not concerned with the hidden states of HMMs, or with decoding the state paths walked by these machines. Our concern is only in the probability distributions represented by the models.

However, it is interesting to note that such posterior decoding problems can be turned into equivalent probability queries on another HMM. That is, for any HMM M with state set Q and a given sequence x emitted from M, there exists a HMM M' with alphabet Q which generates the state paths of M' as its emitted sequences. The decoding problem on M may then be rephrased as an equivalent probability query on M'.

Theorem 4.2.4. *Let M be a hidden Markov model with state set Q. Let x be a string generated by M, and let $p(\Pi \mid x)$ be the posterior probability distribution of state paths walked by M to generate x. Then there exists a HMM M' whose* alphabet *is Q and that generates* emitted *strings π with probability $p'(\pi)$ the same as the posterior distribution $p(\Pi = \pi \mid x)$ of M.*

Proof. To prove this theorem, we show how to construct the HMM M'. Let $M = (Q, \Sigma, \mathcal{B}, \mathcal{E}, A, E)$ and $n = |x|$. Create a HMM M_x with deterministic state transitions and emissions that generates the string x with probability one, i.e., a chain with n states whose ith state always emits x_i:

$$e_i^x(\sigma) = \begin{cases} 1 & \text{for } \sigma = x_i \\ 0 & \text{otherwise.} \end{cases}$$

Now create a HMM M_* by conjoining M and M_x: $M_* = M \otimes M_x$, according to Definition 4.2.3. (Label the n states of M_x with the integers $1, 2, \ldots, n$.) The result is a HMM with $|Q| \cdot n$ non-terminal states. States of M_* are of the form $(q_i, j) \in Q \times [1..n]$. Because of the deterministic state transitions of M_x, the state set of M_* can be partitioned into n subsets, where the jth subset is of

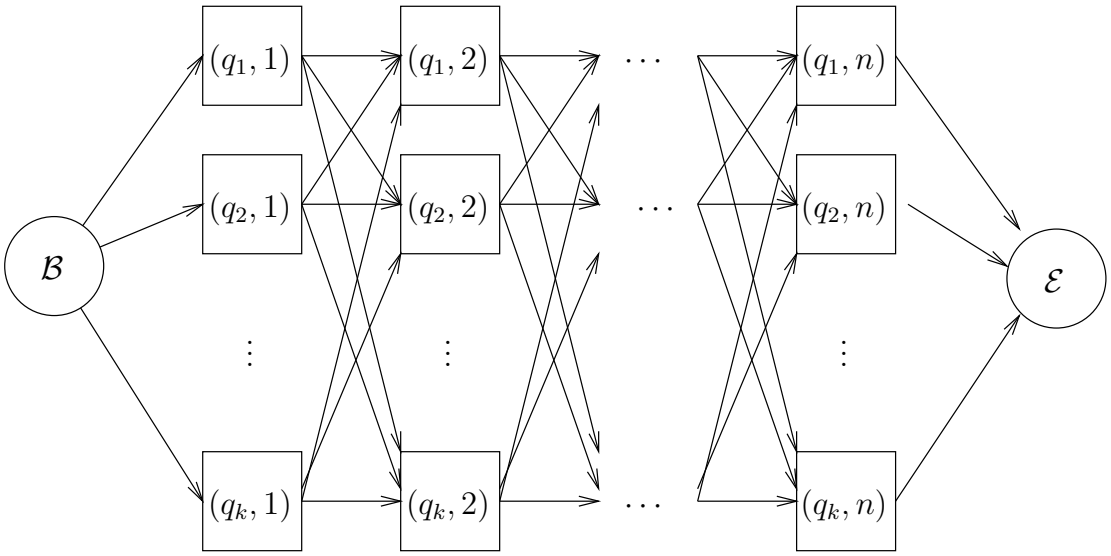

Figure 4.7: A HMM resulting from the conjunction of a HMM (with states $q_1, q_2, \ldots, q_k$) with a HMM with chain topology (with states $1, 2, \ldots, n$). Note how the transitions divide the state set into n subsets.

the form (q_i, j). All transitions from the begin state lead to the first subset. All transitions from states in the jth subset lead to the $j+1$st subset. All transitions from states in the nth subset lead to the end state. Because the deterministic emissions of M_x, all states in the jth subset emit the symbol x_j with probability one. The topology of M_* is illustrated in Figure 4.7.

