Probabilistic Graphical Models and Algorithms for Protein Problems

by

Feng Jiao

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

Waterloo, Ontario, Canada, 2008

AUTHOR'S DECLARATION

Abstract

In this thesis I present research in two fields: machine learning and computational biology. First, I develop new machine learning methods for graphical models that can be applied to protein problems. Then I apply graphical model algorithms to protein problems, obtaining improvements in protein structure prediction and protein structure alignment.

First, in the machine learning work, I focus on a special kind of graphical model—conditional random fields (CRFs). Here, I present a new semi-supervised training procedure for CRFs that can be used to train sequence segmentors and labellers from a combination of labeled and unlabeled training data. Such learning algorithms can be applied to protein and gene name entity recognition problems. This work provides one of the first semi-supervised discriminative training methods for structured classification.

Second, in my computational biology work, I focus mainly on protein problems. In particular, I first propose a tree decomposition method for solving the protein structure prediction and protein structure alignment problems. In so doing, I reveal why tree decomposition is a good method for many protein problems. Then, I propose a computational framework for detection of similar structures of a target protein with sparse NMR data, which can help to predict protein structure using experimental data.

Finally, I propose a new machine learning approach—LS_Boost—to solve the protein fold recognition problem, which is one of the key steps in protein structure prediction. After a thorough comparison, the algorithm is proved to be both more accurate and more efficient than traditional z-Score method and other machine learning methods.

Acknowledgements

I would like to express my great gratitudes to those who help me to finish my degree. I am deeply indebted to my two supervisors: Dale Schuurmans and Ming Li. I feel that I was so lucky to meet two great professors and it was my honor to be their students.

It was Dale who chose me as his student and led me to the door of research, especially in the fields of machine learning. I am deeply grateful to him for his guidance, bringing me energy and giving me thoughtful insight into every research work I have done. I was so impressed and benefited much for his broad knowledge in every fields in machine learning and even in the filed of bioinformatics. He is one of best researcher in the world. And he is the most kind and nice people I have ever met in the world.

Also it was my great luck to have Ming, one of the best researcher in the world, as my co-supervisor in the field of bioinformatics fields. He gave me some great guidance in my research.

And my great thanks go to professor Jinbo Xu in Toyota Technological Institute at Chicago and professor Shaojun Wang in Wright State university. Jinbo led me to the fields of computation biology while Shaojun led me to the fields of machine learning. They are both one of the best researcher in their fields and I obtained so much from their solid knowledge and deep insight. Meanwhile we are very good friends and they also help me a lot in my personal life.

I would like to thanks for my other collaborators such as Li Cheng, Chihoon Lee, etc. It was great to work with them and published these good research papers.

I would like to thanks for the committee members who spent their precious time to review my thesis and give valuable suggestions.

Thanks for University of Waterloo and University of Alberta which provided me great academic research places and also provided me the funding for my research.

Dedication

This is dedicated to my Father and Mother.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Although machine learning is generally considered a sub-area of artificial intelligence, its methodology has also been widely used in many other areas. Recently, there have been a great number of developments in both the theory and application of machine learning techniques. Graphical models, in particular, have proved to be one of the most important machine learning approaches and have been successfully applied in many important areas of data analysis.

Another hot area of computer science research is bioinformatics, which uses computational methods to tackle important biological problems, such as biological data analysis, protein structure and function prediction, and rational drug design. Specific properties of biological data require existing machine learning methods to be extended to provide adequate capabilities for analyzing such data.

There are many unsolved problems at the core of biology, and many biological processes remain mysterious. Since we do not yet have an exact theoretical understanding of many important biological problems, machine learning methods are a crucial tool for advancing the state of our understanding in the field, given that there is an abundance of data but a lack of exact knowledge.

In fact, there have been many successful applications of machine learning methods to bioinformatics problems in the past few years. One good example is the profile Hidden

Markov Model used to advance the state of the art in protein secondary structure prediction.

Over the last few years, major achievements have been made in the area of genomics, especially in the accomplishment of the Human Genome Project. In the post-genomic era, increasing research has focused on proteins. Proteins play a central role in biology, and the understanding of protein function has become a key step toward modeling complete biological systems. It has been established that the functions of a protein are directly linked to its three-dimensional structure. Proteins sharing similar structures are more likely to have similar functions.

In this thesis I will focus on developing algorithms, especially machine learning methods to solve protein structure problems. Also I will propose some research on some machine learning problems, in particular the theory and application of graphical models.

1.2 Contributions

The main contributions of this thesis are list as follows:

I propose a tree decomposition framework to solve both protein structure prediction and protein structure alignment problems. I show that the low tree width properties of a contact graph, which is used to represent the 3-D protein structures, is especially appropriate for the tree decomposition algorithm. Compared with other algorithm, it offer an exact solution and is also very efficient.

I propose a computational framework for detection of similar structures of a target protein with sparse NMR data. Experimental results demonstrate that the model and algorithms proposed in this paper can be used for protein structure prediction even if a limited number of NMR data is available.

I propose a new machine learning approach—LS_Boost—to solve the protein fold recognition problem. I use a regression approach which is proved to be both more accurate and efficient than classification based approaches. The algorithm achieves strong sensitivity results compared to other fold recognition methods, including both machine learning methods and z-score based methods. Moreover, it is significantly more efficient for both the training and testing phases, which may allow genome-scale scale structure prediction.

In the field of machine learning, I present a new semi-supervised training procedure for conditional random fields (CRFs) that can be used to train sequence segmentors and labelers from a combination of labeled and unlabeled training data. My approach is based on extending the minimum entropy regularization framework to the structured prediction case, yielding a training objective that combines unlabeled conditional entropy with labeled conditional likelihood. I apply the new training algorithm to the problem of identifying gene and protein mentions in biological texts and show that incorporating unlabeled data improves the performance of the supervised CRFs in this case.

1.3 Organization

The rest of the thesis is organized as follows:

In Chapter 2 I will introduce some background on graphical models, especially on conditional random fields. Also I will introduce the tree decomposition algorithm, which is also called the junction tree algorithm when used in graphical models.

In Chapter 3 I will introduce some basic knowledge of protein problems, such as protein structure prediction and structure alignment.

Chapter 4 will mainly focus on machine learning and graphical problems. A new semi-supervised conditional random fields (Semi-CRFs) method is proposed. My approach is based on extending the minimum entropy regularization framework to the structured prediction case, yielding a training objective that combines unlabeled conditional entropy with labeled conditional likelihood. I will show how the derivative of the new objective can be computed from the covariance matrix of the features on the unlabeled data. Also a new efficient dynamic programming approach is proposed to efficiently compute the covariance matrix of the features.

Chapters 5, 6, 7 and 8 focus primarily on protein problems, mainly with the machine learning approach. Initially I will show how the tree decomposition algorithm is suitable to solve some protein problems. In Chapter 5 I present a tree decomposition method that can be used to efficiently solve protein threading for backbone prediction. In Chapter 6 I present a tree decomposition method for protein structure alignment.

In Chapter 7 I presented a computational framework for detection of similar structures of a target protein with sparse NMR data. The model and algorithms proposed here can be used for protein structure prediction even if a limited number of NMR data is available.

In Chapter 8, I present a new machine learning approach—LS_Boost—to solve the protein fold recognition problem. I use a regression approach which is proved to be both more accurate and efficient than classification based approaches. Also the algorithm is proved to be more superior than any other machine learning methods both on performance and efficiency.

Finally in Chapter 9, some conclusions are made, along with the proposal of future work.

Chapter 2

Background on Graphical Models

Graphical models, in particular, have proved to be one of the most important machine learning approaches that have been successfully applied in many important areas of data analysis such as computer vision, bioinformatics, information retrieval, and computational finance [80, 60]. Graphical models provide a general framework and methodology for modeling and solving problems that involve a large number of random variables that are linked in complex ways.

Graphical models can be thought of as the combination of probability theory and graph theory. A graphical model is specified, in part, by a graph where nodes denote random variables, and edges (or lack of edges) represent the conditional independence assumptions being made among the random variables. Probability theory ensures that the entire joint distribution specified by the graph structure and local parameters is consistent and provides ways to characterize the interaction between nodes. By using a graphical representation, one can easily and intuitively model a complex joint distribution over a large number of variables with a high amount of interaction among them. Many classical problems and algorithms in multivariate statistics and probability are actually just special cases of general graphical models, such as Hidden Markov Models [103], Kalman filters [64, 99], Hidden Markov Random Fields [39], mixture models [85, 116], and Conditional Random Fields [69].

2.1 Representation

Given a graph $G = (V, E)$, where V is the node set and E is the edge set, we assume that the nodes denote random variables and the edges represent conditional dependence/independence assumptions among these variables. The two most common forms of graphical models are directed and undirected, based on directed acyclic graphs and undirected graphs, respectively. Famous examples of directed graphical models include Hidden Markov models and Kalman filters. Well-known examples of undirected graphical models include Hidden Markov Random Fields and Conditional Random Fields.

2.1.1 Directed Models

In a directed graph, the edge set E consists of a set of directed edges. We assume furthermore that the directed graph is *acyclic*. Therefore the nodes can be topologically sorted, where each node has a set of parent nodes (possibly empty) that occur earlier in the ordering. This directed graph structure specifies that each variable is conditionally independent of its ancestor variables, given the values of its immediate parent variables. Figure 2.1 shows an example of a directed graphical model.

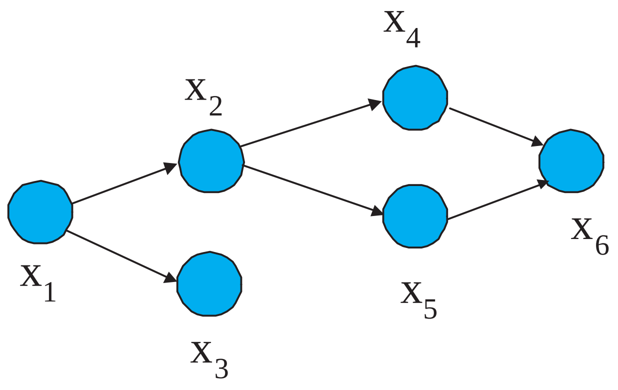

Figure 2.1: Example of a directed graph

Given a node x_i, let x_{p_i} denote its set of parent nodes. For each combination of a node and its parents we define a conditional probability distribution on x_i given each different configuration of x_{p_i}. That is, for each different value u_{p_i} of x_{p_i} we define a probability distribution over the possible values v of the child variable x_i. The joint probability distribution over the entire set of random variables, $x_1, ..., x_n$, is then defined as follows

$$p(x_1, x_2, \ldots, x_n) \;=\; \prod_{i=1}^{n} p(x_i | x_{p_i})$$

To see that this defines a valid joint distribution, we need to sum the joint probabilities over all possible joint configurations $(x_1, ..., x_n)$. For example, if we had 10 variables each with 5 possible values, a straightforward enumeration would consider 5^{10} possible combinations, which is obviously intractable in general. But by introducing the conditional independence assumptions represented by the directed acyclic graph, we effectively factorize the entire joint distribution into a product of local conditional distributions. In particular, if x_1 and x_2 are conditionally independent given another variable x_3, then the conditional probability of x_1 given both x_2 and x_3 can be reduced to

$$p(x_1 | x_2, x_3) \;=\; p(x_1 | x_2)$$

Combining this fact with the chain rule of probability, we get

$$\begin{aligned} p(x_1, x_2, ..., x_n) &= \prod_{i=1}^{n} p(x_i | x_1 ... x_{i-1}) \\ &= \prod_{i=1}^{n} p(x_i | x_{p_i}) \end{aligned}$$

Finally, to show that this results in a valid joint distribution, note that

$$\begin{aligned} \sum_{x_1, ..., x_n} p(x_1, x_2, ..., x_n) &= \sum_{x_1, ..., x_n} \prod_{i=1}^{n} p(x_i | x_{p_i}) \\ &= \sum_{x_1} p(x_1) \sum_{x_2} p(x_2 | x_{p_2}) \cdots \sum_{x_n} p(x_n | x_{p_n}) \\ &= 1 \cdot 1 \cdots 1 \end{aligned}$$

2.1.2 Undirected Models

For an undirected graphical model, we are given an edge set E that consists of a set of undirected edges. The conditional independence assumptions encoded by an undirected graph are different than those encoded by a directed graph. For an undirected graph, a set of nodes A is independent of a set of nodes B, *given* a third set of nodes C, if and only if every path between A and B in the graph goes through C. Figure 2.2 shows an example of an undirected graphical model.

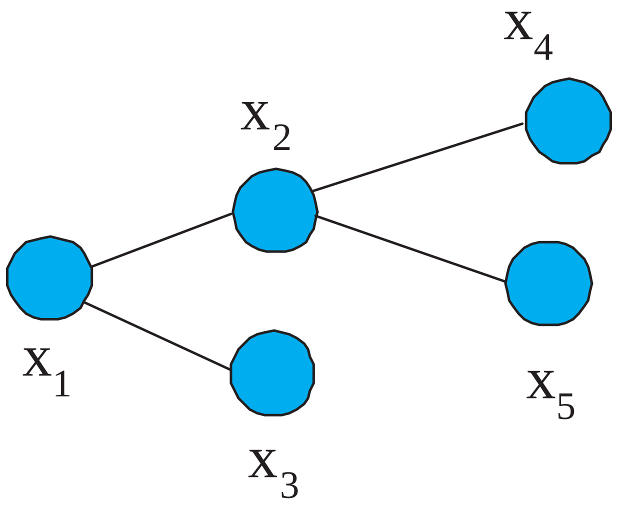

Figure 2.2: Example of an undirected graph

Let $\mathcal{C}$ denote the set of maximal cliques in the graph. We define the numerical component of an undirected graphical model by specifying nonnegative *potential* functions $\psi_C(x_C)$ over the maximal cliques $C \in \mathcal{C}$. The potential functions are assumed to be (strictly) positive, real-valued functions but are otherwise arbitrary. The joint distribution of an undirected graphical model can then be defined as:

$$p(x_1, ..., x_n) \quad = \quad \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C)$$

where Z is the normalization factor

$$Z = \sum_{x_1,\ldots,x_n} \prod_{C \in \mathcal{C}} \psi_C(x_C)$$

The most commonly used undirected graphical model is the Markov random field (MRF), used widely in computer vision and image processing. In this model, each node corresponds to a pixel and the graph is given by the 2-D lattice structure. Edges in this model connect adjacent pixels, and represent the fact that adjacent pixel values are highly correlated [100, 42].

2.2 Important Examples—Conditional Random Fields

A few specialized forms of graphical models have been prominent in various areas of research. In the past, many of the basic representations, as well as many of the basic inference and estimation algorithms, have been independently reinvented for many of these examples. The general field of graphical models has subsequently unified these algorithms in a common framework. For example, in my own research below, I will use an inference algorithm called "Tree Decomposition" that is in fact equivalent to a type of "Junction Tree Algorithm" that has been developed for graphical models. There are many kinds of graphical models which have been proved very successful such as [103], Kalman filters [64, 99], Hidden Markov Random Fields [39], mixture models [85, 116], and conditional random fields [69]. Here I will introduce conditional random fields, which have recently become very popular.

2.2.1 Definition of CRFs

Conditional random fields (CRFs) [69] are a form of undirected graphical models that defines a conditional joint distribution over label sequences y given a particular observation sequence x. Models that represent $p(y|x)$ directly are referred to as *discriminative* (as opposed to generative) models. Since CRFs model the conditional distribution directly, they demonstrate improved prediction accuracy over generative models such as HMMs, when the models are learned from data. Figure 2.3 shows one example of CRFs on chain structure. An intermediate form of the model is a *maximum entropy Markov model*

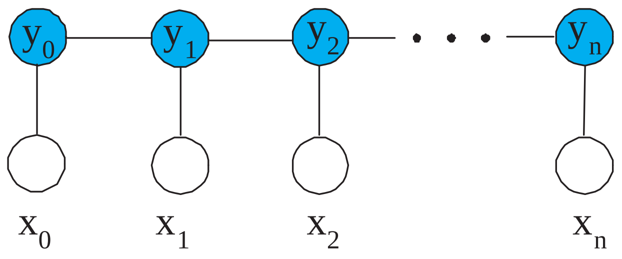

Figure 2.3: Example of a conditional random field model on a chain structure

(MEMM) [83], which represents a conditional model that is nevertheless generative on the y_t variables given the x_t observations:

$$p(y_0, ..., y_N | x_0, .., x_N) = p(y_0) \prod_{t=1}^{N} p(y_t | y_{t-1}, x_0, .., x_N)$$

CRFs go one step beyond MEMMs and use an *undirected* graphical model over $y_0, .., y_N$ given $x_0, .., x_N$. More generally, let $G = (V, E)$ be a graph such that $y = \{y_t\}_{t \in V}$, so that y is indexed by the vertices of G.[1] Then (x, y) is a conditional random field on y given x if the random variablesundirected graphical models y_t obey the Markov property with respect to the graph:

$$p(y_t | x, \{y_s\}_{s \neq t}) = p(y_t | x, \{y_s\}_{s \sim t})$$

where $s \sim t$ means that s and t are neighbors in G. The conditional distribution over $y = \{y_t\}$ given x is compactly represented by introducing *feature functions*

$$\begin{array}{rl} g_k(t, y_t, x) & \text{for} \quad t \in V \text{ and } k = 1, ..., K \\ f_\ell(s, t, y_s, y_t, x) & \text{for} \quad (s \sim t) \in E \text{ and } \ell = 1, ..., L \end{array}$$

associated with the graph. The feature functions are normally binary valued. The features f_ℓ on edges are normally referred to as transition features, and the features g_k on nodes

[1] In particular, above we are considering a graph over y given by a chain where there is an edge between each (y_t, y_{t+1}) pair.

are referred to as node features. For each feature function g_k or f_ℓ we associate a trainable parameter μ_k or λ_ℓ, respectively, that can be estimated from training data. Then, given the graph $G = (V, E)$, the feature functions g_k and f_ℓ and the parameters μ_k and λ_ℓ, the conditional probability of a label set y given the observation x is given by:

$$p(y|x) \quad \propto \quad \exp\left(\sum_{\ell,(s\sim t)\in E} \lambda_\ell f_\ell(s,t,y_s,y_t,x) + \sum_{k,t\in V} \mu_k g_k(t,y_t,x) \right)$$

More precisely, if we let $\theta = (\lambda, \mu)$, the proper conditional distribution defined by the CRF over y given x is given by

$$p(y|x,\theta) \quad = \quad \frac{1}{Z(x,\theta)} \exp\left(\sum_{\ell,(s\sim t)\in E} \lambda_\ell f_\ell(s,t,y_s,y_t,x) + \sum_{k,t\in V} \mu_k g_k(t,y_t,x) \right) \qquad (2.1)$$

where $Z(x, \theta)$ is the normalization factor required to ensure that a valid probability distribution over y is obtained. Note, however, that to perform structured *classification*, the normalization constant $Z(x, \theta)$ does not matter because we only need to determine $y^* = \arg\max_y p(y|x, \theta)$. Therefore, once the model parameters μ_k and λ_ℓ have been estimated, we no longer need to compute $Z(x, \theta)$.

Another important property that we exploit below is that $\log p(y|x, \theta)$ is a *concave* function of the parameters $\theta = (\mu, \lambda)$, which facilitates efficient estimation.

Since they have been proposed, CRFs have demonstrated significant advantages over HMMs and MEMMs in classification accuracy, and as a result have become very popular. The first main advantage over HMMs is that CRFs are discriminative models that directly represent $p(y|x)$ instead of $p(y, x)$. This allows CRFs to use much more flexibility, via feature functions, to capture more complex, long range dependencies of y on x, which would cause intractability in a generative representation. The second main advantage over MEMMs is that CRFs are normalized globally over the entire y sequence, and not locally over each y_t given y_{t-1}, allowing for better training algorithms that overcome the "label bias problem" that afflicts directed graphical models [69].

2.2.2 Parameter Estimation for CRFs

To acquire the parameters for a graphical model, one normally needs to estimate them from data. Different estimation techniques are required depending on whether one is learning parameters for a generative or discriminative model.

For a discriminative model, given paired data $D = (x^i, y^i)_{i=1,..,T}$, one normally trains the parameters to maximize the *conditional* likelihood of the target labels y^i given the inputs x^i. Note that for a sequence model, like CRFs, the input observations $x^i = x^i_0, ..., x^i_{N_i}$ and output observations $y^i = y^i_0, .., y^i_{N_i}$ are themselves sequences, but we still make the IID assumption between these sequences. Thus, we formulate a conditional likelihood objective to be maximized as a function of θ as

$$l(\theta) \quad = \quad \sum_{x,y} \tilde{p}(x,y) \log p(y|x,\theta)$$

where $\tilde{p}(x,y)$ denotes the empirical frequency of the sequence pair (x,y) in the training data D.

For a CRFs model in particular, we can substitute the conditional probability it defines on y given x, (2.1), into the above log-likelihood objective to obtain

$$l(\theta) \quad = \quad \sum_{\ell,(s\sim t)\in E} \lambda_\ell f_\ell(s,t,y_s,y_t,x) + \sum_{k,t\in V} \mu_k g_k(t,y_t,x)) - \log(Z(x,\theta)) \qquad (2.2)$$

where $\theta = (\mu, \lambda)$. The discriminative MLE principle requires us to compute the parameters θ that maximize this objective.

The first step toward computing a maximizer of (2.2) is to compute its gradient with respect to the parameters $\theta = (\mu, \lambda)$. For example, the derivative with respect to λ_ℓ is given by

$$\begin{aligned} \frac{\partial l(\theta)}{\partial \lambda_\ell} &= \sum_{x,y} \tilde{p}(x,y) \sum_{t=1}^{N} f_\ell(t-1,t,y_{t-1},y_t,x) \\ &\quad - \sum_{x,y} \tilde{p}(x) p(y|x,\theta) \sum_{i=1}^{N} f_\ell(t-1,t,y_{t-1},y_t,x) \\ &= E_{\tilde{p}(x,y)}[f_\ell] - E_{\tilde{p}(x)p(y|x,\theta)}[f_\ell] \end{aligned}$$

Note that setting the gradient to zero implies a constraint that must be satisfied by the optimal parameters θ: the expectation of the feature functions with respect to the model distribution must be made equal to their empirical expectations. This is the standard constraint of maximum entropy estimation [104, 3].

There are two basic techniques used to compute an optimal θ for a CRF model. The most obvious approach is just to perform standard optimization based on gradient ascent or Newton optimization. Another approach is based on the iterative scaling method [15, 34].

Gradient-based Methods By computing the gradient vector $g(\theta)$, one can naively make progress in the objective by moving a current parameter estimate θ a small amount in the gradient direction

$$\theta^{(k+1)} \quad = \quad \theta^{(k)} + \alpha^k g(\theta^{(k)})$$

where α^k is the step size and $g(\theta^{(k)})$ is the gradient vector computed at the current parameter estimate $\theta^{(k)}$. Common methods for optimization based solely on gradient vectors are steepest ascent or conjugate gradient. Importantly, as observed above, the objective function $l(\theta)$ is *concave* in θ, and thus standard optimization techniques, once they converge, are guaranteed to produce a globally optimal value of θ.

The main drawback of purely gradient based methods is that their convergence is slow and they require many iterations to solve the problem. For that reason, one is strongly tempted to use a second order technique, such as Newton's method, which computes a parameter update based on the Hessian matrix of second derivatives. If $H(\theta^{(k)})$ denotes the Hessian matrix computed at parameter estimate $\theta^{(k)}$, then the Newton update is give by

$$\theta^{(k+1)} \quad = \quad \theta^{(k)} - H^{-1}(\theta^{(k)})g(\theta^{(k)})$$

Although this update converges to a maxima with much fewer iterations than gradient based methods, each iteration is much more expensive to compute. In fact, the space required to store the Hessian matrix is usually far too great to be practical, even ignoring the time cost needed to invert it. Real training problems on CRFs normally involve thousands or even millions of features, which would result in a Hessian matrix with millions or even trillions of entries to store.

A practical compromise is to use limited-memory forms of quasi-Newton methods, such as the limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [95, 91]. The main idea here is to only store a low rank approximation to the inverse Hessian matrix which can be efficiently updated based on each step direction and each gradient vector computed. Such an approach usually results in reasonably fast convergence with adequate computation.

Iterative Scaling Method Iterative scaling is an alternative optimization method specifically tailored to the training problems encountered with log-linear probability models. Although these methods have an elegant derivation and convergence proof [97], they generally do not converge faster than standard optimization techniques like Newton or quasi-Newton methods. Nevertheless, iterative scaling is an interesting training algorithm where one computes a similar update to gradient ascent:

$$\theta^{(k)} = \theta^{(k)} + \delta^{(k)}\theta^{(k)}$$

Lafferty *et. al* have proposed two iterative scaling methods for estimating the parameters of CRFs [69], which they refer to as the generalized iterative scaling method and improved iterative scaling method respectively.

In my own work, I have focused on standard quasi-Newton based optimization techniques, which generally converge faster [90].

2.3 Inference

In standard applications of a graphical model, the values of some nodes are observed while some other nodes are unobserved, or “hidden”. The inference problem is to calculate or estimate the conditional probability distribution over theses hidden nodes given the values of the observed nodes. Let H denote the hidden nodes and O represent the observed nodes. We need to calculate the conditional probability $P(H|O)$

$$P(H|O) = \frac{P(H,O)}{P(O)} = \frac{P(H,O)}{\sum_H P(H,O)}$$

The difficulty in this computation arises from the need to compute a marginal probability $\sum_H P(H, O)$ to normalize the conditional probability.

Inference is an inherently hard problem in the worst case, which means that one often has to resort to approximate rather than exact inference methods in practice. Nevertheless, a variety of useful exact inference methods, that usefully exploit the structure of the graphical model, also exist. I briefly discuss both exact and approximate inference methods below.

2.3.1 Exact Inference

There are different techniques for computing the exact posterior probabilities of unobserved nodes given observed nodes in a graphical model.

Elimination Algorithm The most straightforward approach for computing marginal or conditional probabilities is to use a graph theoretic elimination algorithm, often referred to as the "vertex elimination algorithm" [60]. The basic idea of this algorithm is to sum out variables from a list of factors one by one. This algorithm requires a specific ordering on the variables, referred to as an elimination ordering, to be provided as input. The algorithm then proceeds mechanically by eliminating variables according to the given order.

Junction Tree Algorithm(Tree decomposition) One shortcoming of the naive elimination algorithm above is that it does not do a good job of caching computations that might be useful for other queries.

The *junction tree algorithm*, also called *tree decomposition algorithm* in tree decomposition, the variable may not be probabilistic distribution , is the best understood, efficient and provably correct method for concurrently computing multiple queries. The detail of tree decomposition for generate graph problem will be introduced in chapter 2.4.

The running time of the tree decomposition algorithm is exponential in the size of the largest clique. In many cases this leads to a practical algorithm for performing inference. However, in the worst case, the tree decomposition algorithm can be impractical because the largest clique size can sometimes grow too large. A good example of this is image modeling, which considers graphical models defined over a grid of pixels. In these models, it is common to have several thousand pixels, but the maximum clique size in any junction tree must be at least the number of pixels in either a row or column of the image. An exponential running time in this quantity makes the exact junction tree algorithm useless for image processing applications. In these applications, it is necessary to consider approximate inference algorithms.

2.3.2 Approximate Inference

In general, it is computationally hard to compute exact conditional probabilities in a graphical model if there is no special structure present (such as being a tree). Even though exact methods can be applied to nontrivial models, they cannot scale up to arbitrarily large models. Ultimately, some form of approximate inference algorithm must be employed to perform inference on large, arbitrary graphical models.

There are several different kinds of approximate inference algorithms for graphical models, each with different advantages and disadvantages.

Loopy Belief Propagation The main idea of loopy belief propagation is to exploit the same propagation process as used in the junction tree algorithm but to simplify the graph to yield the faster and tractable inference at the cost of sacrificing accuracy. Surprisingly, the same propagation process when run on graphs that do not satisfy the junction tree property, yields a good approximation to exact inference in practice. This is even true on graphs with loops. Loopy propagation is an increasingly popular method of approximate inference for graphical models. It is not guaranteed to converge in general, but in addition to its empirical success, some theoretical results are known about its approximation properties [53, 52, 96, 62].

Monte Carlo Estimation Monte Carlo refers to methods that solve problems by generating a series of random numbers and observing that fraction of the numbers obeying some property or properties [81]. For example, to estimate a conditional probability $p(y|x)$, one could first sample from $p(x, y)$ (which is easy in a generative model like an HMM), discard all samples that do not agree on x, and use the remaining samples to estimate the frequency of y.

There are several kinds of sampling methods. Rejection sampling defines an upper bound of the distribution and only accepts samples with a certain probability. However, sometimes it is difficult to find a suitable upper bound. Importance sampling draws data from a different distribution than the target, but then weights the samples to correct for the difference. A more efficient approach in high dimensional problems is Markov Chain Monte Carlo (MCMC) [94]. The MCMC method sets up a suitable Markov chain that has the target distribution as its stationary distribution. Then by observing the simulation and recording the states one can estimate desired probabilities. Special cases of MCMC methods are Gibbs sampling and the Metropolis algorithm [42].

The advantage of the Monte Carlo methods is that they are easy to implement and provide theoretical guarantees of convergence. The disadvantage is that sometimes the convergence times will be very long. These methods are also hard to diagnose and debug.

Variational Methods Variational methods [61] provide an alternative approach to approximate inference. The main idea of these methods is to avoid performing sums over exponentially many summands, and instead find a deterministic bound. For example, the mean-field algorithm, which was widely used in image processing, decouples the grid model into isolated nodes. We define a new parameter for each node (a variational parameter) and then iteratively update these parameters by minimizing the cross-entropy (KL divergence) between the approximate and true probability distributions. In [26], I present a variational Bayesian framework for performing inference, density estimation and model selection in a special case class of graphical models–Hidden Markov Random Fields (HMRFS [39]).

2.4 A Tree-Decomposition Approach to the Graph Labeling Problem

In this section, we first introduce the concept of tree decomposition. Then I will describe several different methods to decompose a graph into a tree decomposition and finally describe how to search for the optimal label assignment based on the tree decomposition of a graph.

2.4.1 Tree Decomposition

Tree decomposition of a graph was introduced by Robertson and Seymour [106] a long time ago. The technique of decomposing a sparse graph to its tree-decomposition has been applied to many NP-hard problems such as frequency assignment problems [68] and Bayesian inference [10].

> *L*:et $G = (V, E)$ be a graph. A tree decomposition of G is a pair (T, X) satisfying the following conditions:
>
> 1. $T = (I, F)$ is a tree with a node set I and an edge set F,
> 2. $X = \{X_i | i \in I, X_i \in V\}$ and $\bigcup_{i \in I} X_i = V$. That is, each node in the tree T represents a subset of V and the union of all the subsets is V,
> 3. for every edge $e = \{v, w\} \in E$, there is at least one $i \in I$ such that both v and w are in X_i, and
> 4. for all $i, j, k \in I$, if j is a node on the path from i to k in T, then $X_i \bigcap X_k \subseteq X_j$.
>
> The width of a tree decomposition is $\max_{i \in I}(|X_i| - 1)$. The tree width of a graph G, denoted by $tw(G)$, is the minimum width over all the tree decompositions of G.

Figures 2.4 and 2.5 give an example of a graph and one of its tree decompositions, respectively. The width of a tree decomposition is a key factor in determining the computational complexity of all the tree-decomposition based algorithms. The smaller the width of a tree decomposition is, the more efficient the tree decomposition based algorithms. Therefore, we need to optimize the tree decomposition of a graph such that we can have a very small tree width.

2.4.2 Algorithms for Tree Decomposition of a Graph

The optimal tree decomposition of a general graph is *NP*-hard [9], which means it is unlikely to find the optimal tree-decomposition of a graph within polynomial time. The graph

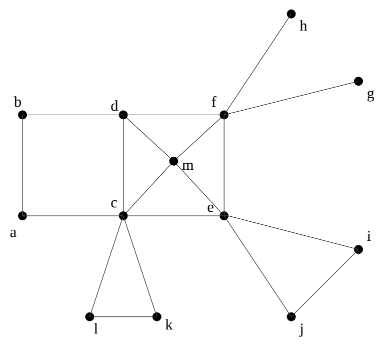

Figure 2.4: Example of a graph.

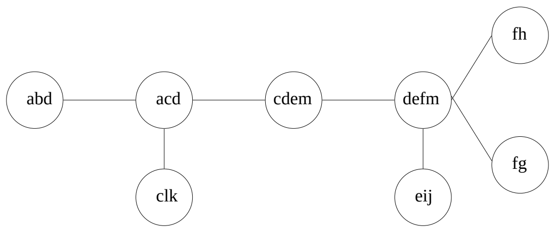

Figure 2.5: Example of a tree decomposition with width 3.

discussed in this paper has two special features: (i) the graph is a geometric neighborhood graph and (ii) the graph is also sparse. Therefore, using the sphere separator theorem [89], we can have a low-degree polynomial-time algorithm to decompose the graph into some components with size $O(\frac{D_u}{D_l}|V|^{\frac{2}{3}}\log|V|)$ [127]. Strictly speaking, we have the following theorem.

Theorem 1 *Let $G = (V, E)$ denote a graph describing the relationship among a set of 3D points. The distance between any two vertices in G is at least a constant D_l and the distance between any two adjacent vertices is no more than D_u. There is a low-degree polynomial-time algorithm to decompose G into some components with size $O(\frac{D_u}{D_l}|V|^{\frac{2}{3}}\log|V|)$.*

The sphere separator based tree decomposition algorithm is not easy to implement. Besides this theoretically-sound tree decomposition method, many heuristic-based tree decomposition methods exist in the literature. Later in this paper we will compare the performance of six heuristic-based tree decomposition algorithms and show that some work very well in practice. The graph under consideration can be decomposed into some very small components, which leads to a very efficient tree-decomposition based graph labeling algorithm. Here we describe six different tree decomposition algorithms.

Method 1: minimum vertex separator This method recursively partitions a graph into two disconnected subgraphs using a minimum vertex separator [37]. Finally, we get a tree with separators being the internal nodes and the final subgraphs being the leaf nodes. All the vertices along the path from the tree root node to any tree node form a decomposition component. Given an undirected graph $G = (V, E)$, a vertex set S is called an (a, b)-*vertex-separator* if it satisfied (i) $\{a, b\} \subset V \setminus S$ and (ii) every path connecting a and b in G passes through at least one vertex contained in S. Among all the separators, the one with the minimum cardinality is called the minimum (a,b)-*vertex-separator*.

Method 2: two-way-1/2-triangle This method is similar to the minimum vertex separator method. The only difference is that this method uses a minimum 1/2-balanced vertex separator rather than the minimum vertex separator. The "1/2-balanced" vertex separator always partitions a graph into two subgraphs, each containing no more than one half of all the vertices.

Method 3: two-way-2/3-triangle This method is similar to the two-way-1/2-triangle method. The only difference is that this method uses a minimum 2/3-balanced vertex separator rather than a minimum 1/2-balanced vertex separator. The "2/3-balanced" vertex separator always partitions a graph into two subgraphs, each containing no more than two thirds of all the vertices.

Method 4: minimum-degree It is a heuristic algorithm that iteratively chooses one vertex and forms a decomposition component based on this vertex [16]. At each iteration, this method chooses a vertex with the smallest number of neighbors and adds edges to the graph such that any two neighbors of the selected vertex are connected by an edge. The selected vertex with its neighbors forms a partition of the graph. Then, this method removes the selected vertex from the graph and recursively chooses the next vertex until the graph is empty.

Method 5: minimum-width It is a heuristic method similar to the "minimum-degree" method. It differs from the "minimum-degree" method in that this method does not connect edges to the neighbors of the chosen vertex.

Method 6: minimum-discrepancy It is a heuristic method similar to the "minimum-degree" method. It differs from the "minimum-degree" method in that at each iteration, this method selects the vertex with the minimum number of edges missing between its neighbors.

2.4.3 Tree Decomposition-Based Graph Labeling Algorithms

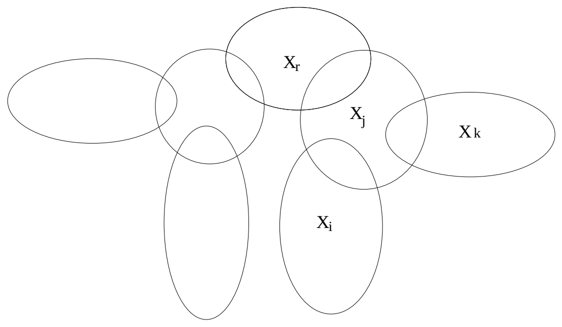

Figure 2.6: A tree decomposition (T, X) of G.

Assume that we have a tree decomposition (T, X) of a geometric neighborhood graph G. We describe an algorithm to search for the optimal label assignment based on the tree decomposition. For simplicity, we assume that tree T has a root X_r and that each node is associated with a height. The height of a node is equal to the maximum height of its child nodes plus one. Figure 2.6 shows an example of a tree decomposition in which component X_r is the root. Let $X_{r,j}$ denote the intersection between X_r and X_j. If we remove all the vertices in $X_{r,j}$, then this tree decomposition becomes two disconnected subtrees. Let $F(X_j, A(X_{r,j}))$ denote the optimal label assignment of the subtree rooted at X_j given that the label assignment to $X_{r,j}$ is fixed to $A(X_{r,j})$. Then $F(X_j, A(X_{r,j}))$ is independent of the rest of the whole tree decomposition. Let $C(j)$ denote the set of child components of X_j and $Score(X_j, A(X_j))$ denote the assignment score of component X_j with the label assignment being $A(X_j)$. Let $D[X]$ denote all the possible label assignments to the vertices in X. Therefore, we have the following recursive equation.

$$\begin{aligned} F(X_j, A(X_{r,j})) = \min_{A \in D[X_j - X_{r,j}]} \{ & \sum_{i \in C(j)} F(X_i, A(X_{j,i})) \\ & + Score(X_j, A(X_j)) \} \end{aligned}$$

Based on the above equation, we can calculate the optimal label assignment in two steps. First, we calculate the optimal energy function from bottom to top and then we extract the optimal label assignment from top to bottom.

Bottom-to-Top Starting from a leaf node i in the tree T, we assume that node j is the parent of i in T. Let $X_{j,i}$ denote the intersection between X_i and X_j and $D[X_{j,i}]$ the set of all the possible label assignments to the vertices in $X_{j,i}$. Given a label assignment $A(X_{j,i}) \in D[X_{j,i}]$ to the vertices in $X_{j,i}$, we enumerate all the possible label assignments to $X_i - X_{j,i}$ and then find the best label assignment such that the energy of the subtree rooted at X_i is minimized. We use $F(X_i, A(X_{j,i}))$ to denote this minimized energy. At the same time, we also save the optimal assignment to $X_i - X_{j,i}$ for a given $A(X_{j,i})$ since in the top-to-bottom step we need it for traceback. For example, in Figure 2.5, if we assume the node acd is the root, then node $defm$ is an internal node with parent $cdem$. For each label assignment to vertices d, e and m, we can find the best label assignment to vertex f such that the energy of the subtree rooted at $defm$ is minimized. In this bottom-to-top process, a tree node can be calculated only after all of its child nodes are calculated. When we calculate the root node of T, we enumerate all the possible label assignments to this node and find the optimal label assignment such that the energy is minimized. This minimized energy is also the minimum energy of the whole system.

Top-to-Bottom After finishing calculating the root node of tree T, we obtain the optimal label assignment to this root node. Now we trace back from the parent node to its child nodes to extract out the optimal label assignment to all the child nodes. Assume that we have the optimal label assignment to node j and node i is a child of j. We can easily extract out the optimal label assignment to $X_i - X_{j,i}$ based on the assignment to $X_{j,i}$ since we have already saved this label assignment in the bottom-to-top step. Recursively, we can track down to the leaf nodes of T to extract out the optimal label assignment to all the vertices in G.

In addition, based on the definition of tree decomposition, one vertex might occur in several tree nodes of the tree decomposition of G. To avoid incorporating the singleton score of this vertex into the overall system energy more than once, we incorporate the singleton score of this vertex into the system only when we are calculating the tree node with the maximal height among all the nodes containing this vertex. We can prove that there is one and only one such a tree node. Similarly, an edge in graph G might also occur in several tree nodes. We can use the same method to avoid a redundant addition of its pairwise score.

In the following chapter, I will show how the tree decomposition algorithm is used for protein structure prediction and protein structure alignment. I will also show the effectiveness of the algorithm and the reason why it works.

Chapter 3

Background on Protein Problems

Over the last few years, major achievements have been made in the area of genomics, especially in the accomplishment of the Human Genome Project. In the post-genomic era, more research has focused on proteins. Proteins play a central role in biology, and the understanding of protein functions has become a key step toward modeling complete biological systems. It has been established that the functions of a protein are directly linked to its three-dimensional structure. Proteins sharing the similar structures are more likely to have similar functions. In the following sections, I review some related problems such as protein structure prediction and protein structure alignment that is related to my research.

3.1 Introduction to Protein Structure

Proteins are large, complex molecules consisting of chains of amino acids, which are the basic units of proteins. The sequence of amino acids in each protein is determined by the gene that encodes it. The gene is transcribed into a messenger RNA (mRNA) and the mRNA is translated into a protein by the ribosome. The spatial conformation of a protein is determined by the order of the amino acids and the side chain. The description of protein structure can be broken down into four levels:

1. The basic unit of a protein is the *amino acid*, which consists of an α-carbon atom in the center, a N-terminal (-NH2), a hydrogen atom, a carboxyl group (-COOH) and a side

chain R group. In nature, there are 20 different kinds of amino acids in total, which differ only in the R group. The primary structure refers to the ”linear” sequence of amino acids.

2. *Secondary structure* is the “local” ordered structure brought about via hydrogen bonding, mainly within the peptide backbone. There are two different types of secondary structure: the α-helix and the β-sheet.

3. *Tertiary structure* is the “global” folding of a single polypeptide chain. With long-range interactions, all secondary fragments in a protein can form a specific tertiary structure with the loops connecting between them. An important type of tertiary structure is also called the fold, and it is much more conserved than sequences during the course of protein evolution. There are limited types of folds that occur in nature according to the protein data bank.

4. *Quaternary structure* involves the association of two or more polypeptide chains into a multi-subunit structure.

When we refer to protein structure, normally we mean tertiary and quaternary structure. Figure 3.1 shows the four levels of protein structures.

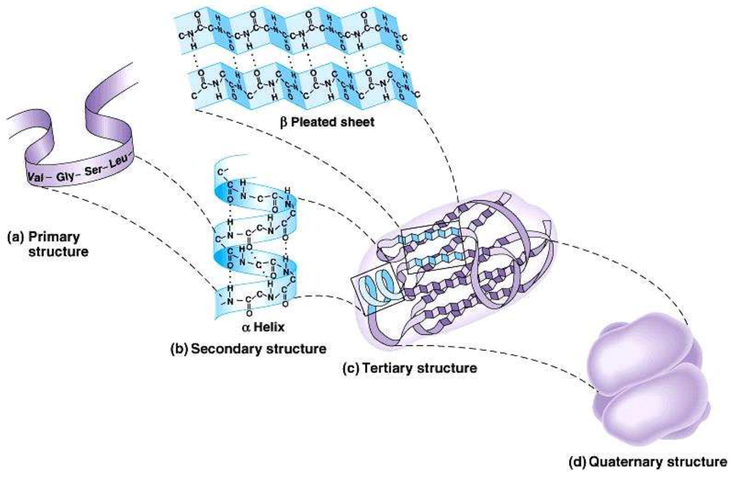

Figure 3.1: Four levels of protein structure

3.2 Representation of 3-D Protein Structure—Contact Graphs

Here we use a contact graph "contact graphs" to represent the 3D protein structure. Each residue is represented by a vertex in V, associated with its secondary structure type, its N-H bond and the coordinate of its residue center. For each residue, we use its C_α atom as the residue center. There is a contact edge $(i, j) \in E$ between two residues i and j if and only if their spatial distance is within a given distance cutoff D_u. Since the distance between two hydrogen atoms related by a long-range NOE restraint usually is no more than $5\dot{A}$ away from each other, D_u ranges from $7\dot{A}$ to $8\dot{A}$. One edge is also added to connect any two sequentially adjacent residues. Given a protein chain A, let $G[A]$ denote its contact map graph. For a substructure P of A, let $G[P]$ denote the contact map subgraph induced by substructure P.