Like M_x, M_* can only generate one word, x. Like M, however, M_* can walk any one of many possible state paths while generating x. A state path $q_{i_1} q_{i_2} \ldots q_{i_n}$ walked by M while generating x corresponds to the state path $(q_{i_1}, 1)(q_{i_2}, 2) \ldots (q_{i_n}, n)$ walked by M_*. Further, the posterior probabilities of walking corresponding paths in M and M_* are the same, i.e., $p(\pi \mid x) = p_*(\pi_* \mid x)$, where $p_*(\pi_* \mid x)$ is the posterior probability that M_* walks path π_*, given that it generated sequence x. However, since $p_*(x) = 1$, the posterior probability of M_* walking path π_* is the same as the prior probability of walking π_*: $p_*(\pi_* \mid x) = p_*(\Pi_* = \pi_*)$.

Now create a HMM M' by modifying M_* so that instead of emitting x_j in state (q_i, j), it emits

the *state* q_i of M that would have emitted x_j. Formally, $M' = (Q', \Sigma', \mathcal{B}', \mathcal{E}', A', E')$, where

$$Q' = Q \times [1..n] \quad \text{(same as } M_*\text{)} \tag{4.3}$$

$$\Sigma' = Q \quad \text{(i.e., the symbols emitted by } M' \text{ are the states of } M\text{)} \tag{4.4}$$

$$A' = A_*, \quad \text{(same as } M_*\text{)} \tag{4.5}$$

$$e'_{(q_i,j)}(q) = \begin{cases} 1 & \text{if } q = q_i \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Now M' generates the possible state paths of M as its *observed* sequences. Further, M' emits sequences $\pi \in Q^*$ with probability $p'(X' = \pi)$ equal to $p(\Pi = \pi \mid X = x)$, the posterior probability distribution of M. □

4.3 Future Work

The question of the closure of hidden Markov models under normalized quotient and exponentiation operations is unsolved. Future work should resolve these closure properties, and provide construction algorithms for those operations which are closed.

Another direction for future work is the study of more complex probabilistic models as representations of expert opinion. There exist classes of probabilistic models which mirror the Chomsky hierarchy of formal language classes. Hidden Markov models correspond to the class of regular languages. The next step up in the hierarchy is the set of stochastic context-free grammars (SCFGs), corresponding to Chomsky's context-free languages. However, it is immediately apparent that SCFGs are not suitable representations of expert opinion:

Theorem 4.3.1. *The class of stochastic context-free grammars is not closed under normalized product.*

Proof. The class of context-free languages is not closed under intersection. This is a well-known result; the classic counterexample is the pair of context-free languages $L_1 = \{a^i b^i c^j \mid i > 0, j > 0\}$ and $L_2 = \{a^i b^j c^j \mid i > 0, j > 0\}$. The normalized product of two probability distributions is the

probabilistic equivalent of the intersection of two languages: $L(p_1 \otimes p_2) = L(p_1) \cap L(p_2)$. Thus the SCFGs cannot be closed under normalized product. □

Bibliography

[1] Carl Branden and John Tooze. *Introduction to Protein Structure*. Garland Publishing, New York, 1991.

[2] Christian B. Anfinsen. Principles that govern the folding of protein chains. *Journal of Molecular Biology*, 181(4096):223–230, 1973.

[3] Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.

[4] Robert D. Smith. *Correlations between Bound N-Alkyl Isocyanide Orientations and Pathways for Ligand Binding in Recombinant Myoglobins*. PhD thesis, Rice University, 1999. Source molecule for my example of an alpha helix. PDB ID 101M.

[5] Z.F. Kanyo, K.M. Pan, R.A. Williamson, D.R. Burton, S.B. Prusiner, R.J. Fletterick, and F.E. Cohen. Antibody binding defines a structure for an epitope that participates in the prpc→prpsc conformational change. *Journal of Molecular Biology*, 293(4):855–863, 1999. Source molecule for my example of an antiparallel beta sheet. PDB ID 1CR9.

[6] K.F. Ahmad, C.K. Engel, and G.G. Prive. Crystal structure of the btb domain from plzf. *Proceedings of the National Academy of Sciences of the United States of America*, 95:12123–, 1998.

[7] Hue Sun Chan and Ken A. Dill. The protein folding problem. *Physics Today*, pages 24–32, February 1993.

[8] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[9] Daniel Fischer and David Eisenberg. Protein fold recognition using sequence-derived predictions. *Protein Science*, 5(5):947–955, May 1996.

[10] Cheng Che Chen, Jaswinder Pal Singh, and Russ B. Altman. Using imperfect secondary structure predictions to improve molecular structure computations. *Bioinformatics*, 15(1):53–65, 1999.

[11] Adam Zemla, Česlovas Venclovas, Krzysztof Fidelis, and Burkhard Rost. A modified definition of Sov, a seqment-based measure for protein secondary structure prediction assessment. *Proteins: Structure, Function, and Genetics*, 34:220–223, 1999.

[12] Burkhard Rost, Chris Sander, and Reinhard Schneider. Redefining the goals of protein secondary structure prediction. *Journal of Molecular Biology*, 235:13–26, 1994.