In a typical protein, two residues cannot be arbitrarily close, which is one of the underlying reasons why lattice models can be used to approximate protein folding. According to simple statistics on the PDB database [14], 99% of inter-residue distances are more than $3.5\dot{A}$. Let the constant D_l ($D_l > 0$) denote the minimum inter-residue distance in a protein. Therefore, it can be easily verified that any residue can be adjacent to at most $(1+\frac{2D_u}{D_l})^3$ residues. Figure 3.2 show a sample contact graph. Suppose a small protein with 5 residues, residue 1 and residue 3 is very close while residue 2 and 5 is very close. Edges are added between node 1,3 and node 2, 5.

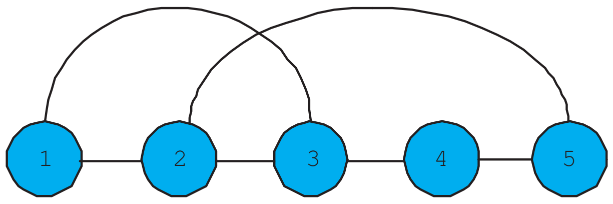

Figure 3.2: Example of contact graph

3.3 Protein Structure Prediction

The problem of protein structure prediction is: Given the sequence of a protein (Primary structure), what is its 3D structure? (i.e. either Tertiary structure or Quaternary structure)

It is well known that the function of the protein is determined by its 3D structure. Currently experimental methods to determine the protein structure, such as X-ray crystallography and magnetic resonance spetroscopy (NMR), are both time-consuming and expensive. For example, using the X-ray method, it will take half a year and millions of dollars to determine the structure of some large proteins. Subsequently, there is a big gap between the numbers of available sequences and available structures, which makes the computation methods greatly needed.

3.3.1 Main Methods for Structure Prediction

There are many approaches to predict the structure of proteins, which fall into four main types.

Homology Modelling The homology modelling approach attempts to predict the structure of a given protein by first searching for homologous proteins from a database and then using the retrieved structures to provide predictions. Factors that account for homology could either be similar sequence or structure or that the proteins evolved from a common ancestor.

***Ab inito* Folding** The *ab initio* folding approach builds the structural model directly from the original primary sequence. By defining a score function, optimization techniques can be used to find the most probable model. Although some achievements have been obtained using this approach in recent years, the performance of this method is still discouraging because the real physical processes of folding are still not very clear yet.

Consensus Methods In this approach, one predicts the structure by analyzing and combining the outputs of different structure prediction methods. Although this approach can achieve a more sensitive and specific prediction than any individual approach, it does not really contribute new understanding to the underlying problem of structure prediction.

Threading Methods Threading methods predict protein structure by exploiting statistical knowledge of the relationship between the structure and the sequence. Predictions are made by first threading the query sequence to each template in a database and then choosing a best-fit template based on the threading results. Protein threading consists of the following five components: (1) a library of template structures; (2) a representation of targets and templates; (3) an objective function for measuring the quality sequence-template alignments; (4) an algorithm finding the best sequence-template alignment; and (5) a method for selecting the best template based on all the sequence-template alignments. A library of template structures is a set of representative structures selected from the PDB. Usually, to save computing time, among all the highly similar protein structures, only one is kept in the template library. To construct a library of template structures, we can cluster all the proteins from PDB into several thousand groups and then choose one representative from each group to serve as a structural template. We can also build a template library using a set of representative proteins from the SCOP database [93].

Compared with the homology method, which mainly considers sequence similarity between the target and the template, protein threading makes use of the structural information encoded in the template to improve prediction accuracy, including the use of secondary structure, solvent accessibility and pairwise interactions. The threading approach is one of the most promising structure prediction methods developed to date [123, 66].

3.3.2 Threading Algorithms

For threading methods, if we use a contact graph based method (that is, a 2D graph based method), it is NP-hard to find the best sequence-template alignment [73, 5], which means it is unlikely to have a polynomial-time algorithm for finding the best alignment. Therefore, one must either resort to approximate threading algorithms or try to develop fast, exact algorithms which nevertheless cannot be scaled up to arbitrarily large problems.

Approximate algorithms Many approximate or heuristic algorithms have been proposed for protein threading. Madej et al. [114, 20] proposed a Gibbs sampling technique to search for the optimal sequence-template alignment. Godzik et al. [1] and Jones et al. [58] proposed an interaction-frozen approximation algorithm to find a good sequence-template alignment iteratively. In each iteration, this approach assumes that one end of a contact is fixed when calculating the pairwise interaction score. Thiele and Zimmer developed a recursive dynamic programming approach [102]. Recently, Balev [13] proposed a Larangian relaxation algorithm to solve this problem. The advantage of the Lagrangian approach is that one can estimate the gap between the optimal value and the best objective value observed so far. This algorithm also runs very efficiently in practice when the target and the template are similar, although it cannot guarantee the optimal solution.

Exact algorithms If the length of the template is limited, a dynamic programming algorithm can be used to find the optimal sequence-template alignment. However, in most cases the running time of dynamic programming is impractical. Lathrop and Smith [75, 101, 74] developed a branch-and-bound algorithm to solve for the optimal sequence-template alignment. In this algorithm, the complete search space is split into many small subspaces by partitioning the alignment domain of a single template position into several intervals. The lower and upper bound of the objective function in each subspace is estimated based on the sequence-template alignment generated without considering pairwise interactions. During the search process, some subspaces can be discarded based on the estimated lower and upper bounds. The algorithm terminates when the pruned search space contains only the best sequence-template alignment. This algorithm runs particularly fast when the similarity between the target and the template is high.

Xu et al. [131, 130] designed a divide-and-conquer algorithm that is used in their structure prediction program, PROSPECT [66], based on an observation that if the contact cutoff distance is not big, then the residue interaction pattern of many templates can be represented as a sparse graph. This algorithm splits a template into two subsegments resulting in few inter-segment contacts, recursively aligns each subsegment to the target respectively, and finally, merges the alignments of two subsegments to form a complete alignment. PROSPECT runs very fast for approximately three quarters of the templates in the library. However, it runs very slowly or runs out of memory on a 32-bit platform on the remaining one-quarter templates, which have many inter-residue contacts.

Both the branch-and-bound and divide-and-conquer algorithms can find the globally optimal solution to the sequence-template alignment problem. However, both algorithms are still computationally expensive and not suitable for genome-scale protein structure prediction. Xu proposed an efficient linear programming (LP) approach to the optimal sequence-template alignment problem. Empirically, the LP approach can find the optimal alignment for 99% of threading instances within polynomial-time [131, 130].

3.3.3 Fold Recognition

Fold recognition identifies the best template for a given target based on all the generated sequence-template alignments. The sequence-template alignment score cannot be directly used to rank the templates due to the bias introduced by residue composition and the number of alternative sequence-template alignments for a given pair of target and template [21]. Both Z-score [114, 21] and machine learning methods [130, 57] are used to do fold recognition. Most of the current structure prediction programs use Z-score [108, 66, 113] to recognize the best-fit templates, whereas several programs such as GenTHREADER [57] and PROSPECT-I [134] use a neural network model to rank the templates. The neural network method formulates the fold recognition problem as a classification problem. The machine learning methods extract some features from a sequence-template alignment to describe the quality of this alignment in many different aspects and then try to predict if this pair of target and template is in the same fold or to predict the overall quality of the alignment. A machine learning model directly predicting the quality of a three-dimensional structural model built from a sequence-template alignment can also be used to conduct fold recognition [120]. According to [124], the machine learning methods are better than Z-score in terms of both sensitivity and specificity. In fact, Z-score cannot cancel out all the bias introduced by the protein sizes. A large target protein tends to have a large Z-score.

The Z-score was proposed to cancel out the bias caused by sequence residue composition and by the number of alternative sequence-template alignments. A typical procedure to calculate Z-score [21] is as follows: (1) shuffle the residues of the target randomly; (2) find the optimal alignment between the shuffled target and the template and calculate the alignment score; (3) repeat the above two steps as many as 100 times or until the distribution of the generated alignment scores converges. Z-score is the alignment score in standard deviation units relative to the mean alignment score. The higher the Z-score is, the better the alignment.

The Z-score method has the following two drawbacks. First, it takes a lot of extra time to calculate Z-score for a pair of target and template. In order to calculate Z-score for each pair, the target has to be shuffled and threaded many times to the template. This hinders the use of Z-score methods in genome-scale structure prediction. In contrast, machine learning methods require only one-time threading for a given target and template. Secondly, the Z-score is hard to interpret, especially when the scoring function is the weighted sum of various energy items such as mutation score, environmental fitness score, pairwise score, secondary structure score, gap penalty and score induced from NMR data. For example, when the target is shuffled, shall we shuffle the position specific profile information and the predicted secondary structure type at each sequence residue? If we choose to shuffle the secondary structure, then the shuffled secondary structure arrangement does not look like a protein's since the regular secondary structure types (i.e., -helix and - strand) disperse randomly in the target. Otherwise, if we choose to predict the secondary structure again, the whole process will take a very long time.

3.4 Protein Structure Alignment

It is known that a protein's function is determined by its 3D structure and that two proteins sharing similar structures are more likely to have similar function. Therefore, it is important to provide structure alignment algorithms and software to biologists to aid in protein function determination. Normally, protein structure alignment tools are the most common used tools to analyze the similarity of two proteins [71, 76].

There are two major methods to measure the similarity between two proteins: the coordinate distance-based method and inter-residue contact-based method. The first type of measure uses the Euclidean distance between two matched residues or atoms in the two proteins compared. Many programs such as STRUCTAL [43], 3dSearch [112], and VAST [44] belong to this category. To use this method, the optimal rigid-body transformation between two proteins must be determined. The other type of measure employs a contact map graph to describe the structure of a protein and compares the contact map graphs of two proteins under consideration [70, 23]. A contact in a protein is a pair of residues that are spatially close to each other. A contact map graph consists of all the residues (i.e, vertices) and their contacts (i.e., edges) and is inferred from crystal structures. Using this method, the protein structure alignment problem is often formulated as a maximum common subgraph problem. It is unnecessary to find the optimal rigid-body transformation before obtaining the best match between two proteins. Usually the rigid-body transformation is calculated after the best match is determined. A variant of a contact map representation of a protein structure is a distance matrix in which an element is the spatial distance between two residues. Two distance matrices are compared to render the best common submatrix. Several widely used protein structure alignment tools such as DALI [49], CE [111] and SARF [7] employ the distance matrix representation of a protein structure.

Previous studies show that contact map-based protein structure alignment is NP-hard and also hard to approximate [45, 118, 56], regardless of whether the alignment is sequential or non-sequential. A non-sequential alignment refers to one in which the sequential order of residues in a protein is ignored, and only the spatial proximity between two residues is taken into consideration. Many structure alignment tools support both sequential or non-sequential structure alignment [49, 136, 35, 138].

Many protein structure comparison programs such as DALI [49] use heuristic algorithms to find a good, but not the best, alignment. The advantage of these algorithms is that they are computationally efficient. While these algorithms have no performance guarantee, empirically they generate good alignment accuracy. There are also some globally optimal algorithms for this problem. For example, Lancia et al. [70] used a branch-and-cut method to find the optimal alignment between two proteins when a protein is modeled by a contact map. Later, Caprara & Lancia also developed a Lagrangian relaxation algorithm [23], which runs fast and sometimes can generate a globally optimal solution. The disadvantage of these algorithms is that they do not have good theoretical time complexity. Recently, Kolodny & Linial [67] proposed an interesting polynomial-time approximation scheme for this problem when a STRUCTAL-type objective function [43] (i.e., Gerstein &

Levitt's coordinate distance based measurement) is used to measure the similarity between two proteins. However, there is still no good approximation algorithm in the case where the two proteins under consideration are modeled by a contact map. Instead, Goldman, Papadimitriou & Istrail have shown that, based on the maximum common subgraph formulation, the contact map-based protein structure alignment problem is hard to approximate [45]. The hardness of the protein structure alignment problem partially comes from the fact that when two contact maps are aligned, the geometric information in the protein structure is not taken into consideration.

Chapter 4

Semi-Supervised Conditional Random Fields

In this chapter, I will introduce some research work on machine learning, special on graphical models, from theory to application. The background for graphical models and conditional random fields has been introduced in chapter 3.

I present a new semi-supervised training procedure for conditional random fields (CRFs) that can be used to train sequence segmentors and labelers from a combination of labeled and unlabeled training data. My approach is based on extending the minimum entropy regularization framework to the structured prediction case, yielding a training objective that combines unlabeled conditional entropy with labeled conditional likelihood. Although the training objective is no longer concave, it can still be used to improve an initial model (e.g. obtained from supervised training) by iterative ascent. I apply the new training algorithm to the problem of identifying gene and protein mentions in biological texts, and show that incorporating unlabeled data improves the performance of the supervised CRFs in this case. This work was published in [54].

4.1 Introduction

Semi-supervised learning is often touted as one of the most natural forms of training for language processing tasks, since unlabeled data is so plentiful whereas labeled data is usually quite limited or expensive to obtain.

The attractiveness of semi-supervised learning for language tasks is further heightened by the fact that the models learned are large and complex, and generally even thousands of labeled examples can only sparsely cover the parameter space. Moreover, in complex *structured* prediction tasks, such as parsing or sequence modeling (part-of-speech tagging, word segmentation, named entity recognition, and so on), it is considerably more difficult to obtain labeled training data than for classification tasks (such as document classification), since hand-labeling individual words and word boundaries is much harder than assigning text-level class labels.

Many approaches have been proposed for semi-supervised learning in the past, including: generative models [24, 29, 63], self-learning [25, 135], co-training [17], information-theoretic regularization [30, 46], and graph-based transductive methods [33, 32, 122]. Unfortunately, these techniques have been developed primarily for single class label classification problems, or class label classification with a structured input [33, 32, 122]. Although still highly desirable, semi-supervised learning for structured classification problems like sequence segmentation and labeling have not been as widely studied as in the other semi-supervised settings mentioned above, with the sole exception of generative models.

With generative models, it is natural to include unlabeled data using an expectation-maximization approach [63]. However, generative models generally do not achieve the same accuracy as discriminatively trained models, and therefore it is preferable to focus on discriminative approaches. Unfortunately, it is far from obvious how unlabeled training data can be naturally incorporated into a discriminative training criterion. For example, unlabeled data simply cancels from the objective if one attempts to use a traditional conditional likelihood criterion. Nevertheless, recent progress has been made on incorporating unlabeled data in discriminative training procedures. For example, dependencies can be introduced between the labels of nearby instances and thereby have an effect on training [122, 77, 8]. These models are trained to encourage nearby data points to have the same class label, and they can obtain impressive accuracy using a very small amount of labeled data. However, since they model pairwise similarities among data points, most of these approaches require joint inference over the whole data set at test time, which is not practical for large data sets.

In this section, I propose a new semi-supervised training method for conditional random fields (CRFs) that incorporates both labeled and unlabeled sequence data to estimate a discriminative structured predictor. CRFs are a flexible and powerful model for structured predictors based on undirected graphical models that have been globally conditioned on a set of input covariates [69]. CRFs have proved to be particularly useful for sequence segmentation and labeling tasks, since, as conditional models of the labels given inputs, they relax the independence assumptions made by traditional generative models like hidden Markov models. As such, CRFs provide additional flexibility for using arbitrary overlapping features of the input sequence to define a structured conditional model over the output sequence, while maintaining two advantages: first, efficient dynamic program can be used for inference in both classification and training, and second, the training objective is concave in the model parameters, which permits global optimization.

To obtain a new semi-supervised training algorithm for CRFs, I extend the minimum entropy regularization framework of [46] to structured predictors. The resulting objective combines the likelihood of the CRFs on labeled training data with its conditional entropy on unlabeled training data. Unfortunately, the maximization objective is no longer concave, but I can still use it to effectively improve an initial supervised model. To develop an effective training procedure, I first show how the derivative of the new objective can be computed from the covariance matrix of the features on the unlabeled data (combined with the labeled conditional likelihood). This relationship facilitates the development of an efficient dynamic programming for computing the gradient, and thereby allows us to perform efficient iterative ascent for training. I apply the new training technique to the problem of sequence labeling and segmentation, and demonstrate it specifically on the problem of identifying gene and protein mentions in biological texts. The results show the advantage of semi-supervised learning over the standard supervised algorithm.

4.2 Semi-supervised CRFs training

In what follows, I use the same notation as [69]. Let $\mathbf{X}$ be a random variable over data sequences to be labeled, and $\mathbf{Y}$ be a random variable over corresponding label sequences. All components, $\mathbf{Y_i}$, of $\mathbf{Y}$ are assumed to range over a finite label alphabet $\mathcal{Y}$. For example,

$\mathbf{X}$ might range over sentences and $\mathbf{Y}$ over part-of-speech taggings of those sentences; hence $\mathcal{Y}$ would be the set of possible part-of-speech tags in this case.

Assume I have a set of labeled examples, $\mathcal{D}^l = \Big((\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \cdots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\Big)$, and unlabeled examples, $\mathcal{D}^u = \Big(\mathbf{x}^{(N+1)}, \cdots, \mathbf{x}^{(M)}\Big)$. I would like to build a CRFs model

$$\begin{aligned} p_\theta(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z_\theta(\mathbf{x})} \exp\Big(\sum_{k=1}^{K} \theta_k f_k(\mathbf{x}, \mathbf{y})\Big) \\ &= \frac{1}{Z_\theta(\mathbf{x})} \exp\Big(\langle \theta, f(\mathbf{x}, \mathbf{y}) \rangle\Big) \end{aligned}$$

over sequential input and output data $\mathbf{x}, \mathbf{y}$,

where $\theta = (\theta_1, \cdots, \theta_K)^\top$, $f(\mathbf{x}, \mathbf{y}) = (f_1(\mathbf{x}, \mathbf{y}), \cdots, f_K(\mathbf{x}, \mathbf{y}))^\top$ and

$$Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}} \exp\Big(\langle \theta, f(\mathbf{x}, \mathbf{y}) \rangle\Big)$$

The goal is to learn such a model from the combined set of labeled and unlabeled examples, $\mathcal{D}^l \cup \mathcal{D}^u$.

The standard supervised CRFs training procedure is based upon maximizing the log conditional likelihood of the labeled examples in $\mathcal{D}^l$

$$CL(\theta) = \sum_{i=1}^{N} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - U(\theta) \tag{4.1}$$

where $U(\theta)$ is any standard regularizer on θ, e.g. $U(\theta) = \|\theta\|^2/2$. Regularization can be used to limit over-fitting on rare features and avoid degeneracy in the case of correlated features. Obviously, (4.1) ignores the unlabeled examples in $\mathcal{D}^u$.

To make full use of the available training data, I propose a semi-supervised learning algorithm that exploits a form of *entropy regularization* on the unlabeled data. Specifically, for a semi-supervised CRFs, I propose to maximize the following objective

$$\begin{aligned} RL(\theta) &= \sum_{i=1}^{N} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - U(\theta) \qquad (4.2) \\ &+ \gamma \sum_{i=N+1}^{M} \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) \end{aligned}$$

where the first term is the penalized log conditional likelihood of the labeled data under the CRFs, (4.1), and the second line is the negative conditional entropy of the CRFs on the unlabeled data. Here, γ is a tradeoff parameter that controls the influence of the unlabeled data.

This approach resembles the one taken by Grandvalet and Bengio [46] for single variable classification, but here applied to structured CRFs training. The motivation is that minimizing conditional entropy over unlabeled data encourages the algorithm to find putative labelling for the unlabeled data that are mutually reinforcing with the supervised labels; that is, greater certainty on the putative labelling coincides with greater conditional likelihood on the supervised labels, and vice versa. For a single classification variable this criterion has been shown to effectively partition unlabeled data into clusters [46, 105].

To motivate the approach in more detail, consider the overlap between the probability distribution over a label sequence y and the empirical distribution of $\tilde{p}(\mathbf{x})$ on the unlabeled data $\mathcal{D}^u$. The overlap can be measured by the Kullback-Leibler divergence $D(p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})\|\tilde{p}(\mathbf{x}))$. It is well known that Kullback-Leibler divergence (Cover and Thomas 1991) is positive and increases as the overlap between the two distributions decreases. In other words, maximizing Kullback-Leibler divergence implies that the overlap between two distributions is minimized. The total overlap over all possible label sequences can be defined as

$$\begin{aligned}
&\sum_y D(p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})\|\tilde{p}(\mathbf{x})) \\
&= \sum_{\mathbf{y}} \sum_{\mathbf{x}\in\mathcal{D}^u} p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x}) \log \frac{p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})}{\tilde{p}(\mathbf{x})} \\
&= \sum_{\mathbf{x}\in\mathcal{D}^u} \tilde{p}(\mathbf{x}) \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}) \log p_\theta(\mathbf{y}|\mathbf{x})
\end{aligned}$$

which motivates the negative entropy term in (4.2).

The combined training objective (4.2) exploits unlabeled data to improve the CRFs model, as I will show. However, one drawback with this approach is that the entropy regularization term is not concave. To see why, note that the entropy regularizer can be seen as a composition, $h(\theta) = f(g(\theta))$, where $f : \Re^{|\mathcal{Y}|} \to \Re$, $f(g) = \sum_{\mathbf{y}} g_{\mathbf{y}} \log g_{\mathbf{y}}$ and $g_{\mathbf{y}} : \Re^K \to \Re$, $g_{\mathbf{y}}(\theta) = \frac{1}{Z_\theta(\mathbf{x})} \exp\left(\sum_{k=1}^K \theta_k f_k(\mathbf{x},\mathbf{y})\right)$. For scalar θ, the second derivative of a composition, $h = f \circ g$, is given by [19]

$$h''(\theta) \;=\; g'(\theta)^{\top}\nabla^2 f(g(\theta))g'(\theta)+\nabla f(g(\theta))^{\top} g''(\theta)$$

Although f and g_y are concave here, since f is not nondecreasing, h is not necessarily concave. So in general there are local maxima in (4.2).

4.3 An Efficient Training Procedure

As (2) is not concave, many of the standard global maximization techniques do not apply. However, one can still use unlabeled data to improve a supervised CRFs via iterative ascent. To derive an efficient iterative ascent procedure, I need to compute gradient of (4.2) with respect to the parameters θ. Taking derivative of the objective function (4.2) with respect to θ yields Appendix A for the derivation)

$$\begin{aligned} &\frac{\partial}{\partial\theta} RL(\theta) \\ &= \sum_{i=1}^{N} f(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) - \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) \\ &- \frac{\partial}{\partial\theta} U(\theta) \;+\; \gamma \sum_{i=N+1}^{M} \mathrm{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\Big[f(\mathbf{x}^{(i)},\mathbf{y})\Big]\theta \end{aligned} \tag{4.3}$$

The first three items on the right hand side are just the standard gradient of the CRFs objective, $\partial CL(\theta)/\partial\theta$ [69], and the final item is the gradient of the entropy regularizer (the derivation of which is given in Appendix A.

Here, $\mathrm{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\big[f(\mathbf{x}^{(i)},\mathbf{y})\big]$ is the conditional covariance matrix of the features, $f_j(\mathbf{x},\mathbf{y})$, given sample sequence $\mathbf{x}^{(i)}$. In particular, the (j,k)th element of this matrix is given by

$$\begin{aligned}
&\mathrm{cov}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big[f_j(\mathbf{x},\mathbf{y})f_k(\mathbf{x},\mathbf{y})\Big] \\
&= \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})f_k(\mathbf{x},\mathbf{y})\Big) \\
&\quad -\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})\Big)\mathrm{E}_{p_\theta(y|x)}\Big(f_k(\mathbf{x},\mathbf{y})\Big) \\
&= \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x})\Big(f_j(\mathbf{x},\mathbf{y})f_k(\mathbf{x},\mathbf{y})\Big) \qquad (4.4)\\
&\quad -\Big(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x})f_j(\mathbf{x},\mathbf{y})\Big)\Big(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x})f_k(\mathbf{x},\mathbf{y})\Big)
\end{aligned}$$

To efficiently calculate the gradient, I need to be able to efficiently compute the expectations with respect to $\mathbf{y}$ in (4.3) and (4.4). However, this can pose a challenge in general, because there are exponentially many values for $\mathbf{y}$. Techniques for computing the *linear* feature expectations in (4.3) are already well known if $\mathbf{y}$ is sufficiently structured (e.g. $\mathbf{y}$ forms a Markov chain) [69]. However, I now have to develop efficient techniques for computing the *quadratic* feature expectations in (4.4).

For the quadratic feature expectations, first note that the diagonal terms, $j = l$, are straightforward, since each feature is an indicator, I have that $f_j(\mathbf{x},\mathbf{y})^2 = f_j(\mathbf{x},\mathbf{y})$, and therefore the diagonal terms in the conditional covariance are just linear feature expectations $\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})^2\Big) = \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})\Big)$ as before.

For the off diagonal terms, $j \neq l$, however, I need to develop a new algorithm. Fortunately, for structured label sequences, $\mathbf{Y}$, one can devise an efficient algorithm for calculating the quadratic expectations based on nested dynamic programming. To illustrate the idea, I assume that the dependencies of $\mathbf{Y}$, conditioned on $\mathbf{X}$, form a *Markov chain*.

Define one feature for each state pair (y', y), and one feature for each state-observation pair (y, x), which I express with indicator functions $f_{y',y}(\langle u,v\rangle, \mathbf{y}|_{\langle u,v\rangle}, \mathbf{x}) = \delta(\mathbf{y}_u, y')\delta(\mathbf{y}_v, y)$ and $g_{y,x}(v, \mathbf{y}|_v, \mathbf{x}) = \delta(\mathbf{y}_v, y)\delta(\mathbf{x}_v, x)$ respectively. Following [69], I also add special start and stop states, $\mathbf{Y}_0 = $ start and $\mathbf{Y}_{n+1} = $ stop. The conditional probability of a label sequence can now be expressed concisely in a matrix form. For each position j in the observation sequence $\mathbf{x}$, define the $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix random variable $M_j(\mathbf{x}) = [M_j(y', y|\mathbf{x})]$ by

$$\begin{aligned}
M_j(y', y|\mathbf{x}) &= \exp(\Lambda_j(y', y|\mathbf{x})) \quad \text{where} \\
\Lambda_j(y', y|\mathbf{x}) &= \sum_k \lambda_k f_k\left(e_j, \mathbf{y}|_{e_j} = (y', y), \mathbf{x}\right) \\
&\quad + \sum_k \mu_k g_k\left(v_j, \mathbf{y}|_{v_j} = y, \mathbf{x}\right)
\end{aligned}$$

Here e_j is the edge with labels $(\mathbf{Y}_{j-1}, \mathbf{Y}_j)$ and v_j is the vertex with label $\mathbf{Y}_j$.

For each index $j = 0, \cdots, n+1$ define the forward vectors $\alpha_j(\mathbf{x})$ with base case

$$\alpha_0(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \text{start} \\ 0 & \text{otherwise} \end{cases}$$

and recurrence

$$\alpha_j(\mathbf{x}) = \alpha_{j-1}(\mathbf{x}) M_j(\mathbf{x})$$

Similarly, the backward vectors $\beta_j(\mathbf{x})$ are given by

$$\begin{aligned}
\beta_{n+1}(y|\mathbf{x}) &= \begin{cases} 1 & \text{if } y = \text{stop} \\ 0 & \text{otherwise} \end{cases} \\
\beta_j(\mathbf{x}) &= M_{j+1}(\mathbf{x}) \beta_{j+1}(\mathbf{x})
\end{aligned}$$

With these definitions, the expectation of the product of each pair of feature functions, $(f_j(\mathbf{x},\mathbf{y}), f_k(\mathbf{x},\mathbf{y}))$, $(f_j(\mathbf{x},\mathbf{y}), g_k(\mathbf{x},\mathbf{y}))$, and $(g_j(\mathbf{x},\mathbf{y}), g_k(\mathbf{x},\mathbf{y}))$, for $j, k = 1, \cdots, K$, $j \neq k$, can be recursively calculated.

First define the summary matrix

$$M_{s+1,t-1}(y, y'|\mathbf{x}) = \Big(\prod_{l=s+1}^{t-1} M_l(\mathbf{x}) \Big)_{y,y'}$$

Then the quadratic feature expectations can be computed by the following recursion, where the two double sums in each expectation correspond to the two cases depending on which feature occurs first (e_s occuring before e_t).

$$\begin{aligned}
&\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})f_k(\mathbf{x},\mathbf{y})\Big) \\
&= \sum_{\mathbf{x}\in\mathcal{D}^u} \sum_{s,t=1,s<t}^{n+1} \sum_{y',y} f_j\left(e_s, \mathbf{y}|_{e_s} = (y',y), \mathbf{x}\right) \\
&\quad \sum_{y'',y'''} f_k\left(e_t, \mathbf{y}|_{e_t} = (y'',y'''), \mathbf{x}\right) \\
&\quad \alpha_{s-1}(y'|\mathbf{x}) M_s(y',y|\mathbf{x}) M_{s+1,t-1}(y,y''|\mathbf{x}) \\
&\quad M_t(y'',y'''|\mathbf{x}) \beta_t(y'''|\mathbf{x})/Z_\theta(\mathbf{x}) \\
&\quad + \sum_{\mathbf{x}\in\mathcal{D}^u} \sum_{s,t=1,t<s}^{n+1} \sum_{y',y} f_j\left(e_t, \mathbf{Y}|_{e_t} = (y',y), \mathbf{x}\right) \\
&\quad \sum_{y'',y'''} f_k\left(e_s, \mathbf{Y}|_{e_s} = (y'',y'''), \mathbf{x}\right) \\
&\quad \alpha_{t-1}(y'''|\mathbf{x}) M_t(y''',y''|\mathbf{x}) M_{t+1,s-1}(y'',y'|\mathbf{x}) \\
&\quad M_s(y',y|\mathbf{x}) \beta_t(y|\mathbf{x})/Z_\theta(\mathbf{x})
\end{aligned}$$

$$\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})g_k(\mathbf{x},\mathbf{y})\Big)$$

$$= \sum_{\mathbf{x}\in\mathcal{D}^u}\sum_{s,t=1,s\le t}^{n+1}\sum_{y',y} f_j\left(e_s,\mathbf{y}|_{e_s}=(y',y),\mathbf{x}\right)$$

$$\sum_{y''} g_k\left(v_t,\mathbf{Y}|_{v_t}=y'',\mathbf{x}\right)\alpha_{s-1}(y'|\mathbf{x})M_s(y',y|\mathbf{x})$$

$$M_{s+1,t-1}(y,y''|x)\beta_t(y''|\mathbf{x})/Z_\theta(\mathbf{x})$$

$$+\sum_{x\in\mathcal{D}^u}\sum_{s,t=1,t<s}^{n+1}\sum_{y',y} f_j\left(e_t,\mathbf{y}|_{e_t}=(y',y),\mathbf{x}\right)$$

$$\sum_{y''} g_k\left(v_s,\mathbf{y}|_{v_s}=y'',\mathbf{x}\right)\alpha_{t-1}(y''|\mathbf{x})$$

$$M_{t+1,s-1}(y'',y'|x)M_s(y',y|\mathbf{x})\beta_t(y|\mathbf{x})/Z_\theta(\mathbf{x})$$

$$\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(g_j(\mathbf{x},\mathbf{y})g_k(\mathbf{x},\mathbf{y})\Big)$$

$$= \sum_{\mathbf{x}\in\mathcal{D}^u}\sum_{s,t=1,s<t}^{n+1}$$

$$\sum_{y'} g_j\left(v_s,\mathbf{y}|_{v_s}=y',\mathbf{x}\right)\sum_{y} g_k\left(v_t,\mathbf{y}|_{v_t}=y,\mathbf{x}\right)$$

$$\frac{\alpha_{s-1}(y'|\mathbf{x})M_{s+1,t-1}(y',y|\mathbf{x})\beta_t(y|\mathbf{x})}{Z_\theta(\mathbf{x})}$$

$$+\sum_{\mathbf{x}\in\mathcal{D}^u}\sum_{s,t=1,t<s}^{n+1}$$

$$\sum_{y'} g_j\left(v_t,\mathbf{y}|_{v_t}=y',\mathbf{x}\right)\sum_{y'} g_k\left(v_s,\mathbf{y}|_{v_s}=y,\mathbf{x}\right)$$

$$\frac{\alpha_{t-1}(y|\mathbf{x})M_{t+1,s-1}(y,y'|\mathbf{x})\beta_t(y'|\mathbf{x})}{Z_\theta(\mathbf{x})}$$

The computation of these expectations can be organized in a trellis, as illustrated in Figure 4.1.

Once I obtain the gradient of the objective function (2), I use limited-memory L-BFGS, a quasi-Newton optimization algorithm [82], to find the local maxima with the initial value being set to be the optimal solution of the supervised CRFs on labeled data.

4.4 Time and Space Complexity

The time and space complexity of the semi-supervised CRFs training procedure is greater than that of standard supervised CRFs training, but nevertheless remains a small degree polynomial in the size of the training data. Let

m_l	$=$	size of the labeled set
m_u	$=$	size of the unlabeled set
n_l	$=$	labeled sequence length
n_u	$=$	unlabeled sequence length
n_t	$=$	test sequence length
s	$=$	number of states
c	$=$	number of training iterations.

Then the time required to classify a test sequence is $O(n_t s^2)$, independent of training method, since the Viterbi decoder needs to access each path.

For training, supervised CRFs training requires $O(cm_l n_l s^2)$ time, whereas semi-supervised CRFs training requires $O(cm_l n_l s^2 + cm_u n_u^2 s^3)$ time. The additional cost for semi-supervised training arises from the extra nested loop required to calculated the quadratic feature expectations, which introduces in an additional $n_u s$ factor.

However, the space requirements of the two training methods are the same. That is, even though the covariance matrix has size $O(K^2)$, there is never any need to store the entire matrix in memory. Rather, since I only need to compute the product of the covariance with θ, the calculation can be performed iteratively without using extra space beyond that already required by supervised CRFs training.

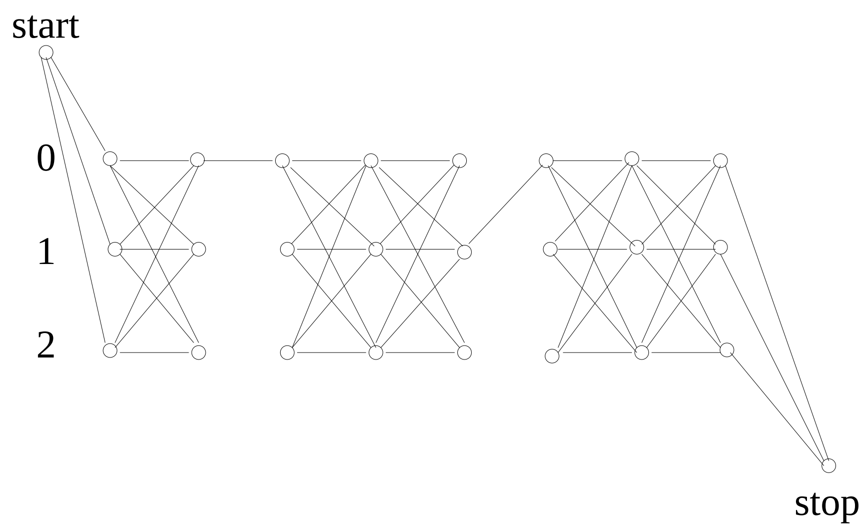

Figure 4.1: Small Trellis for computing the expectation of a feature product over a pair of feature functions, f_{00} vs f_{10}, where the feature f_{00} occurs first. This leads to one double sum.

4.5 Experimental Results

I have developed the new semi-supervised training procedure to address the problem of information extraction from biomedical text, which has received significant attention in the past few years.

I have specifically focused on the problem of identifying explicit mentions of gene and protein names

Recently, McDonald and Pereira [84] have obtained interesting results on this problem by using a standard supervised CRFs approach. However, our contention is that stronger results could be obtained in this domain by exploiting a large corpus of un-annotated biomedical text to improve the quality of the predictions, which I now show.

Given a biomedical text, the task of identifying gene mentions can be interpreted as a tagging task, where each word in the text can be labeled with a tag that indicates whether it is the beginning of gene mention (B), the continuation of a gene mention (I), or outside of any gene mention (O). To compare the performance of different taggers learned by different

Table 4.1: Performance of the supervised CRF

	Precision	Recall	F-Measure
Set B	0.8	0.36	0.50
Set C	0.77	0.29	0.43
Set D	0.74	0.30	0.43

mechanisms, one can measure the precision, recall and F-measure, given by

$$\begin{aligned} \textit{precision} &= \frac{\#\ \textit{correct predictions}}{\#\ \textit{predicted gene mentions}} \\ \textit{recall} &= \frac{\#\ \textit{correct predictions}}{\#\ \textit{true gene mentions}} \\ \textit{F-measure} &= \frac{2\times \textit{precision}\times \textit{recall}}{\textit{precision}+\textit{recall}} \end{aligned}$$

In my evaluation, I compared the proposed semi-supervised learning approach to the state of the art supervised CRFs of McDonald and Pereira (2005), and also to self-training [25, 135], using the same feature set as [84]. The CRFs training procedures, supervised and semi-supervised, were run with the same regularization function, $U(\theta) = \|\theta\|^2/2$, used in (McDonald and Pereira 2005).

First I evaluated the performance of the semi-supervised CRFs in detail, by varying the ratio between the amount of labeled and unlabeled data, and also varying the tradeoff parameter γ. I choose a labeled training set A consisting of 5448 words, and considered alternative unlabeled training sets, B(5210 words), C(10,208 words), and D (25,145 words), consisting of the same, 2 times and 5 times as many sentences as A respectively. All of these sets were disjoint and selected randomly from the full corpus, the smaller one in [84], consisting of 184,903 words in total. To determine sensitivity to the parameter γ I examined a range of discrete values $0, 0.1, 0.5, 1, 5, 10, 20, 50$.

In the first experiment, I train the CRFs models using labeled set A and unlabeled sets B, C and D respectively. Then test the performance on the sets B, C and D respectively The results of The evaluation are shown in Table 1. The performance of the supervised CRFs algorithm, trained only on the labeled set A, is given on the first row in Table 1 (corresponding to $\gamma = 0$). By comparison, the results obtained by the semi-supervised CRFs on the held-out sets B, C and D are given in Table 1 by increasing the value of γ.

Table 4.2: Performance of the semi-supervised CRFs obtained on the held-out sets B, C and D. P=Precision, R=Recall, F=F-Measure

	Test Set B			Test Set C			Test Set D		
γ	P	R	F	P	R	F	P	R	F
0	0.80	0.36	0.50	0.77	0.29	0.43	0.74	0.30	0.43
0.1	0.82	0.4	0.54	0.79	0.32	0.46	0.74	0.31	0.44
0.5	0.82	0.4	0.54	0.79	0.33	0.46	0.74	0.31	0.44
1	0.82	0.4	0.54	0.77	0.34	0.47	0.73	0.33	0.45
5	0.84	0.45	0.59	0.78	0.38	0.51	0.72	0.36	0.48
10	0.78	0.46	0.58	0.66	0.38	0.48	0.66	0.38	0.47

The results of this experiment demonstrate quite clearly that in most cases the semi-supervised CRFs obtains higher precision, recall and F-measure than the fully supervised CRFs, yielding a 20% improvement in the best case.

In the second experiment, again I train the CRFs models using labeled set A and unlabeled sets B, C and D respectively with increasing values of γ, but I test the performance on the held-out set E which is the full corpus minus the labeled set A and unlabeled sets B, C and D. The results of evaluation are shown in Table 2 and Figure 2. The blue line in Figure 2 is the result of the supervised CRFs algorithm, trained only on the labeled set A. In particular, by using the supervised CRFs model, the system predicted 3334 out of 7472 gene mentions, of which 2435 were correct, resulting in a precision of 0.73, recall of 0.33 and F-measure of 0.45. The other curves are those of the semi-supervised CRFs.

The results of this experiment demonstrate quite clearly that the semi-supervised CRFs simultaneously increase both the number of predicted gene mentions and the number of correct predictions, thus the precision remains almost the same as the supervised CRFs, and the recall increases significantly.

Both experiments as illustrated in Figure 2 and Tables 1 and 2 show that clearly better results are obtained by incorporating additional unlabeled training data, even when evaluating on disjoint testing data (Figure 2). The performance of the semi-supervised CRFs is not overly sensitive to the tradeoff parameter γ, except that γ cannot be set too large.

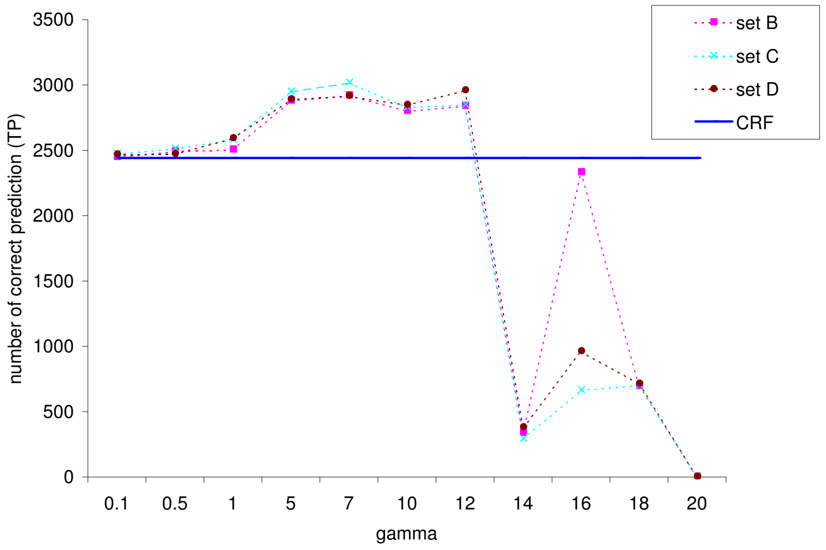

Figure 4.2: Performance of the supervised and semi-supervised CRFs. The sets B, C and D refer to the unlabeled training set used by the semi-supervised algorithm.

For completeness, I also compared the results to the self-learning algorithm, which has commonly been referred to as bootstrapping in natural language processing and originally popularized by the work of Yarowsky in word sense disambiguation [2, 135]. In fact, similar ideas have been developed in pattern recognition under the name of the decision-directed algorithm [36], and also traced back to 1970s in the EM literature [25]. The basic algorithm works as follows:

1. Given $\mathcal{D}^l$ and $\mathcal{D}^u$, begin with a seed set of labeled examples, $\mathcal{D}^{(0)}$, chosen from $\mathcal{D}^l$.
2. For $m = 0, 1, \cdots$
   (a) Train the supervised CRFs on labeled examples $\mathcal{D}^{(m)}$, obtaining $\theta^{(m)}$.

Table 4.3: Performance of the semi-supervised CRFs trained by using unlabeled sets B, C and D with Test Set E. P=num of predictions, C=num of correct predictions

	B		C		D	
γ	# P	# C	# P	# C	# P	# C
0.1	3345	2446	3376	2470	3366	2466
0.5	3413	2489	3450	2510	3376	2469
1	3446	2503	3588	2580	3607	2590
5	4089	2878	4206	2947	4165	2888
10	4450	2799	4762	2827	4778	2845

(b) For each sequence $\mathbf{x}^{(i)} \in \mathcal{D}^u$, find $\mathbf{y}^{(i)}_{(m)} = \arg\max_{\mathbf{y}} p_{\theta^{(m)}}(\mathbf{y}|\mathbf{x}^{(i)})$ via Viterbi decoding or other inference algorithm, and add the pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}_{(m)})$ to the set of labeled examples (replacing any previous label for $\mathbf{x}^{(i)}$ if present).

(c) If for each $\mathbf{x}^{(i)} \in \mathcal{D}^u$, $\mathbf{y}^{(i)}_{(m)} = \mathbf{y}^{(i)}_{(m-1)}$, stop; otherwise $m = m + 1$, iterate.

I implemented this self training approach and tried it in for experiments. Unfortunately, I were not able to obtain any improvements over the standard supervised CRFs with self-learning, using the sets $\mathcal{D}^l = A$, and $\mathcal{D}^u \in \{B, C, D\}$. The semi-supervised CRFs remains the best of the approaches I have tried on this problem.

4.6 Conclusions

I have presented a new semi-supervised training algorithm for CRFs, based on extending minimum conditional entropy regularization to the structured prediction case. The approach is motivated by the information-theoretic argument [46, 105] that unlabeled examples can provide the most benefit when classes have small overlap. An iterative ascent optimization procedure was developed for this new criterion, which exploits a nested dynamic programming approach to efficiently compute the covariance matrix of the features.

I applied the new approach to the problem of identifying gene name occurrences in biological text, exploiting the availability of auxiliary unlabeled data to improve the performance of the state of the art supervised CRFs approach in this domain.

The semi-supervised CRFs approach shares all of the benefits of the standard CRFs training, including the ability to exploit arbitrary features of the inputs, while obtaining improved accuracy through the use of unlabeled data. The main drawback is that training time is increased because of the extra nested loop needed to calculate feature covariances. Nevertheless, the algorithm is sufficiently efficient to be trained on unlabeled data sets that yield a notable improvement in classification accuracy over standard supervised training. To further accelerate the training process of the semi-supervised CRFs, I may apply stochastic gradient optimization method with adaptive gain adjustment as proposed by [119].