[13] Steve A. Benner, Gina Cannarozzi, Dietlind Gerloff, Marcel Turcotte, and Gareth Chelvanayagam. Bone fide predictions of protein secondary structure using transparent analyses of multiple sequence alignments. *Chemical Reviews*, 97:2725–2843, 1997.

[14] Peter Y. Chou and Gerald D. Fasman. Prediction of protein conformation. *Biochemistry*, 13(2):222–245, 1974.

[15] Peter Y. Chou and Gerald D. Fasman. Prediction of the secondary structure of proteins from their amino acid sequence. In Alton Meister, editor, *Advances in Enzymology*, volume 47, pages 45–148. John Wiley & Sons, New York, 1978.

[16] Jonathan M. Levin, Barry Robson, and Jean Garnier. An algorithm for secondary structure determination in proteins based on sequence similarity. *FEBS Letters*, 205(2):303–308, 1986.

[17] Ning Qian and Terrence J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884, 1988.

[18] Burkhard Rost. PHD: Predicting 1D protein structure by profile based neural networks. *Methods in Enzymology*, 266:525–539, 1996.

[19] David T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292, 1999.

[20] Burkhard Rost and Chris Sander. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 90:7558–7562, aug 1993.

[21] Dmitrij Frishman and Patrick Argos. Incorporation of non-local interactions in protein secondary structure prediction from the amino acid sequence. *Protein Engineering*, 9(2):133–142, 1996.

[22] C.A. Orengo, J.E. Bray, T. Hubbard, L. LoConte, and I. Sillitoe. Analysis and assessment of ab initio three-dimensional prediction, secondary structure, and contacts prediction. *Proteins: Structure, Function, and Genetics Supplement*, 3:149–170, 1999.

[23] Burkhard Rost and Chris Sander. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232:584–599, 1993.

[24] Kiyoshi Asai, Satoru Hayamizu, and Ken'ichi Handa. Prediction of secondary structure by the hidden Markov model. *Computer Applications in the Biosciences*, 9(2):141–146, 1993.

[25] Christopher Bystroff, Vesteinn Thorsson, and David Baker. HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, 301:173–190, 2000.

[26] Kevin Karplus, Christian Barrett, Melissa Diekhans, Leslie Grate, and Richard Hughey. Predicting protein structure using only sequence information. *Proteins: Structure, Function, and Genetics Supplement*, 3:121–125, 1999.

[27] Christian Genest and James V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–148, 1986.

[28] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, second edition, 1985.

[29] Peter A. Morris. Decision analysis expert use. *Management Science*, 20(9):1233–1241, 1974.

[30] Peter A. Morris. Combining expert judgements: a Bayesian approach. *Management Science*, 23(7):679–693, 1977.

[31] Peter A. Morris. An axiomatic approach to expert resolution. *Management Science*, 29(1):24–32, 1983.

[32] Geoffrey Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999.

[33] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Computational Neuroscience Unit, University College London, 2000.

[34] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden Markov models. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 472–478, Cambridge, Massachusetts, 1996. MIT Press.

[35] Matthew Brand. Coupled hidden Markov models for modeling interacting processes. Learning and Common Sense Technical Report 405, MIT Media Lab, November 1996.

[36] Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. Learning and Common Sense Technical Report 407, MIT Media Lab, November 1996.

[37] Isidore Rigoutsos, Yuan Gao, Aris Floratos, and Laxmi Parida. Building dictionaries of 1d and 3d motifs by mining the *Unaligned* 1d sequences of 17 archaeal and bacterial genomes. In *ISMB '99 Proceedings, Seventh International Conference on Intelligent Systems for Molecular Biology*. AAAI Press, 1999.

[38] N. Sreerama and R.W. Woody. Protein secondary structure from circular dichroism spectroscopy: Combining variable selection principle and cluster analysis with neural network, ridge regression and self-consistent methods. *Journal of Molecular Biology*, 242(4):497–507, 1994.

[39] Jr. Strunk, William and E.B. White. *The Elements of Style.* Allyn and Bacon, Boston, third edition, 1979.

[40] J.N. Kapur and H.K. Kesavan. *The Generalized Maximum Entropy Principle (with Applications).* Sandford Educational Press, Waterloo, Ontario, Canada, 1987.

[41] J.N. Kapur and H.K. Kesavan. *Entropy Optimization Principles with Applications.* Academic Press, San Diego, 1992.

[42] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[43] Anders Krogh. Two methods for improving performance of an HMM and their application for gene finding. In T. Gaasterland et al., editor, *Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 179–186. AAAI Press, 1997.