4.7 Appendix A: Deriving the gradient of the entropy

I wish to show that

$$\begin{aligned}&\frac{\partial}{\partial\theta}\left(\sum_{i=N+1}^{M}\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\log p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\right)\\ &= \sum_{i=N+1}^{M}\mathrm{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\left[f(\mathbf{x}^{(i)},\mathbf{y})\right]\theta\end{aligned} \tag{4.5}$$

First, note that some simple calculation yields

$$\frac{\partial \log Z_\theta(\mathbf{x}^{(i)})}{\partial\theta_j} = \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})$$

and

$$\begin{aligned}\frac{\partial p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}{\partial\theta_j} &= \frac{\partial}{\partial\theta_j}\left(\frac{\exp\left(\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle\right)}{Z_\theta(\mathbf{x}^{(i)})}\right)\\ &= p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\\ &\quad - p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\end{aligned}$$

Therefore

$$
\begin{aligned}
&\frac{\partial}{\partial \theta_j}\left(\sum_{i=N+1}^{M}\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\right) \\
&= \sum_{i=N+1}^{M} \frac{\partial}{\partial \theta_j}\Bigg(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle \\
&\qquad\qquad - \log Z_\theta(\mathbf{x}^{(i)})\Bigg) \\
&= \sum_{i=N+1}^{M}\Bigg(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y}) \\
&\qquad + \sum_{\mathbf{y}} \frac{\partial p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}{\partial \theta_j}\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle \\
&\qquad - \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\Bigg) \\
&= \sum_{i=N+1}^{M}\Bigg(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle \\
&\qquad -[\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle] \\
&\qquad [\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})]\Bigg) \\
&= \sum_{i=N+1}^{M}\Bigg(\sum_{k}\theta_k\Big[\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y}) f_k(\mathbf{x}^{(i)},\mathbf{y}) \\
&\qquad -[\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_k(\mathbf{x}^{(i)},\mathbf{y})] \\
&\qquad [\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})]\Big]\Bigg)
\end{aligned}
$$

In the vector form, this can be written as (5)

$$\frac{\partial}{\partial\theta}\left(\sum_{i=N+1}^{M}\sum_{y}p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\log p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\right)$$
$$=\sum_{i=N+1}^{M}\mathrm{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\Big[f(\mathbf{x}^{(i)},\mathbf{y})\Big]\theta$$

Chapter 5

Tree Decomposition for Protein Structure Prediction

By using a 2-D contact graph, threading problem can be formulated as an alignment between chain structure graph (sequence) and contact graph, while structure alignment can be formulated as an alignment between two contact graphs. Figure 5.1 show the relationship of the two problems.

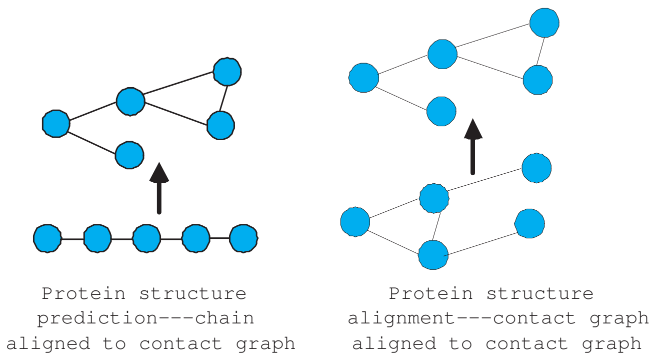

Figure 5.1: Example of protein structure prediction alignment

These two problem can be treated as one frequency assignment problem and tree decomposition algorithm might be a good choice since it is an exact solution and also is efficient with the small tree width.

As we discuss in chapter 2, the width of a tree decomposition is a key factor in determining the computational complexity of all the tree-decomposition based algorithms. The smaller the width of a tree decomposition is, the more efficient the tree decomposition based algorithms. Also we have introduced six different decomposition algorithms.

Table 5 lists the performance of six tree-decomposition methods for the decomposition of 5280 template contact graphs. Any two templates share no more than 40% sequence identity. As shown in this table, the minimum-degree and the minimum-discrepancy methods are the best for the decomposition of the template contact graphs. This table also indicates that more than 98% template contact graphs can be decomposed into components containing no more than 6 vertices if a good decomposition method is employed.

Table 5.1: Performance of six different tree decomposition methods. The numbers in this table are the percentage of template contact graphs with a given treewidth.

tree decomposition method	tree width					average
	≤ 2	3	4	5	≥ 6	tree width
minimum vertex separator	18.13	10.53	11.86	12.33	47.16	5.676
two-way-1/2-triangle	18.86	20.25	16.52	22.92	21.46	4.121
two-way-2/3-triangle	21.19	14.53	29.72	19.05	15.51	3.874
minimum-degree	22.97	34.41	31.12	9.53	1.97	3.198
minimum-width	22.92	27.35	25.72	13.84	10.34	3.565
minimum-discrepancy	22.95	34.39	31.93	9.09	1.63	3.185

We can see by using the minimum-degree or minimum-discrepancy method, we can obtain a very small average tree width.

The low tree width property of the contact graph suggests that the tree decomposition algorithm is very suitable for protein problems because it provides an exact solution while being very efficient in this case. This algorithm had first been used in protein side-chain packing [126]. In the next two chapters, I will show how this algorithm can be used on protein structure prediction and structure alignment. The research for structure prediction was published in [128] and structure alignment was published in [129].

In this chapter I propose a tree decomposition of protein structure that can be used to efficiently solve protein threading for backbone prediction. I model this problem as a geometric neighborhood graph labelling problem. Theoretically, one can have a low-degree polynomial time algorithm to decompose a geometric neighborhood graph $G = (V, E)$ into components with size $O(|V|^{\frac{2}{3}} \log |V|)$. The computational complexity of the tree-decomposition based graph labelling algorithms is $O(|V|\Delta^{tw+1})$ where Δ is the average number of possible labels for each vertex and $tw(= O(|V|^{\frac{2}{3}} \log |V|))$ the tree width of G. Empirically, tw is very small and the tree-decomposition method can solve these the problem very efficiently. I also compare the computational efficiency of the tree-decomposition approach with the linear programming approach and identify the conditions under which tree-decomposition is more efficient than linear programming. My experimental results indicate that the tree-decomposition approach is often more efficient. This work was published in [128].

5.1 Problem Formulation-Protein Threading

A structural template can be modeled using a template contact graph as follows. The primary structure of a template is parsed as a linear series of cores with a connecting loop between two adjacent cores. Cores are the most conserved segments in a protein structure. When aligning a protein sequence with structure to be predicted to a template, alignment gaps are confined to loops. The biological justification is that cores are so conserved that the chance of insertions or deletions within them is very slim. I consider only interactions between residues in the cores. It is generally believed that interactions involving loop residues can be ignored as their contribution to fold recognition is relatively insignificant. We say that an interaction exists between two residues if the spatial distance between their C_β atoms is within $7\dot{A}$ and they are at least 4 residues apart along the template sequence. We say that an interaction exists between two cores if there exists at least one inter-residue interaction between the two cores. One can model a protein structural template using a template contact graph $G = (V, E)$. Each template core is represented by a vertex in V. There is one edge between two cores if and only if an interaction exists between them.

Therefore, the protein threading problem can be formulated as follows [131, 130]. Let $D[i]$ denote the set of possible alignment positions for core i. For each possible alignment position $l \in D[i]$, there is an associated singleton score, denoted by $S_i(l)$. This singleton score measures how well to align core i to sequence position l. In the energy function, $S_i(l)$ includes mutation score, environmental fitness score and secondary structure score. For any two alignment positions $l \in D[i]$ and $k \in D[j]$ $(i \neq j)$, there is also an associated pairwise score, denoted by $P_{i,j}(l,k)$, if there is an edge between core i and core j. $P_{i,j}(l,k)$ is the interaction score between cores i and j when their alignment positions are l and k, respectively. In the sequence-template alignment, there is no crossover allowed. That is, if $i < j$, then k must be larger than l plus the length of core i. In order to guarantee a valid alignment, one can set $P_{i,j}(l,k)$ to be $+\infty$ when crossover occurs. For the alignment involved with the loop regions of a template, some gaps may exist. In order to penalize gaps, the scoring function also contains some gap penalty. Assume that core i is aligned to sequence position $A(i)$. The quality of this sequence-template alignment is measured by the following energy function.

$$E(G) = \sum_{i \in V} S_i(A(i)) + \sum_{i \neq j, (i,j) \in E} P_{i,j}(A(i), A(j)) \tag{5.1}$$

The smaller the system energy $E(G)$ is, the better the sequence-template alignment.

5.2 Experimental Results

This section compares the computational efficiency of the tree-decomposition approach and the linear programming approach to protein structure prediction and identifies the condition under which the tree-decomposition based approach is more efficient than the linear programming approach. Both approaches can solve the problems to their optimal solutions. Therefore, both approaches have the same prediction accuracy.

To compare the computational efficiency of the tree-decomposition based approach and the linear programming approach, I randomly chose 1000 structural templates from RAPTOR's template database and 100 sequences from the Lindahl's benchmark [78], respectively. Any two structural templates share no more than 40% sequence identity, so do any two sequences. I threaded each sequence to each template using both approaches. Tested on a PC Linux box with a 1.7GHz CPU, it takes the linear programming approach approximately 100 hours to finish all the 100,000 threading pairs and the tree-decomposition approach approximately 58 hours. In my experiment, I used the linear program solver CLP in the COIN package to solve all the linear programs.

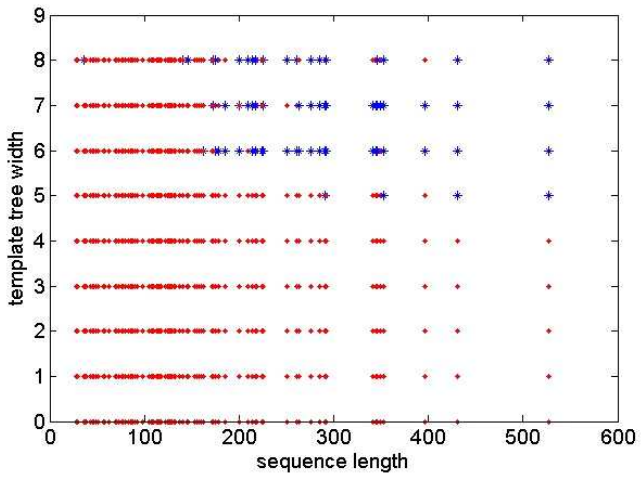

Figure 5.2: Computational efficiency of the tree-decomposition based approach and the linear programming approach to the protein threading problem. A '*' indicates that the linear programming approach is better, while a '.' indicates that the tree-decomposition based approach is better.

I further examined the conditions under which the tree-decomposition approach is better than the linear programming approach. The running time of the tree-decomposition based approach is related to both the tree width of the tree decomposition, the template length and the sequence length. I calculated the average running time of threading a given sequence to all the templates with the same tree width. In figure 5.2, I use a '*' to indicate that the linear programming method is more efficient than the tree-decomposition based method and a '.' to indicate the reverse situation. As shown in Figure 5.2, the tree-decomposition based method runs faster than the linear programming approach when the tree width is smaller than 5. From this figure we can see that when the tree width is smaller than 5, the tree-decomposition method is always more efficient than the linear programming method. If the tree width is equal to or large than 5, the linear programming method is better if the sequence length is large. When the tree width is equal to 5, the tree-decomposition based method is better if the sequence has no more than 350 residues. When the tree width is equal to 6 or 7, the tree-decomposition based method is better if the sequence length is less than 170. When the tree width is equal to 8, the tree-decomposition based method is better if the sequence length is less than 150. In a summary, the trend is that when the tree width is big and the sequence is long, then the linear programming method is better than the tree-decomposition method, otherwise the tree-decomposition method is better.

One can further improve the computational efficiency by combining these two methods. That is, I use the tree-decomposition based approach to the protein threading problem when one of the following conditions is satisfied: (i) the template tree width is smaller than 5; or (ii) the template tree width is equal to 5 and the sequence has no more than 350 residues; or (iii) the template tree width is less than 9 and the sequence has no more than 150 residues. Otherwise, I use the linear programming approach. Then the total running time of threading all the 100,000 pairs can be improved to 52 hours, which is approximately half of the running time of the linear programming approach.

Both the tree-decomposition based approach and the linear programming approach have the same prediction accuracy. I compared the prediction accuracy of these two approaches using thirty CASP6 test proteins, which were released from June 2004 to August 2004. These test proteins are available at the CASP6 website. The template database was generated from the PDB database in April 2004. In total there are about 5000 templates and any two templates share no more than 40% sequence identity. Table 5.2 lists the best template predicted for each test protein using two different threading approaches (i.e., linear programming and tree-decomposition). Both approaches generate the same top template for each test protein.

5.3 Conclusions

Here I used a tree-decomposition based approach instead of linear programming approach to solve protein structure prediction problems. Both approaches have their own advantages, but the tree-decomposition based approach is more efficient than the linear programming approach. We also obtained the rule by which for a given threading pair, one can easily

Table 5.2: Top templates chosen by both the linear programming approach and the tree-decomposition approach for the thirty CASP6 test proteins.

test protein	t0200	t0201	t0202	t0203	t0204	t0205	t0206	t0207
top template	1hp1a	1unnc	1ko7a	1rxxa	1guqa	1h0xa	1mv3a	1h75a
test protein	t0210	t0211	t0212	t0213	t0214	t0215	t0216	t0217
top template	1j58a	1gvna	1fw9a	1ukfa	1nwaa	1vjqa	1iruf	1i60a
test protein	t0220	t0221	t0222	t0223	t0224	t0225	t0226	t0227
top template	1nvta	1dnya	1o60a	1nox	1jx7a	1n7ha	1c7qa	1mq0a

choose the approach that is the most efficient for aligning this pair. Combing these two approaches, one can achieve a better computational efficiency than any single method.

Chapter 6

Tree Decomposition for Protein Structure Alignment

I have also proposed a tree decomposition algorithm for aligning two protein structures. The result is achieved by decomposing the protein structure using tree decomposition and discretizing the rigid-body transformation space. I implemented the proposed algorithm and preliminary experimental results indicate that on a Linux PC, it takes from ten minutes to one hour to align two proteins with approximately 100 residues. This was joint worked with Jinbo Xu and was published in [129].

6.1 Problem Formulation

Given two proteins A and B, each is represented by a contact map graph. There is a contact between two residues if their distance is no more than D_u. The optimal alignment between A and B is an alignment such that the number of equivalent contact edges is maximized and after the two proteins are superimposed, the Euclidean distance between two equivalent residues is no more than a threshold D_c.

Let $E[A]$ and $E[B]$ denote the set of contacts in proteins A and B, respectively. For any residue u in A, let $M(u)$ denote its equivalent residue in B. If there is no equivalent residue for u, then $M(u) = \phi$. The protein structure alignment problem is to maximize the following objective function:

$$\sum_{u,v \in V[A], u<v} f(u, v, M(u), M(v)) \tag{6.1}$$

where

$$f(u, v, M(u), M(v)) = \begin{cases} -\infty & M(u) = M(v) \neq \phi \\ 1 & (u, v) \in E[A], (M(u), M(v)) \in E[B] \\ 0 & otherwise \end{cases} \tag{6.2}$$

Note that $f(u, v, M(u), M(v)) = -\infty$ is used to avoid two different residues u and v being aligned to the same residue in B.

We can further generalize the above problem to the case where protein A is represented as a contact graph and protein B as a distance matrix. That is, $f(u, v, M(u), M(v)) = h(|u-v|, |M(u)-M(v)|)$ when $(u, v) \in E[A]$ where $h(x, y)$ takes two contact distances and outputs a positive value. The closer these two contact distances, the higher the output.

The algorithm described here can solve the protein structure alignment problem with (7.6) as the objective function. To enforce sequential order in the alignment, one can set $f(u, v, M(u), M(v))$ to be $-\infty$ if $u < v$ while $M(u) > M(v)$.

6.2 Tree-decomposition for Structure Alignment Algorithm

Here I describe a tree-decomposition based algorithm for the optimal protein structure alignment problem, assuming that the positions of both proteins are fixed. This algorithm has an exponential time complexity and will be used as a subroutine of the final algorithm described in the following section.

In (7.6), in order to detect if two residues in A align to the same residue in B, one has to enumerate all the residue pairs in A. To be able to easily detect if two residues in protein A are aligned to the same residue in B or not, I extend the contact graph $G[A]$ to $G'[A] = (V[A], E'[A])$ by adding more edges to $G[A]$. Besides all the edges in $G[A]$, I add extra edge (u, v) to $G'[A]$ if the distance between u and v is less than $2D_c$ but more than D_u. Therefore, for any two residues u and v in A, if there is no edge between them in $G'[A]$, then they cannot align to the same residue in B since the distance between two equivalent residues is no more than D_c. Using the extended graph, one can revise the objective function in (7.4) as follows:

$$\sum_{(u,v)\in E'[A]} f(u, v, M(u), M(v)) \tag{6.3}$$

where

$$f(u, v, M(u), M(v)) = \begin{cases} -\infty & M(u) = M(v) \neq \phi \\ 1 & (u, v) \in E[A], (M(u), M(v)) \in E[B] \\ 0 & otherwise \end{cases}$$

Since now we only need to enumerate all the edges in $G'[A]$ to calculate the objective function in (6.3), we can perform a tree-decomposition on graph $G'[A]$ and then use the same tree-decomposition based algorithm as described chapter 2.4 to maximize the objective function. Any two residues in A that might align to the same residue in B appear simultaneously in at least one tree decomposition component. So when calculating on this tree decomposition component, we can detect if these two residues are aligned to the same residue or not. Using the same proof technique as in paper [126], one can prove that the tree width of $G'[A]$ is no more than $O(\frac{\max\{2D_c, D_u\}}{D_l} n^{2/3} \lg n)$. Since the distance between two matched residues is no more than D_c, each residue in A can be aligned to at most $O\left((1 + \frac{2D_c}{D_l})^3\right)$ residues in B. So we have the following theorem.

Theorem 2 *Let A and B be two protein structures in $\Re^3$. Assume that the spatial positions of A and B are fixed and the distance between two equivalent residues is no more than D_c. There is an algorithm with time complexity $O(n2^{tw \lg \Delta})$ generating the optimal non-sequential alignment between A and B, where n is the protein size, $\Delta = O\left((1 + \frac{2D_c}{D_l})^3\right)$, and $tw = O(\frac{\max\{2D_c, D_u\}}{D_l} n^{2/3} \lg n)$.*

Assume that protein A is inscribed in a minimal axis-parallel 3D rectangle and the widths along each dimension are W_x, W_y, and W_z respectively. The following lemma gives another upper bound on the running time of the tree-decomposition based algorithm.

Let A and B be two protein structures in $\Re^3$. Assume that the spatial positions of A and B are fixed and the distance between two equivalent residues is no more than D_c. There is an algorithm with time complexity $O(n2^{tw \lg \Delta})$ generating the optimal non-sequential alignment between A and B, where n is the protein size, $\Delta = O\left((1 + \frac{2D_c}{D_l})^3\right)$, and $tw = O(\frac{\max\{2D_c, D_u\}}{D_l^3} \min\{W_x W_y, W_x W_z, W_y W_z\})$.

6.3 Structure Alignment with Rigid-body Transformations

In this section, we assume that we can move protein A in any way and the position of protein B is fixed. We are going to find the best transformation of A such that the objective function in Eq. 7.4 is maximized. Kolodny and Linial [67] achieved a PTAS algorithm for the coordinate based structure alignment problem by discretizing the rigid-body transformation space into a polynomial number of discrete transformations. We will present a similar but more involved discretization technique for our problem.

A rigid-body transformation consists of two steps: rotation and translation. Mathematically, it can be represented by a triple (w, θ, t), where w is a normalized vector in $\Re^3$, θ the rotation angle and t the translation. The vector w and the angle θ form a quaternion, which is the classic representation for rotation. The normalized vector w is the unit axis around which an object is rotated by θ. Assume $\hat{v}$ to be the resultant vector for rotating a vector v by an angle of θ around a unit axis w. Then $\hat{v}$ can be calculated using the following formula:

$$\hat{v} = w(v \cdot w) + (v - w(v \cdot w))cos(\theta) + (v \times w)sin(\theta) \tag{6.4}$$

where $\cdot$ is the dot product of two vectors and $\times$, the cross product. According to Eq. 7.12, if the unit axis w is changed by a small degree δw, then $|\hat{v}|$ will be changed by $O(|v||\delta w|)$. If the rotation angle θ is changed by $\delta\theta$, then $|\hat{v}|$ will be changed by $O(|v||\delta\theta|)$. Without loss of generality, we can assume that the unit axis w originates at the center point of a protein structure. Then $|v| \leq R$ where R is the radius of a protein structure. A small change in the unit axis w by ϵ/R or the rotation angle θ by ϵ/R will change $|\hat{v}|$ by at most ϵ. All the unit axes form the surface of a sphere with radius 1, and the rotation angle ranges from 0 to 2π.

For any given vector v, a translation t will lead to a new vector $\hat{v} = v + t$. Therefore, a small change in the translation t by $(\epsilon, \epsilon, \epsilon)$ will change $|\hat{v}|$ by at most $O(\epsilon)$. Assume that a protein structure A is enclosed in a rectangle with dimensions $W_x(A)$, $W_y(A)$ and $W_z(A)$. Then all the possible translations between proteins A and B are in a rectangle with dimensions $W_x(A) + W_x(B)$, $W_y(A) + W_y(B)$, and $W_z(A) + W_z(B)$.

Since a small change in the transformation will not greatly change the spatial position of protein A, we can discretize the whole transformation space into a polynomial number of possible transformations. By working on these possible discrete transformations, we can find an alignment between two proteins with an alignment score very close to the optimal. In fact, we can find all the possible transformations that lead to a near-optimal alignment.

6.4 Experimental Results

The algorithm is implemented on a cluster of Linux PCs with 2.5 GHz CPU. In total, I used 15 proteins from two different folds in the test set described in [22] to test the algorithm. I set the contact distance cutoff D_u to 6.75 $\dot{A}$ and the maximum distance between two matched residues D_c to 3.0 $\dot{A}$. In doing structure alignment, I always fix protein B and transform protein A. The space of unit rotation axis is discretized into a 36×18 longitude-latitude grid. The rotation angle is evenly discretized into 36 possible angles. The translation space is discretized into $35 \times 35 \times 35$ discrete points. That is, if we fix the center of protein B to the origin, then the possible center positions of protein A form a set $\{(x/2, y/2, z/2)| -17 \leq x \leq 17, -17 \leq y \leq 17, -17 \leq z \leq 17\}$. I start from $(0, 0, 0)$ and gradually increase the distance between two protein centers to search for the best translation position. In total, the rigid-body transformation space is discretized into $1,000,188,000$ discrete transformations.

Currently, only the non-sequential alignment result is tested. In our implementation, before calling the tree decomposition algorithm, we estimate the maximum number of aligned contacts. If this estimation is no more than the current best result, then the tree decomposition algorithm would not be called so that we can save some computational time. Because of this, aligning two similar proteins is faster than aligning two dissimilar proteins since the algorithm can obtain a good alignment between two similar proteins in an early stage due to our translation space search strategy. Tables 6.1, 6.2 and 6.3 show the detailed alignment results of some protein pairs. The running time of aligning one protein pair ranges from ten minutes to one hour. According to Caprara *et. al.* [22], for the contact distance threshold 6.75 $\dot{A}$, one can cluster two proteins into the same fold if the number of aligned contacts is at least 0.559 times $\min\{c_A, c_B\}$ where c_A and c_B are the numbers of contacts of both proteins, respectively. The experimental results comply with this criterion very well. However, to achieve the maximum number of aligned contacts, $D_c = 3.0\dot{A}$ may not be big enough for some protein pairs. For example, we need a bigger D_c to obtain more aligned contacts between 1booa and 1dbwa although $D_c = 3.0\dot{A}$ gives a very good alignment between 2pcy and 2plt. I plan to investigate the cutoff value of D_c further. While the sequential order in the alignment is not required, there are almost no sequential disorders in the generated alignment if two proteins are in the same class.

Table 6.1: Structure alignments of some proteins with the Flavodoxin-like fold.

protein (A)	protein (B)	# contacts (A)	# contacts (B)	time (s)	# aligned contacts
1booa	1dbwa	441	457	2244	249
1booa	1nat	441	435	2268	279
1booa	1qmpc	441	452	1604	317
1nat	1boob	435	444	1650	262
1nat	1dbwa	435	457	1315	285
1nat	4tmya	435	446	1626	351
1qmpc	1boob	461	444	2277	332
1qmpc	4tmya	461	446	1044	373
4tmya	1boob	446	444	1501	289

6.5 Conclusions

I have presented a tree-decomposed algorithm for the contact map-based protein structure alignment problem, which has been proven to be *NP*-hard. The time complexity is polynomial in the protein size and exponential with respect to several parameters, which usually

Table 6.2: Structure alignments of some proteins with Cupredoxins fold.

protein (A)	protein (B)	# contacts (A)	# contacts (B)	time (s)	# aligned contacts
1bawa	1byob	387	350	799	306
1bawb	1byoa	389	355	795	307
1bawa	1pla	387	340	1309	298
1byoa	1kdi	355	361	1746	257
1byob	1nin	350	376	1679	231
1byob	1kdi	350	361	1478	264
1pla	2b3ia	340	341	1188	226
2b3ia	2pcy	341	357	1053	265
2b3ia	2plt	341	367	1010	293
2pcy	2plt	357	367	527	343

can be treated as constants. However, the method proposed here might not be useful for everyday structure alignment since while theoretically significant, the computational time complexity is still expensive. A tool based on this method can be used as a benchmark to evaluate the performance of other heuristic-based structure alignment algorithms. The experimental results reported in this paper are still preliminary. It is still essentially a threading method, and depending on templates. The NMR data might have helped, however it has not fully utilized as we would have liked to use them—that is, using them to adjust structures.

I plan to carefully investigate how alignment accuracy depends on D_u, D_c and the discretization step size, and how the empirical computational time depends on D_u and D_c. Another problem worth studying is how to search through the transformation space so that we can use some branch-and-bound technique to speed up our algorithm.

Table 6.3: Structure alignments of some proteins from two different folds Flavodoxin-like and Cupredoxins.

protein (A)	protein (B)	# contacts (A)	# contacts (B)	time (s)	# aligned contacts
1booa	1bawa	441	387	3588	109
1booa	1byoa	441	355	3609	92
1booa	1dpsb	441	586	3384	125
1nat	1amk	435	933	2963	166
1nat	1dpsb	435	586	3163	136
1qmpc	2pcy	452	357	3778	107
1qmpa	8tima	463	930	3516	169
4tmya	1bawa	446	387	3306	110
4tmya	1amk	446	933	2952	154
4tmya	1dpsc	446	587	3242	121
1bawa	1aw2b	387	966	2954	103
1bawa	1b9ba	387	953	2745	120
1bawa	1dpsb	387	586	2543	114

Chapter 7

Combining Spare NMR Data to Improve Protein Structure Prediction

In this chapter I presents a computational framework for detecting similar structures of a target protein with unknown structure using sparse NMR data including chemical shifts, nuclear Overhauser effects (NOE) distance restraints and residual dipolar coupling (RDC). Based on the algorithm proposed in this chapter, I have developed a computer program RAPTOR-NMR for protein structure prediction, through searching for a structural homolog or analog of the target protein in the Protein Data Bank. RAPTOR-NMR can simultaneously take several types of NMR data as input and predict the structure of a target protein. The performance of RAPTOR-NMR is independent of sequence similarity. Therefore, RAPTOR-NMR can go beyond the limitation of current many protein threading or homology modeling based protein structure prediction programs. Experimental results demonstrate that RAPTOR-NMR can correctly identify structural folds for many target proteins with only a limited number of NMR data.

7.1 Introduction

X-ray crystallography and NMR are two major experimental methods for the determination of a protein structure. Compared to the x-ray crystallography technique which requires a high-quality crystal of a protein, the NMR technique can solve the structure of a protein in solution. The available NMR data includes nuclear Overhauser effects (NOE) distance restraints, chemical shifts and residual dipolar coupling (RDC). Chemical shifts identify atoms on the spectrum while an NOE restraint relates two atoms together. The secondary structure type of a protein can be predicted from chemical shifts. For example, several programs such as PsiCSI [51] and TALOS [31] have been developed for secondary structure prediction based on available chemical shifts of a protein.

An NOE distance restraint can specify the spatial distance between two hydrogen atoms, which are usually no more than $5\dot{A}$ away from each other. The residues that the related two hydrogen atoms belong to are not necessarily sequentially adjacent along the primary sequence. The secondary structure and tertiary structure of a protein can be inferred if enough number of high-quality NOE restraints are available from the NMR experiment.

RDC in weak alignment media represents a new NMR technique and is gaining great popularity because it can overcome some limitations of NOE-based NMR structure determination methods. RDC provides information about the angles of atomic bonds, e.g., N-H bonds, of a residue with respect to a particular three-dimensional alignment frame (x,y,z). Let θ denote the angle between the bond and the z-axis of the principal alignment frame and ϕ the angle between the bond's projection in the x-y plane and the x-axis, respectively. The RDC value of this bond can be calculated as follows.

$$D = D_a(3\cos^2\theta - 1) + 1.5D_r(\sin^2\theta\cos 2\phi) \tag{7.1}$$

where D_a and D_r represent the axial and rhombic component of the alignment tensor. Intuitively, D_a and D_r are the intensity of alignment. They can be estimated using a histogram method as follows [27].

$$D_a = D_{zz}/2 \tag{7.2}$$

$$D_r = -\frac{2}{3}D_{yy} - \frac{1}{3}D_{zz} \tag{7.3}$$

where D_{zz} and D_{yy} are the maximum or the minimum values of the experimental RDC values, respectively, with $|D_{zz}| > |D_{yy}|$.

If enough data is available from NMR experiments, then theoretically, the structure of a protein can be solved using energy minimization and molecular dynamics simulation. However, the NMR techniques cannot guarantee to generate enough high-quality data for all the proteins, especially a large protein. It is also time-consuming to generate all the necessary data for the accurate determination of a protein structure. Instead it is relatively easy and less time-consuming to obtain some sparse NMR data for a large protein. Another difficulty in applying RDC data to protein structure determination is that the mapping between an N-H bond and an RDC value is not one-to-one. According to Eq. 7.1, an RDC value does not uniquely define a single physical orientation of an N-H bond. In fact, a single RDC value only restricts the orientation of the bond to two symmetric cones. If not enough RDC values are available, then this kind of degeneracy will render a huge number of possible conformations.

A reliable protein structure prediction tool based on an incomplete set of NMR data fully exploit the available experimental data and extend the capability of current protein structure determination techniques. The NMR data based protein structure prediction procedure is also useful to some NMR peak assignment algorithms [121] that usually use this procedure as a subroutine. There are a number of studies on how to do protein structure prediction using sparse NMR data, including (i) combination of *ab initio* protein structure prediction methods with sparse NMR data [47, 86, 18, 107, 137, 92, 12, 115]; and (ii) combination of comparative modeling methods (i.e., protein threading or homology modeling) with sparse NMR data [98, 132, 87, 38]. The comparative modeling methods search for a similar structure from the Protein Data Bank for a target protein with only a limited number of NMR data. A measure is used to calculate the compatibility between the existing protein structure and the NMR data of the target protein. According to an observation that in nature there are a limited number of unique protein folds, there is a good chance that a new protein has a similar structure in the Protein Data Bank. The comparative modeling methods plays an important role in protein structure prediction, partially because that the *Structural Genomics Initiative* has been producing more and more unique protein folds. Although there are a few papers studying how to combine comparative modeling methods with sparse NMR data, they either study how to combine RDC data with comparative modeling methods [98, 87] or how to combine NOE data with comparative modeling methods [132, 6, 4]. To the best of our knowledge, few-if any papers study how to combine comparative modelling methods with both RDC data and NOE distance restraints simultaneously.

The main focus of this chapter is to develop a computational framework for protein structure prediction with only sparse NMR data including NOE restraints, chemical shifts and RDC data. In contrast to many other methods that predict protein structures based on either NOE restraints or RDC data, our method can do structure prediction using both NOE restraints and RDC data simultaneously, which enables us to fully exploit available NMR data such that a higher prediction accuracy can be achieved. To deal with RDC data, I developed a knowledge-based scoring function, which, to the best of our knowledge, is the first one for RDC data. I formulate this problem as a combinatorial problem and solve it using a linear programming (LP) approach plus discretization of the rotation space. The linear programming approach can treat the NOE data in a strict way and allows us to utilize the existing powerful LP solvers to rapidly reach the optimal solution. In this chapter, I assume that all the NMR data are already assigned to the target protein. NMR assignment is another important and challenging problem, which has been studied by many groups [28, 133, 11, 65, 72].

This chapter is organized as follows. In Section 7.2, I model both the protein structures and the target protein with NMR data using graphs and formulates the problem as a combinatorial optimization problem. Section 7.3 describes a knowledge-based scoring function for RDC data. In Section 7.4, I present two algorithms to process RDC and NOE data, respectively. For the RDC data, a rotation space discretization algorithm is presented, while for the NOE data, a linear programming approach is described. Using the linear program algorithm as an inner-loop subroutine of the discretization algorithm, I can have an algorithm to deal with both RDC and NOE data simultaneously. Section 7.5 presents some implementation details of our program RAPTOR-NMR. In Section 7.6, I present some experimental results including alignment accuracy and fold recognition rate. Finally, Section 7.7 draws a conclusion and discusses future extensions of the algorithm.

7.2 Problem Formulation

Representation of Protein Structures. As described in chapter 3, I use a contact map graph $G = (V, E)$ to model a protein structure in $\Re^3$. Each residue is represented by a vertex in V, associated with its secondary structure type, its N-H bond and the coordinate of its residue center. For each residue, I use its C_α atom as the residue center. There is a contact edge $(i, j) \in E$ between two residues i and j if and only if their spatial distance is within a given distance cutoff D_u. Since the distance between two hydrogen atoms related by a long-range NOE restraint usually is no more than $5\dot{A}$ away from each other, D_u ranges from $7\dot{A}$ to $8\dot{A}$. One edge is also added to connect any two sequentially adjacent residues. Given a protein chain A, let $G[A]$ denote its contact map graph. For a substructure P of A, let $G[P]$ denote the contact map subgraph induced by substructure P.

In a typical protein, two residues cannot be arbitrarily close, which is one of the underlying reasons why lattice models can be used to approximate protein folding. According to simple statistics on the PDB database [14], 99% of inter-residue distances are more than $3.5\dot{A}$. Let the constant D_l ($D_l > 0$) denote the minimum inter-residue distance in a protein. Therefore, it can be easily verified that any residue can be adjacent to at most $(1 + \frac{2D_u}{D_l})^3$ residues.

Representation of A Target Protein With Sparse NMR Data. I also use a graph to represent a target protein with sparse NMR restraints. In this graph, each vertex corresponds to a residue in the target protein. Each edge corresponds to one long-range NOE restraint between two residues. One edge is also added to connect any two sequentially adjacent residues. Each residue is labeled with its predicted secondary structure type and experimental RDC values. The secondary structure of a target protein can be predicted by PsiCSI [51] if its chemical shift data is available or by PSIPRED [59] if only sequence information is available.

NMR-template Alignment. The NMR data including chemical shift, NOE restraints and RDC data measures the geometric features of a protein structure. Two proteins with a similar structure are more likely to produce similar NMR measure on their backbone atoms. Therefore, given a target protein with a limited number of NMR data, to find its similar protein structures in the PDB database, I align the target protein to each of the existing protein structures to see how well the NMR measurement calculated from the existing protein structure is compatible with the sparse NMR constraints. We call this kind of alignment as NMR-template alignment where a template refers to an existing protein structure. Since both the target protein and the template are modeled using graphs, the NMR-template alignment problem can be formulated in a similar way as the contact graph based protein structure alignment problem [129].

Given a structural template A and a target protein B with sparse NMR data, an alignment between A and B is a pair of subchains P and Q satisfying the following conditions:

- P is a substructure of A and Q is a subchain of B;
- There is a one-to-one mapping between the residues in P and Q. One residue p in A is equivalent to residue q in B if and only if p is mapped to q; One contact edge in A is equivalent to one NOE restraint in B if and only if their two end points are equivalent.
- The alignment is sequential. That is, if two residues p_1 and p_2 of A are aligned to q_1 and q_2 of B and p_1 is before p_2 along the primary sequence A, then q_1 must be before q_2 along the primary sequence of B.

The optimal NMR-template alignment is one that minimizes the following scoring function. The scoring function consists of NOE compatibility score E_{noe}, secondary structure compatibility score E_{ss}, residual dipolar coupling score E_{rdc} and gap penalty E_g. The overall scoring function E has the following form.

$$E = W_{noe}E_{noe} + W_{ss}E_{ss} + W_{rdc}E_{rdc} + W_gE_g \tag{7.4}$$

where W_{noe}, W_{ss}, W_{rdc} and W_g are weight factors to be determined later. The gap penalty can be calculated using a linear function $E_g = ag + b$ where g is the number of gaps and a and b are gap extension penalty and gap open penalty, respectively. Here we only use part of RAPTOR scores because the goal in this chapter is to test whether the NOE and RDC data will improve protein structure prediction.

Let $E[A]$ denote the set of contacts in proteins A and $E[B]$ the set of NOE restraints in B. For any residue u in A, let $M(u)$ denote its equivalent residue in B. If there is no equivalent residue for u, then $M(u) = \Lambda$. Then the NOE compatibility score can be calculated as follows:

$$E_{noe} = -\sum_{u,v\in V[A],u<v} f(u,v,M(u),M(v)) \tag{7.5}$$

where

$$f(u,v,M(u),M(v)) = \begin{bmatrix} -\infty & M(u) \geq M(v) \neq \Lambda \\ 1 & (u,v) \in E[A], (M(u),M(v)) \in E[B] \\ 0 & otherwise \end{bmatrix} \tag{7.6}$$

Note that $f(u,v,M(u),M(v)) = -\infty$ is used to avoid two different residues u and v being aligned to the same residue in B or the alignment violating the sequential constraint.

The RDC compatibility score can be calculated as follows.

$$E_{rdc} = \sum_{u\in V[A],M(u)\neq\Lambda} E_{rdc}(u,M(u)) \tag{7.7}$$

where $E_{rdc}(u,M(u))$ is the RDC score for position u being aligned to position $M(u)$. The RDC score will be carefully designed in Section 7.3.

The secondary structure compatibility score can be calculated as follows.

$$E_{ss} = \sum_{u\in V[A],M(u)\neq\Lambda} (Prob(loop,M(u)) - Prob(SS(u),M(u))) \tag{7.8}$$

where $SS(u)$ is the secondary structure type at position u, and $Prob(s,u)$ is the probability of position u with secondary structure type s. The secondary structure type at one position can be *helix*, *beta-strand* or *loop*.

7.3 A Knowledge-Based Scoring Function For RDC Data

To develop a knowledge-based scoring function for RDC data, I randomly chose approximately 1,000 protein pairs from the Protein Data Bank. Two proteins in a pair can be similar in a family level, in a superfamily level or in a fold level. I did structure alignment on each pair using a structure alignment program SARF [7] and superimposed each pair using the transformation generated by SARF. Then for each pair, I calculated the difference of the RDC values between two equivalent residues by setting $D_a = 1$ and $\frac{D_r}{D_a} = 0.1, 0.2, ..., 0.9$. Finally, we can have nine different statistical distributions on the difference of the RDC values between two equivalent residues. From these statistical distributions, I can estimate the probability of a given RDC difference value $P(X_{rdc} > RDC_{diff})$ where X_{rdc} is a random variable representing RDC difference and RDC_{diff} is a given RDC difference value. Figure 7.1 shows one statistical distribution of the RDC difference.

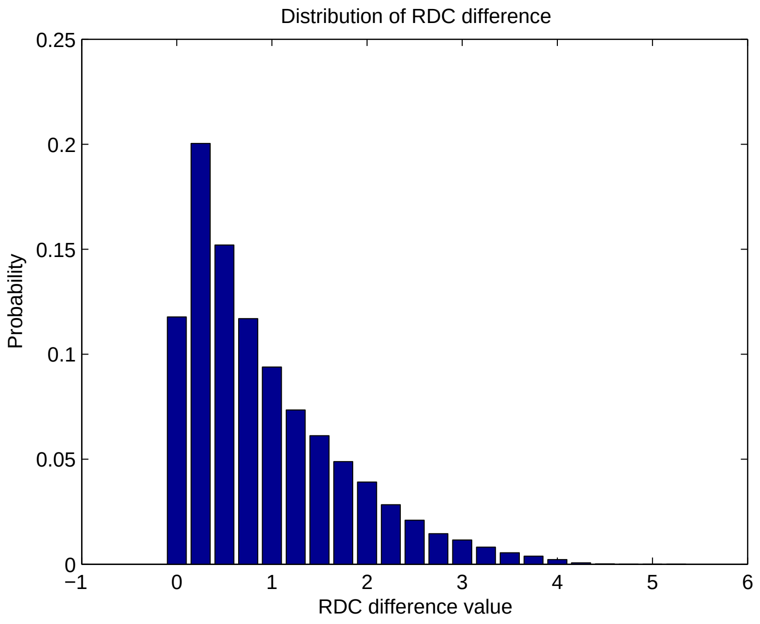

Figure 7.1: An example of RDC difference distribution.

I also examined the statistical distribution of the RDC difference with respect to the similarity relationship between two proteins. In the same superfamily or family level, the mean RDC difference ranges from 0.52 to 0.62 depending on $\frac{D_r}{D_a}$. In the same fold level, the mean RDC difference ranges from 0.72 to 0.82, depending on $\frac{D_r}{D_a}$. Please refer to our supplemental data at http://ttic.uchicago.edu/~jinbo/RAPTOR-NMR/ for more detailed results. The statistical data indicates that RDC data will be more helpful for the alignment between two proteins in the same superfamily or family.

Based on these empirical results, I designed the following three scoring functions for RDC data.

$$E_{rdc}(u, M(u)) = \frac{|RDC(u) - RDC(M(u))|}{\max\{|RDC(u)|, |RDC(M(u))|\}} - m(\frac{D_r}{D_a}) \tag{7.9}$$

where $m(\frac{D_r}{D_a})$ is the mean of $\frac{|RDC(u)-RDC(M(u))|}{\max\{|RDC(u)|,|RDC(M(u))|\}}$ and ranges from 0.62 to 0.8, depending on $\frac{D_r}{D_a}$.

$$E_{rdc}(u, M(u)) = -\log(P(X_{rdc} > |RDC(u) - RDC(M(u))|)) \tag{7.10}$$

$$E_{rdc}(u, M(u)) = -P(X_{rdc} > |RDC(u) - RDC(M(u))|) \tag{7.11}$$

Experimental results show that Eq. 7.11 generates the best alignment accuracy [1]. Therefore, I incorporate Eq. 7.11 into the scoring function Eq. 7.7 to deal with RDC data. Please notice that in both Eq. 7.10 and Eq. 7.11, the probability depends on $\frac{D_r}{D_a}$.

7.4 Algorithms For NMR-template Alignment

A Discretization Algorithm For RDC Data One of the key issues with the RDC data is that we do not know the principal alignment frame which is used to generate the RDC data. We cannot easily infer the principal alignment frame from the RDC data unless we have enough RDC data. Without the alignment frame, we have no way to calculate the RDC values of the N-H bonds of the protein structure. We need an algorithm to search for the principal alignment frame so that the best match between the RDC data of the target protein and the N-H bonds of the template can be achieved. This problem is equivalent to fixing the principal alignment frame to an arbitrary coordinate system and rotating all the N-H bonds of protein A such that the objective function in Eq. 7.4 is minimized.

A rotation can be represented by a triple (α, β, γ) where α, β and γ are Euler angles. Assume $\hat{v}$ to be the resultant vector for rotating a vector v by (α, β, γ). Then $\hat{v}$ can be

[1] In this chaper, the alignment accuracy is defined as the number of correctly aligned positions. One target position is correctly aligned if its alignment is no more than 4 positions away from the correct one, which is generated by a structure alignment program SARF.

calculated using the following formula:

$$\hat{v} = R_z(\gamma)R_y(\beta)R_x(\alpha)v \tag{7.12}$$

where

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{bmatrix} \tag{7.13}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{7.14}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.15}$$

A small change $\delta\alpha$ of the rotation angle α will change the resultant vector $\hat{v}$ by at most $O(|v||\delta\alpha|)$. Since the bond length can be treated as a constant, a small change $\delta\alpha$ of the rotation angle α will change the (θ, ϕ) angles of the N-H bonds by $O(\delta\alpha)$. Similarly, a small change of β and γ will not change the bond angles dramatically. According to Eq. 7.1, a small change of bond angles θ and ϕ will change the RDC value in the same order of magnitude. This means that a small change of the principal alignment frame will change the RDC value in the same order of magnitude. Therefore, we can discretize the whole rotation space into a polynomial number of possible rotations. By working on these possible discrete rotations, we can find an NMR-template alignment with an alignment score very close to the optimal. In fact, we can find all the possible rotations that lead to a near-optimal NMR-template alignment.