List of Notation

$[a = b]$	One if $a = b$, zero otherwise.
Λ	The empty word.
$\mathcal{S}$	The set of all possible amino acid sequences.
Σ_{AA}	The alphabet of twenty amino acids.
Σ_{SS}	The alphabet of protein secondary structure classes $\{H, E, C\}$.
Σ	An alphabet.
Σ^*	The Kleene closure of Σ, $\bigcup_{i=0}^{\infty} \Sigma^i$, i.e., the set of all words of length zero or greater made from symbols in Σ.
σ	A symbol in the alphabet Σ.
$\mathcal{X}$	The set of all possible secondary structures of a protein. Equivalent to Σ_{SS}^*.
$\mathcal{X}_n$	The set of all possible secondary structures of a length-n protein. Equivalent to Σ_{SS}^n.
x	A sequence representing an assignment of secondary structure to a protein; an element of $\mathcal{X} = \Sigma_{SS}^*$.
x_i	The ith element of x, representing the secondary structure of the ith residue of a protein; an element of Σ.
p, p', p_i	Probability distributions over the set of protein secondary structure assignments, Σ^*.

$p(X = x)$, $p(x)$ The probability that the target protein has secondary structure x.

X A random variable representing the actual secondary structure of a protein.

A The transition probability matrix of a Markov chain or hidden Markov model. Elements of A are denoted a_{qr}.

a_{qr} An element in the transition probability matrix of a Markov chain or hidden Markov model. Represents the probability of making a transition from state $q \in Q$ to state $r \in Q$.

$\mathcal{B}$ The begin state of a Markov chain or hidden Markov model.

E The emissions probability table of a hidden Markov model. Elements of E are denoted $e_q(\sigma)$.

$e_q(\sigma)$ An element in the emissions probability table E of a hidden Markov model. Represents the probability of emitting symbol σ while in state q.

$\mathcal{E}$ The end state of a Markov chain or hidden Markov model.

Π A random variable representing the state path walked by a Markov chain or hidden Markov model.

Π_i A random variable representing the ith state in a state path walked by a Markov chain or hidden Markov model.

π A particular state path of a Markov chain or hidden Markov model; an element of $\{\mathcal{B}\}Q^*\{\mathcal{E}\}$.

π_i The ith state of a state path walked by a Markov chain or hidden Markov model; an element of Q.

Q The set of states of a Markov chain or hidden Markov model, excluding the begin and end states.

Q^*	The Kleene closure of Q, $\bigcup_{i=0}^{\infty} Q^i$, i.e., the set of all non-terminal state paths of length zero or greater that a Markov chain or hidden Markov model can walk between $\mathcal{B}$ and $\mathcal{E}$.
q, r	An element of Q; a state in a Markov chain or hidden Markov model.

Glossary

alphabet A set of symbols used to construct words (strings, sequences).

backbone The chain of N–C^{α}–C' atoms and the covalent bonds that join them in a protein.

conformation Three-dimensional structure.

conjoined A generative model created from two or more other models, such that the probability distribution represented by the first model is the normalized product of the probability distributions of the others.

decoding The process of inferring, from the generated string, the hidden state path that a generative model walked in order to generate that string.

emission A string generated by a generative model, or a single symbol from that string.

execution The process by which a generative model generates a string. A single execution generates a single string.

expert A representation of incomplete knowledge about the folding function. A function that maps protein amino acid sequences to probability distributions over secondary structure.

expert opinion A probability distribution over secondary structure, representing an expert's incomplete knowledge about the structure of a specific protein.

generative model A representation of a probability distribution as an abstract, stochastic machine.

native environment The environment in which a protein exists, functions, and adopts its native conformation. Several variables define a protein's native environment: solvent (e.g., water), temperature, pH, salinity, etc.

native fold The preferred tertiary structure of a protein molecule in a particular solvent, generally considered to be the conformation minimizing the free energy of the molecule in the solvent.

observed string A string generated by a generative model.

protein-coding A gene that encodes the primary structure of a protein.

Protein Folding Problem The problem of determining the tertiary structure of a protein molecule given only its amino acid sequence.

Ramachandran angles The pair of angles, denoted (ϕ, ψ), that define the backbone conformation of a single amino acid residue. ϕ measures the angle of rotation of the N–C^{α} bond; ψ measures the angle of rotation of the C^{α}–C' bond.

rational expert A calibrated expert whose knowledge of the folding function is based on a partition of the sequence space.

secondary structure The classification of the local conformation of a protein molecule, usually into one of three groups: α helix, β strand, coil.

segment A maximal series of amino acid residues sharing the same secondary structure.

silent state A state in a hidden Markov model that does not emit symbols.

state path The sequence of states walked by a stochastic finite-state machine such as a Markov chain or a hidden Markov model.

tertiary structure The three-dimensional conformation of a protein molecule.

threading A method of solving the Protein Folding Problem by searching among a set of solved protein structures for the structure which best matches the target sequence.