If only RDC data is used with the target protein, then for each discrete rotation, we can use a dynamic programming algorithm to find the optimal NMR-template alignment. This algorithm has been described in Qu *et. al.*'s paper [98]. In fact, combining the discretization technique and dynamic programming algorithm, we can prove that there is a polynomial-time approximation algorithm for this case, using a similar proof technique described in Kolodny and Linial's paper [67].

Simplified Representation of Structural Templates NOE data can define the distance between two residues as long as there are pairwise items and variables gaps in the alignment the threading problem is NP hard and we can use linear programming to solve this problem. To decrease computational time, I further simplify the representation of a protein structural template. I assume that the primary sequence of the template is parsed as a linear series of cores with the connecting loops between the adjacent cores. Each core is a conserved segment of an α-helix or β-sheet secondary structure. Although the secondary structure is often conserved, insertion or deletion may occur at the two ends of a secondary structure. So I only keep the most conserved part. In doing NMR-template alignment, I assume that no gap occurs within a core and only consider the contacts between two core residues. The reason is that the contacts between two core residues are more conserved than other contacts. Let c_i $(i = 1, 2, \ldots, M)$ denote all the cores of one structural template, where M is the number of the cores. Two cores c_i and c_{i+1} are sequentially adjacent along the primary sequence of the structural template. The segment between c_i and c_{i+1} is a loop, for each i. Let $head_i$ denote the position of the first residue in core c_i and len_i denote the length of core c_i.

The original contact graph representation of the structural template can be simplified accordingly. All the vertices in the original graph are merged into a single vertex and all the edges between the residues of two cores are merged into a single edge. I use $\mathcal{E}_{core}[A]$ to denote the set of inter-core edges of protein A. I assume that there is no gap occurring within a core since cores are the most conserved segments in a protein. Therefore, the search space of feasible NMR-template alignments is greatly reduced.

Linear Programming Formulation For simplicity, when we say core c_i is aligned to target protein position s_j, we always mean that the template segment that this core represents is aligned to target protein segment (s_j, s_{j+len_i-1}). Let $D[i]$ denote all valid target protein positions that c_i could be aligned to. For any i, j and $l \in D[i]$, let $R[i, j, l]$ denote all the valid target protein positions of c_j given c_i is aligned to target protein position s_l. To keep the sequential order in the alignment, if $i < j$, then, for any $k \in R[i, j, l]$, $k - l > len_i$ holds. See Figure 7.2 for an example of $D[i]$ and $R[i, j, l]$.

An alignment is valid if and only if the following three conditions are satisfied: (1) each

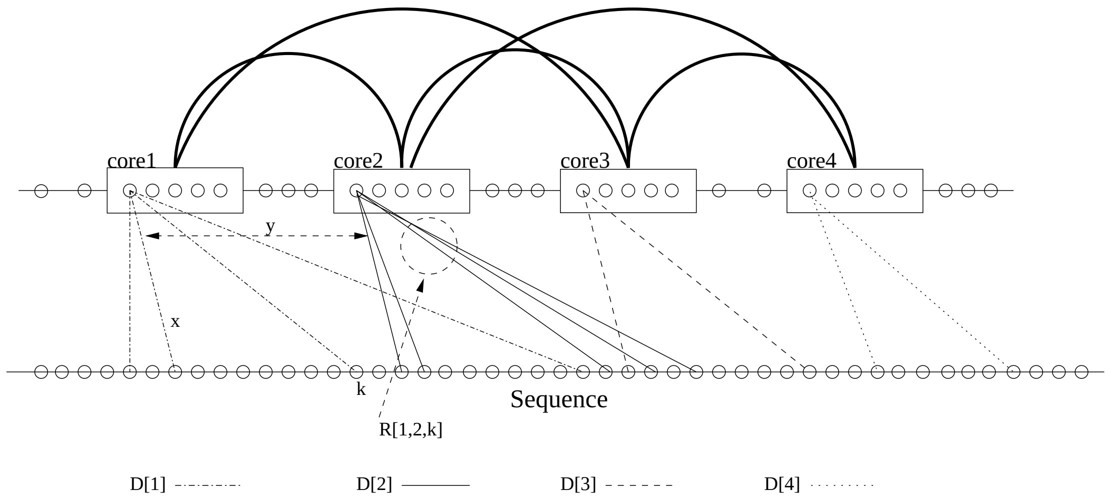

Figure 7.2: Example of $D[i]$ and $R[i, j, l]$.

core of protein A is aligned to one position of protein B [2]; (2) the alignment orders of any two different cores c_{i_1} and c_{i_2} are kept; and (3) if c_{i_j} is aligned to target protein positions s_{l_j}, $j = 1, 2$, then $s_{l_1} \in R[i_2, i_1, s_{l_2}]$ and $s_{l_2} \in R[i_1, i_2, s_{l_1}]$.

In formulating the problem as a linear program, I introduce two different kinds of binary variables x and y. Let $x_{i,l}$ be a binary variable such that $x_{i,l} = 1$ if and only if core c_i is aligned to target protein position s_l. Similarly, for any two cores c_{i_1} and c_{i_2} that are connected by an edge in $\mathcal{E}_{core}$, let $y_{(i_1,l_1),(i_2,l_2)}$ indicate that core c_{i_1} is aligned to target protein position s_{l_1} and simultaneously core c_{i_2} is aligned to target protein position s_{l_2}. The variable $y_{(i_1,l_1),(i_2,l_2)}$ is equal to 1 if and only if both x_{i_1,l_1} and x_{i_2,l_2} are equal to 1. We say that $y_{(i_1,l_1),(i_2,l_2)}$ is generated by x_{i_1,l_1} and x_{i_2,l_2}. The x variables are called the alignment variables and y variables are called the contact variables.

Now the NMR-template alignment problem can be formulated as the following integer program.

$$\min E = W_{noe}E_{noe} + W_{ss}E_{ss} + W_{rdc}E_{rdc} + W_gE_g \tag{7.16}$$

[2]We add some artificial amino acids at the two ends of protein B to guarantee that any core of A can be aligned to one position in B.

$$E_{noe} = \sum_{(c_i,c_j)\in\mathcal{E}_{core}[A]} \sum_{l\in D[i]} \sum_{k\in R[i,j,l]} y_{(i,l),(j,k)} NOE(i,j,l,k), \tag{7.17}$$

$$NOE(i,j,l,k) = \sum_{u=0}^{len_i-1} \sum_{v=0}^{len_j-1} f(head_i+u, head_j+v, l+u, k+v) \tag{7.18}$$

$$E_{rdc} = \sum_{i=1}^{M} \sum_{l\in D[i]} [x_{i,l} \times \sum_{u=0}^{len_i-1} E_{rdc}(head_i+u, l+u)] \tag{7.19}$$

$$E_{ss} = \sum_{i=1}^{M} \sum_{l\in D[i]} [x_{i,l} \times \sum_{u=0}^{len_i-1} (Prob(loop, l+u) - Prob(SS(head_i+u), l+u))] \tag{7.20}$$

$$E_g = \sum_{i=1}^{M} \sum_{l\in D[i]} \sum_{k\in R[i,i+1,l]} y_{(i,l),(i+1,k)} G(i,l,k) \tag{7.21}$$

where Eq.7.17 and Eq.7.18 calculate the NOE compatibility score, Eq. 7.19 calculates the RDC compatibility score, Eq. 7.20 calculates the secondary structure compatibility score, and Eq. 7.21 calculates the gap penalty in the alignment. $NOE(i,j,l,k)$ is the NOE score corresponding to the contacts between c_i and c_j when they are aligned to target protein positions s_l and s_k, respectively. $G(i,l,k)$ is the gap penalty score when the loop region between c_i and c_{i+1} is aligned to the target protein segment s_l,s_{l+1},...,s_k. Given i,l,k, $G(i,l,k)$ can be computed by a dynamic programming algorithm in advance.

The constraint set is as follows:

$$\sum_{j\in D[i]} x_{i,j} = 1, \quad i = 1,2,\ldots,M; \tag{7.22}$$

$$\sum_{k\in R[i,j,l]} y_{(i,l)(j,k)} = x_{i,l}, \quad (c_i,c_j)\in\mathcal{E}_{core}[A]; \tag{7.23}$$

$$\sum_{l\in R[j,i,k]} y_{(i,l)(j,k)} = x_{j,k}, \quad (c_i,c_j)\in\mathcal{E}_{core}[A]; \tag{7.24}$$

$$x_{i,j} \in \{0,1\}, \quad j\in D[i],\ i = 1,2,\ldots,M; \tag{7.25}$$

$$y_{(i,l)(j,k)} \in \{0,1\}, \quad \forall l \in D[i], k \in D[j],\ i,j = 1,2,\ldots,M. \tag{7.26}$$

Constraint 7.22 says that one core can be aligned to a unique target protein position. Constraints 7.25 and 7.26 guarantee x and y variables to be either 0 or 1. Constraints 7.23 and 7.24 imply that one y variable is equal to 1 if and only if its two x variables are equal to 1. The linear program formulation used here is very similar to that used in the protein structure prediction server RAPTOR [131, 130]. By minimizing the objective function (7.16) under constraints (7.22)-(7.26), we can obtain an alignment with minimized alignment score.

7.5 Implementation

To achieve a tradeoff between computational time and alignment accuracy, the rotation space is discretized in two stages. In the first stage, the step size for rotation space discretization is 30 degrees. After I find the best rotation in this stage, I further discretize the rotation space in the region around the best rotation using step size 15 degrees.

If both RDC and NOE data are available, it takes a long time to find the best alignment between two large proteins. To save computational time, I choose only 10 or 20 rotations that can lead to a good match between the set of template RDC values and the set of experimental RDC values. I calculate the similarity between two sets of RDC values using their histograms. The histogram similarity is measured as the area of the intersection of two histograms normalized by the target protein size. When the template size is much larger than the target protein size, the histogram similarity is biased toward bigger. To offset this bias, I cut the template into many segments every 10 residues along its primary sequence. Each segment has the same length as the target protein. Then I calculate one histogram for each segment. For a given rotation, the histogram similarity between the target protein and the template is defined as the best histogram similarity between the target protein and all the template segments. Experimental results demonstrate that top 10 rotations chosen in this way can produce a good NMR-template alignment.

The weight factors W_{ss},W_{noe}, W_{rdc} and W_g are chosen by optimizing the overall alignment accuracy on the training set, which is arbitrarily chosen from the Protein Data Bank. In fact, I did not conduct a strict optimization procedure to determine the weight factors.

Instead, I randomly generated 300 combinations of weight factors with uniform distribution in range 0 to 1 and tested their performance on the training set and picked up the best combination. The alignment accuracy is not very sensitive to the above four weight factors since there are also many other weight factors used in RAPTOR. Many sets of weight factors that can produce a good alignment accuracy.

7.6 Experimental Results

Alignment accuracy I random chose 40 protein pairs with a similar fold to test the alignment accuracy of the NMR data. I tested the performance of RAPTOR-NMR using only RDC data, only NOE data, and both RDC and NOE data, respectively. For a given target protein, all the available RDC data is used and some long-range NOE restraints are randomly chosen. The number of long-range NOE restraints is no more than the target protein size. The secondary structure of a target protein is predicted using PSIPRED [59]. Due to space limit, I only list the alignment accuracy of some test pairs in Table 7.1. As shown in this table, if two proteins are in the same family, then any method works very well. However, if two proteins are similar in only a fold level, then RAPTOR-NMR performs better than RAPTOR without experimental data. The NOE data can produce a better alignment accuracy than the RDC data. RAPTOR-NMR's performance is more robust if both RDC and NOE data are used, compared to only RDC data or only NOE data is used.

Fold recognition. I tested 18 target proteins with both RDC and NOE data against a template database to measure the fold recognition rate of different NMR data. The template database consists of approximately 2,900 domains taken from the most recent SCOP database (version 1.69). In the template database, any two domains share no more than 40% sequence identity and there is only one representative for each SCOP family. The templates are ranked by the alignment score and the best template is chosen as the predicted fold of the target protein. In fact, since the target proteins have a similar structure in the template database, any kind of NMR data performs similarly. Due to space limit, I only list the prediction result of RAPTOR-NMR with only RDC data in

Table 7.2. Please refer to http://ttic.uchicago.edu/˜jinbo/RAPTOR-NMR/ for all the detailed results.

As shown in Table 7.2, RAPTOR-NMR performs very well with only RDC data for all the target proteins but 1f3ya, 1nyaa and 1b4ca. RAPTOR-NMR can have a very good prediction for 1f3ya and a reasonably good prediction for 1nyaa if both RDC and NOE data are used. If I enlarge the template database to 7,200 domains by removing the restriction that only one representative is chosen for each SCOP family, then RAPTOR-NMR with only RDC data can also give a very good prediction for 1b4ca, 1f3ya and 1nyaa.

Computational efficiency. If only RDC data is used, RAPTOR-NMR runs very fast. It takes an average of one second to conduct one NMR-template alignment on a Linux PC. If both RDC and NOE data are used, it takes approximately 10 seconds to conduct one NMR-template alignment. An optimization of our C++ source codes will further improve the computational efficiency. Therefore, with a small Linux cluster, I can do structure prediction for a target protein within several hours.

7.7 Conclusions

This chapter presented a computational framework for detection of similar structures of a target protein with sparse NMR data. Experimental results demonstrate that the model and algorithms proposed in this chapter can be used for protein structure prediction even if a limited number of NMR data is available.

The prediction accuracy of RAPTOR-NMR is limited by several factors. First, RAPTOR-NMR currently uses the alignment score (i.e., objective function value) to rank all the templates for a given target protein, to avoid the time-consuming calculation of Z-score. The alignment score is usually biased by the length difference of the target and the template. A better method is to develop a machine learning algorithm to rank all the templates based on their alignments to the target protein, just like what our RAPTOR server does [131, 130]. Second, in current alignment model, if two template cores are aligned to the target protein, then all the cores between them are also required to be aligned to the target protein. Sometimes this requirement will limit the alignment accuracy. Finally, our alignment model does not allow gaps within a core, which might impact the alignment accuracy slightly. To allow for gaps within a single core, I need to develop a more efficient alignment algorithm.

In the alignment scoring function, I does not take into consideration the sequence similarity. This feature enables us to detect the analog structures of a target protein. An interesting question is what if I use the sequence information in RAPTOR-NMR? In this chapter, I assume that all the NMR data are already assigned to the target protein. NMR peak assignment itself is an important and challenging problem. The next step is to develop an efficient algorithm to conduct NMR assignment and NMR-template alignment simultaneously.

Table 7.1: Alignment accuracy of different experimental data on 22 test protein pairs. The right four columns are the alignment accuracy with different NMR data. The fifth column is the maximum alignment accuracy that can be achieved by any method. (seq ide=sequence identity)

template	target protein	#RDC used	seq ide (%)	SARF accuracy	RAPTOR accuracy	RDC only	NOE only	RDC +NOE
1cb1	1b4ca	53	30	52	38	52	48	52
1k8ua	1b4ca	53	39	68	68	68	68	68
1agre	1cmza	72	45	124	124	124	124	123
1dk8a	1cmza	72	11	118	116	105	115	114
1eova	1d2ba	85	12	66	18	0	46	52
1fr3a	1d2ba	85	6	50	25	26	50	50
3chbd	1d2ba	85	3	65	11	20	53	41
1czpa	1d3za	63	6	60	38	41	41	50
1n62a	1d3za	63	12	56	30	54	41	54
1b9oa	1e8la	108	36	117	117	117	109	117
1qsaa	1e8la	108	14	95	39	52	37	58
1gg3a	1i42a	56	4	47	30	45	44	44
1jroa	1i42a	56	2	46	13	11	26	30
2pia	1i42a	56	12	47	13	37	33	37
1alvb	1kqva	64	9	65	20	25	28	28
1wdcc	1kqva	64	16	60	28	28	50	36
1ra9	1luda	143	30	148	145	147	147	146
1axib	1n6ua	109	10	139	122	114	123	125
1fyhb	1n6va	109	8	124	121	121	119	124
1kshb	1n6va	109	1	48	10	16	43	43
1d5ga	1n7ta	82	32	77	75	75	73	75
2sas	1nyaa	73	14	105	104	89	104	104

Table 7.2: The best templates found for 18 target proteins. Only RDC data is used.

target protein	1b4ca	1c06a	1cmza	1d2ba	1d3za	1e8la	1f3ya	1i42a
target size	92	159	128	126	76	129	165	89
best template	1s7ba	1fjgd	1agre	1oo9b	1ogwa	3lzt	1aqca	1h8ca
alignment accuracy	27	155	124	117	72	129	10	65
target protein	1jwea	1khma	1ksma	1l3ga	1luda	1m12a	1n7ta	1nyaa
target size	114	89	76	123	162	84	103	179
best template	1b79a	1dtja	1qx2a	1bm8	1ra9	1nkl	1kwaa	1jj2u
alignment accuracy	98	65	70	91	147	75	74	17

Chapter 8

LS_Boosting for Protein Fold Recognition

As introduced in chapter 3, protein structure prediction is one of the most important and difficult problems in computational molecular biology. Protein threading represents one of the most promising techniques for this problem. One of the critical steps in protein threading, fold recognition, is to choose the best-fit template for the query protein with the structure to be predicted. The standard method for template selection is to rank candidates according to the z-score of the sequence-template alignment. However, the z-score calculation is time-consuming, which greatly hinders structure prediction at a genome scale.

In this chapter, we present a machine learning approach that treats the fold recognition problem as a regression task and uses a least-squares boosting algorithm (LS_Boost) to solve it efficiently. We test our method on Lindahl's benchmark and compare it with other methods. According to our experimental results we can draw the conclusions that: (1) Machine learning techniques offer an effective way to solve the fold recognition problem. (2) Formulating protein fold recognition as a regression rather than a classification problem leads to a more effective outcome. (3) Importantly, the LS_Boost algorithm does not require the calculation of the z-score as an input, and therefore can obtain significant computational savings over standard approaches. (4) The LS_Boost algorithm obtains superior accuracy, with less computation for both training and testing, than alternative machine learning approaches such as SVMs and neural networks, which also need not calculate the z-score.

Finally, by using the LS_Boost algorithm, one can identify important features in the fold recognition protocol, something that cannot be done using a straightforward SVM approach.

8.1 Introduction

The traditional fold recognition technique is based on calculating the z-score, which statistically tests the possibility of the target sequence folding into a structure very similar to the template [21]. In this technique, the z-score is calculated for each sequence-template alignment by first determining the distribution of alignment scores among random re-shufflings of the sequence, and then comparing the alignment score of the correct sequence (in standard deviation units) to the average alignment score over random sequences. Note that the z-score calculation requires the alignment score distribution to be determined by randomly shuffling the sequence many times (approx. 100 times), meaning that the shuffled sequence has to be threaded to the template repeatedly. Thus, the entire process of calculating the z-score is very time-consuming. In this chapter, instead of using the traditional z-score technique, we propose to solve the fold recognition problem by treating it as a machine learning problem.

Several research groups have already proposed machine learning methods, such as neural networks [57, 134] and support vector machines (SVMs) [125, 131] for fold recognition. In this general framework, for each sequence-template alignment, one generates a set of features to describe the instance, treats the extracted features as input data, and the alignment accuracy or similarity level as a response variable. Thus, the fold recognition problem can be expressed as a standard prediction problem that can be solved by supervised machine learning techniques for regression or classification. In this chapter we investigate a new approach that proves to be simpler to implement, more accurate and more computationally efficient. In particular, we combine the gradient boosting algorithm of Friedman [41] with a least-squares loss criterion to obtain a least-squares boosting algorithm, LS_Boost. We use LS_Boost to estimate the alignment accuracy of each sequence-template alignment and employ this as part of our fold recognition technique.

To evaluate our approach, we experimentally test it on Lindahl's benchmark [78] and compare the resulting performance with other fold recognition methods, such as the z-score method, SVM regression, SVM classification, neural networks and Bayes classification. Our experimental results demonstrate that the LS_Boost method outperforms the other techniques in terms of both prediction accuracy and computational efficiency. It is also a much easier algorithm to implement.

The remainder of the chapter is organized as follows. We first show how to generate features from each sequence-template alignment and convert protein threading into a standard prediction problem (making it amenable to supervised machine learning techniques). We discuss how to design the least-squares boosting algorithm by combining gradient boosting with a least-squares loss criterion, and then describe how to use our algorithm to solve the fold recognition problem. Finally, we will describe our experimental set-up and compare LS_Boost with other methods, leading to the conclusions we present in the end.

8.2 Protein Threading and Fold Recognition

8.2.1 The z-score Method for Fold Recognition

The z-score is defined to be the "distance" (in standard deviation units) between the optimal alignment score and the mean alignment score obtained by randomly shuffling the target sequence. An accurate z-score can cancel out the sequence composition bias and offset the mismatch between the sequence size and the template length. Bryant et al. [21] proposed the following procedures to calculate z-score:

1. Shuffle the aligned sequence residues randomly.
2. Find the optimal alignment between the shuffled sequence and the template.
3. Repeat the above two steps N times, where N is on the order of one hundred. Then calculate the distribution of these N alignment scores.

After the N alignment scores are obtained, we calculate the deviation of the optimal alignment score from the distribution of these N alignment scores.

We can see from above that in order to calculate the z-score for each sequence-template alignment, we need to shuffle and rethread the target sequence many times, which takes a significant amount of time and essentially prevents this technique from being applied to genome-scale structure prediction.

8.2.2 Machine Learning Methods for Fold Recognition

Another approach to the fold recognition problem is to use machine learning methods, such as neural networks, as in the GenTHREADER [57] and PROSPECT-I systems [134], or SVMs, as in the RAPTOR system [131]. Current machine learning methods generally treat the fold recognition problem as a classification problem. However, there is a limitation to the classification approach that arises when one realizes that there are three levels of similarity that one can draw between two proteins: fold level similarity, superfamily level similarity and family level similarity. Currently, classification-based methods treat the three different similarity levels as a single level, and thus are unable to effectively differentiate one similarity level from another while maintaining a hierarchical relationship between the three levels. Even a multi-class classifier cannot deal with this limitation very well since the three levels are in a hierarchical relationship.

Instead, we use a regression approach, which simply uses the alignment accuracy as the response value. That is, we reformulate the fold recognition problem as predicting the alignment accuracy of a threading pair, which then is used to differentiate the similarity level between proteins. In our approach, we use SARF [7] to generate the alignment accuracy between the target protein and the template protein. The alignment accuracy of threading pair is defined to be the number of correctly aligned positions, based on the correct alignment generated by SARF. A position is correctly aligned only if its alignment position is no more than four position shifts away from its correct alignment. On average, the higher the similarity level between two proteins, the higher the value of the alignment accuracy will be. Thus alignment accuracy can help to effectively differentiate the three similarity levels. Below we will show in our experiments that the regression approach obtains much better results than the standard classification approach.

8.3 Feature Extraction

One of the key steps in the machine learning approach is to choose a set of proper features to be used as inputs for predicting the similarity between two proteins. After optimally threading a given sequence to each template in the database, we generate the following features from each threading pair.

1. Sequence size, which is the number of residues in the sequence.
2. Template size, which is the number of residues in the template.
3. Alignment length, which is the number of aligned residues. Usually, two proteins from the same fold class should share a large portion of similar sub-structure. If the alignment length is considerably smaller than the sequence size or the template size, then it indicates that this threading pair is unlikely to be in the same SCOP class.
4. Sequence identity. Although a low sequence identity does not imply that two proteins are not similar, a high sequence identity can indicate that two proteins should be considered as similar.
5. Number of contacts with both ends being aligned to the sequence. There is a contact between two residues if their spatial distance is within a given cutoff. Usually, a longer protein should have more contacts.
6. Number of contacts with only one end being aligned to the sequence. If this number is big, then it might indicate that the sequence is aligned to an incomplete domain of the template, which is not good since the sequence should fold into a complete structure.
7. Total alignment score.
8. Mutation score, which measures the sequence similarity between the target protein and the template protein.
9. Environment fitness score. This feature measures how well to put a residue into a specific environment.

10. Alignment gap penalty. When aligning a sequence and a template, some gaps are allowed. However, if there are too many gaps, it might indicate that the quality of the alignment is bad, and therefore the two sequences may not be in the same similarity level.

11. Secondary structure compatibility score, which measures the secondary structure difference between the template and the sequence in all positions.

12. Pairwise potential score, which characterizes the capability of a residue to make a contact with another residue.

13. The z-score of the total alignment score and the z-score of a single score item such as mutation score, environment fitness score, secondary structure score and pairwise potential score.

Notice that here we still take into consideration the traditional z-score for the sake of performance comparison. But later we will show that we can obtain nearly the same performance without using the z-score, which means it is unnecessary to calculate the z-score as one of the features.

We calculate the alignment accuracy between the target protein and the template protein using a structure comparison program SARF. We use the alignment accuracy as the response variable. Given the training set with input feature vectors and the response variable, we need to find a prediction function that maps the features to the response variable. By using this function, we can estimate the alignment accuracy for each sequence-template alignment. Then, all the sequence-template alignments can be ranked based on the predicted alignment accuracy and the first-ranked one is chosen as the best alignment for the sequence. Thus we have converted the protein structure problem to a function estimation problem. In the next section, we will show how to design our LS_Boost algorithm by combining the gradient boosting algorithm of Friedman [41] with a least-squares loss criterion.

8.4 Least-squares Boosting Algorithm for Fold Recognition

The problem can be formulated as follows. Let x denote the feature vector and y the alignment accuracy. Given an input variable x, a response variable y and some samples $\{y_i, x_i\}_{i=1}^{N}$, we want to find a function $F^*(x)$ that can predict y from x such that over the joint distribution of $\{y, x\}$ values, the expected value of a specific loss function $L(y, F(x))$ is minimized [41]. The loss function is used to measure the deviation between the real y value and the predicted y value.

$$\begin{aligned} F^*(x) &= \arg\min_{F(x)} E_{y,x} L(y, F(x)) \\ &= \arg\min_{F(x)} E_x[E_y L(y, F(x))|x] \end{aligned} \tag{8.1}$$

Normally $F(x)$ is a member of a parameterized class of functions $F(x; P)$, where P is a set of parameters. We use the form of the "additive" expansions to design the function as follows:

$$F(x; P) = \sum_{m=0}^{M} \beta_m h(x; \alpha_m) \tag{8.2}$$

where $P = \{\beta_m, \alpha_m\}_{m=0}^{M}$. The functions $h(x; \alpha)$ are usually simple functions of x with parameters $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_M\}$. When we wish to estimate $F(x)$ non-parametrically the task becomes more difficult. In general, we can choose a parameterized model $F(x; P)$ and change the function optimization problem to parameter optimization. That is, we fix the form of the function and optimize the parameters instead. A typical parameter optimization method is a "greedy-stagewise" approach. That is, we optimize $\{\beta_m, \alpha_m\}$ after all of the $\{\beta_i, \alpha_i\}(i = 0, 1, \ldots, m-1)$ are optimized. This process can be represented by the following two recursive equations.

$$(\beta_m, \alpha_m) = \arg\min_{\beta,\alpha} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta h(x_i; \alpha)) \tag{8.3}$$

$$F_m = F_{m-1}(x) + \beta_m h(x; \alpha_m) \tag{8.4}$$

Friedman proposed a steepest-descent method to solve the optimization problem described in Equation 8.2 [41]. This algorithm is called the Gradient Boosting algorithm and its entire procedure is given in Figure 8.1.

By employing the least square loss function $(L(y,F)) = (y-F)^2/2$ we have a least-squares boosting algorithm shown in Figure 8.2. For this procedure, ρ is calculated as follows:

$$\begin{aligned}(\rho, \alpha_m) &= \arg\min_{\rho,\alpha} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(x_i;\alpha_m)]^2 \\ \text{and therefore} \qquad \rho &= N \times \tilde{y}_i / \sum_{i=1}^{N} h(x_i;\alpha_m) \qquad (8.5)\end{aligned}$$

The simple function $h(x, \alpha)$ can have any form that can be conveniently optimized over α. In terms of boosting, optimizing over α to fit the training data is called *weak learning.* In this chapter, for considerations of speed, we choose some function for which it is easy to obtain α. The simplest function to use here is the linear regression function: note $\alpha = (a, b)$

$$y = ax + b \qquad (8.6)$$

where x is the input feature and y is the alignment accuracy. The parameters of the linear regression function can be solved easily by the following equation:

$$\begin{aligned}a &= \frac{l_{xy}}{l_{xx}}, b = y - ax \\ \text{where} \qquad l_{xx} &= n \times \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2 \\ l_{xy} &= n \times \sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)\end{aligned}$$

There are many other simple functions one can use, such as an exponential function $y = a + e^b x$, logarithmic function $y = a + blnx$, quadratic function $y = ax^2 + bx + c$, or hyperbolic function $y = a + b/x$, etc.

For each round, we choose one feature and obtain the simple function $h(x, \alpha)$ with the minimum least-squares error. The underlying reasons for choosing a single feature at each round are: i) we would like to see the role of each feature in fold recognition; and ii) we notice that alignment accuracy is proportional to some features. For example, the higher the alignment accuracy, the lower the mutation score, fitness score and pairwise score. Figure 8.3 shows the relation between alignment accuracy and mutation score.

In the end, we combine these simple functions to form the final regression function. As such, Algorithm 8.2 translates to the following procedures.

1. Calculate the difference between the real alignment accuracy and the predicted alignment accuracy. We call this difference the alignment accuracy residual. Assume the initial predicted alignment accuracy is the average alignment accuracy of the training data.

2. Choose a single feature which correlates best with the alignment accuracy residual. The parameter ρ is calculated by using Equation 8.5. Then the alignment accuracy residual is predicted by using this chosen feature and the parameter.

3. Update the predicted alignment accuracy by adding the predicted alignment accuracy residual. Repeat the above two steps until the predicted alignment accuracy does not change significantly.

8.5 Experimental Results

When predicting protein structure , we thread its sequence to each template in the database and obtain the predicted alignment accuracy using the LS_Boost algorithm. We choose the template with the highest alignment accuracy as the basis to build the structure of the target sequence.

We can describe the relationship between two proteins at three different levels: family level, superfamily level and fold level. If two proteins are similar at the family level, then these two proteins have evolved from a common ancestor and usually share more than 30% sequence identity. If two proteins are similar only at the fold level, then their structures are similar even though their sequences are not similar. The superfamily-level similarity is something in between family level and fold level. If the target sequence has a template that is in the same family as the sequence, then it is easier to predict the structure of the sequence. If two proteins are similar only at fold level, it means they share less sequence similarity and it is harder to predict their relationship.

We use the SCOP database [93] to judge the similarity between two proteins and evaluate our predicted results at different levels. If the predicted template is similar to the target sequence at the family level according to the SCOP database, we treat it as correct prediction at the family level. If the predicted template is similar at the superfamily level but not at the family level, then we assess this prediction as being correct at the superfamily level. Similarly, if the predicted template is similar at the fold level but not at the other two levels, we assess the prediction as correct at the fold level. When we say a prediction is correct according to the top K criterion, we mean that there are no more than $K - 1$ incorrect predictions ranked before this prediction. The fold-level relationship is the hardest to predict because two proteins share very little sequence similarity in this case.

To train the parameters in our algorithm, we randomly choose 300 templates from the FSSP list [5] and 200 sequences from Holm's test set [50]. By threading each sequence to all the templates, we obtain a set of 60,000 training examples.

To test the algorithm, we use Lindahl 's benchmark, which contains 976 proteins, each pair of which shares at most 40% sequence identity. By threading each one against all the others, we obtain a set of 976×975 threading pairs. Since the training set is chosen randomly from a set of non-redundant proteins, the overlap between the training set and Lindahl's benchmark is fairly small, which is no more than 0.4 percent of the whole test set. To ensure the complete separation of training and testing sets, these overlap pairs are removed from the test data. We calculate the recognition rate of each method at the three similarity levels.

Sensitivity Figure 8.4 shows the sensitivity of our algorithm at each round. We can see that the LS_Boost algorithm nearly converges within 100 rounds, although we train the algorithm further to obtain higher performance.

Table 8.1 lists the results of our algorithm against several other algorithms. PROSPECT II uses the z-score method, and its results are taken from Kim et al.'s paper [66]. We can see that the LS_Boost algorithm is better than PROSPECT II at all three levels. The

Table 8.1: Sensitivity of the LS_Boost method compared with other structure prediction servers.

	Family Level		Superfamily Level		Fold Level	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
RAPTOR (LS_Boost)	86.5%	89.2%	60.2%	74.4%	38.8%	61.7%
PROSPECT II	84.1 %	88.2%	52.6%	64.8%	27.7%	50.3%
FUGUE	82.3%	85.8%	41.9%	53.2%	12.5%	26.8%
PSI_BLAST	71.2%	72.3%	27.4%	27.9%	4.0%	4.7%
HMMER_PSIBLAST	67.7%	73.5%	20.7%	31.3%	4.4%	14.6%
SAMT98-PSIBLAST	70.1%	75.4%	28.3%	38.9%	3.4%	18.7%
BLASTLINK	74.6%	78.9%	29.3%	40.6%	6.9%	16.5%
SSEARCH	68.6%	75.7%	20.7%	32.5%	5.6%	15.6%
THREADER	49.2%	58.9%	10.8%	24.7%	14.6%	37.7%

results for the other methods are taken from Shi et al's paper [110]. Here we can see that our method apparently outperforms the other methods. However, since we use different sequence-structure alignment methods, this disparity may be partially due to different threading techniques. Nevertheless, we can see that the machine learning approaches normally perform much better than the other methods.

Table 8.2 shows the results of our algorithm against several other popular machine learning methods. Here we will not describe the detail of each method. In this experiment, we use RAPTOR to generate all the sequence-template alignments. For each different method, we tune the parameters on the training set and test the model on the test set. In total we test the following six other machine learning methods.

1. SVM regression. Support vector machines are based on the concept of structural risk minimization from statistical learning theory [117]. The fold recognition problem is treated as a regression problem, therefore we consider SVMs used for regression. Here we use the svm_light software package [55] and an RBF kernel to obtain the best performance. As shown in Table 8.2, LS_Boost performs slightly better than SVM regression.

Table 8.2: Performance comparison of seven machine learning methods. The sequence-template alignments are generated by RAPTOR.

	Family Level		Superfamily Level		Fold Level	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
LS_Boost	86.5%	89.2%	60.2%	74.4%	38.8%	61.7%
SVM (regression)	85.0%	89.1%	55.4%	71.8%	38.6%	60.6%
SVM (classification)	82.6%	83.6%	45.7%	58.8%	30.4%	52.6%
Ada_Boost	82.8%	84.1%	50.7%	61.1%	32.2%	53.3%
Neural Networks	81.1%	83.2%	47.4%	58.3%	30.1%	54.8%
Bayes classifier	69.9%	72.5%	29.2%	42.6%	13.6%	40.0%
Naïve Bayes Classifier	68.0%	70.8%	31.0%	41.7%	15.1%	37.4%

2. SVM classification. The fold recognition problem is treated as a classification problem, and we consider an SVM for classification. The software and kernel we consider is the same as for SVM regression. In this case, one can see that SVM classification performs worse than SVM regression, especially at the superfamily level and the fold level.

3. AdaBoost. Boosting is a procedure that combine the outputs of many "weak" classifiers to produce a powerful "committee". We use the standard AdaBoost algorithm [40] for classification, which is similar to LS_Boost except that it performs classification rather than regression and uses the exponential instead of least-squares loss function. The AdaBoost algorithm achieves a comparable result to SVM classification but is worse than both of the regression approaches, LS_Boost and SVM regression.

4. Neural networks. Neural networks are one of the most popular methods used in machine learning [96]. Here we use a multi-layer perceptron for classification, based on the Matlab neural network toolbox. The performance of the neural network is similar to SVM classification and Adaboost.

5. Bayesian classifier. A Bayesian classifier is a probability based classifier which assigns a sample to a class based on the probability that it belongs to the class [88].

6. Naïve Bayesian classifier. The Naïve Bayesian classifier is similar to the Bayesian classifier except that it assumes that the features of each class are independent, which greatly decreases computation [88]. We can see both Bayesian classifier and Naïve Bayesian classifier obtain poor performance.

Our experimental results show clearly that: (1) The regression based approaches demonstrate better performance than the classification based approaches. (2) LS_Boost performs slightly better than SVM regression and significantly better than the other methods. (3) The computational efficiency of LS_Boost is much better than SVM regression, SVM classification and the neural network.

One of the advantages of our boosting approach over SVM regression is its ability to identify important features, since at each round LS_Boost only chooses a single feature to approximate the alignment accuracy residual. The following are the top five features chosen by our algorithm. The corresponding simple functions associated with each feature are all linear regression functions $y = ax + b$, showing that there is a strong linear relation between the features and the alignment accuracy. For example, from the figure 8.3, we can see that the linear regression function is the best fit.

1. Sequence identity;
2. Total alignment score;
3. Fitness score;
4. Mutation score;
5. Pairwise potential score.

It seems surprising that the widely used z-score is not chosen as one of the most important features. This indicates to us that the z-score may not be the most important feature and redundant. To confirm our hypothesis, we re-trained our model using all the features except all the z-scores. That is, we conducted the same training and test procedures as before, but with the reduced feature set. The results given in Table 8.3 show that for LS_Boost there is almost no difference between using the z-score as an additional

Table 8.3: Comparison of fold recognition performance with zscore and without zscore.

	Family Level		Superfamily Level		Fold Level	
	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
LS_Boost with z-score	86.5%	89.2%	60.2%	74.4%	38.8%	61.7%
LS_Boost without z-score	85.8%	89.2%	60.2%	73.9%	38.3%	62.9%

Table 8.4: Error Analysis of seven machine learning methods for fold level. The sequence-template alignments are generated by RAPTOR.

	Top 1		Top 5	
	mean	std	mean	std
LS_Boost	38.9%	0.027	61.8%	0.036
SVM (R)	38.7%	0.027	60.7%	0.035
SVM (C)	30.4%	0.024	52.8%	0.032
Ada_Boost	32.1%	0.025	53.4%	0.034
NN	30.2%	0.024	55.0%	0.033
B C	13.7%	0.016	40.1%	0.028
N B C	15.1%	0.017	37.3%	0.027

feature or without using it. Thus, we conclude that by using the LS_Boost approach it is unnecessary to calculate z-score to obtain the best performance. This means that we can greatly improve the computational efficiency of protein threading without sacrificing accuracy, by completely avoiding the calculation of the expensive z-score.

To quantify the margin of superiority of LS_Boost over the other machine-learning methods, we use bootstrap method to get the error analysis. After training the model, we randomly sample 600 sequences from Lindahl's benchmark and calculate the sensitivity using the same method as before. We repeat the sampling for 1000 times and get the mean and standard deviation of the sensitivity of each method as listed in table 8.4, 8.5 and 8.6. We can see that LS_Boost method is slightly better than SVM regression and much better than other methods.

Specificity We further examine the specificity of the LS_Boost method with Lindahl's benchmark. All threading pairs are ranked by their confidence score (i.e., the predicted

Table 8.5: Error Analysis of seven machine learning methods for family level. The sequence-template alignments are generated by RAPTOR.

	Top 1		Top 5	
	mean	std	mean	std
LS_Boost	86.6%	0.029	89.2%	0.031
SVM (R)	85.2%	0.031	89.2%	0.031
SVM (C)	82.5%	0.028	83.8%	0.030
Ada_Boost	82.9%	0.030	84.2%	0.029
NN	81.8%	0.029	83.5%	0.030
B C	70.0%	0.027	72.6%	0.027
N B C	68.8%	0.026	71.0%	0.028

Table 8.6: Error Analysis of seven machine learning methods for super-family level. The sequence-template alignments are generated by RAPTOR.

	Top 1		Top 5	
	mean	std	mean	std
LS_Boost	60.2%	0.029	74.3%	0.034
SVM (R)	55.6%	0.029	72.0%	0.033
SVM (C)	45.8%	0.026	58.9%	0.030
Ada_Boost	50.7%	0.028	61.2%	0.031
NN	47.5%	0.027	58.4%	0.031
B C	29.1%	0.021	42.6%	0.026
N B C	31.1%	0.022	41.9%	0.025

alignment accuracy or the classification score if an SVM classifier is used) and the sensitivity-specificity curves are drawn in Figure 8.5, 8.6 and 8.7. Figure 8.6 demonstrates that at the superfamily level, the LS_boost method is consistently better than SVM regression and classification within the whole spectrum of sensitivity. At both the family level and fold level, LS_Boost is a little better when the specificity is high while worse when the specificity is low. At the family level, LS_Boost achieves a sensitivity of 55.0% and 64.0% at 99% and 50% specificities, respectively, whereas SVM regression achieves a sensitivity of 44.2% and 71.3%, and SVM classification achieves a sensitivity of 27.0% and 70.9% respectively. At the superfamily level, LS_Boost has a sensitivity of 8.2% and 20.8% at 99% and 50% specificities, respectively. In contrast, SVM regression has a sensitivity of 3.6% and 17.8%, and SVM classification has a sensitivity of 2.0% and 16.1% respectively. Figure 8.7 shows that at the fold level, there is no big difference between LS_Boost method, SVM regression and SVM classification method.

Computational Efficiency Overall, the LS_Boost procedure achieves superior computational efficiency during both training and testing. By running our program on a 2.53 GHz Pentium IV processor, after extracting the features, the training time is less than thirty seconds and the total test time is approximately two seconds. Thus we can see that our technique is very fast compared to other approaches, in particular the machine learning approaches such as neural networks and SVMs which require much more time to train. Table 8.7lists the running time of several different fold recognition methods. From this table, we can see that the boosting approach is more efficient than the SVM regression method, which is desirable for genome-scale structure prediction. The running time shown in this table does not contain the computational time of sequence-template alignment.

8.6 Conclusions

In this chapter, we propose a new machine learning approach—LS_Boost—to solve the protein fold recognition problem. We use a regression approach which is proved to be both more accurate and efficient than classification based approaches. One of the most significant conclusions of our experimental evaluation is that we do not need to calculate

Table 8.7: Running time of different machine learning approaches.

	Training time	Testing time
LS_Boost	30 seconds	2 seconds
SVM classification	19 mins	26 mins
SVM regression	1 hour	4.3 hours
Neural Network	2.3 hours	2 mins
Naïve Bayes Classifier	1.8 hours	2 mins
Bayes Classifier	1.9 hours	2 mins

the standard z-score, and can thereby achieve a substantial computational savings without sacrificing prediction accuracy. Our algorithm achieves strong sensitivity results compared to other fold recognition methods, including both machine learning methods and z-score based methods. Moreover, our approach is significantly more efficient for both the training and testing phases, which may allow genome-scale scale structure prediction.

```
Algorithm 1: Gradient Boost

• Initialize $F_0(x) = \arg\min_\rho \sum_{i-1}^{N} L(y_i, \rho)$
• For $m = 1$ to $M$ do:
    • Step 1. Compute the negative gradient
          $$\tilde{y}_i = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F_{x_i}}\right]$$
    • Step 2. Fit a model
          $$\alpha_m = \arg\min_{\alpha,\beta} \sum_{i=1}^{N} [\tilde{y} - \beta h(x_i; \alpha)]^2$$
    • Step 3. Choose a gradient descent step size as
          $$\rho_m = \arg\min_{\rho} \sum_{i-1}^{N} L(y_i, Fm - 1(x_i)$$
          $$+\rho h(x_i; \alpha_m))$$
    • Step 4. Update the estimation of $F(x)$
          $$F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m)$$
• end for
• Output the final regression function $F_m(x)$
```

Figure 8.1: Gradient boosting algorithm

```
Algorithm 2: LS_Boost

• Initialize $F_0 = \bar{y} = \frac{1}{N}\sum_i y_i$
• For $m = 1$ to $M$ do:
    • $\tilde{y}_i = y_i - F_{m-1}(x_i, i = 1, \ldots, N)$
    • $(\rho_m, \alpha_m) = \arg\min_{\rho,\alpha} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(x_i; \alpha_m)]^2$
    • $F_m(x) = F_{m-1}(x) + \rho_m h(x; \alpha_m)$
• end for
• Output the final regression function $F_m(x)$
```

Figure 8.2: LS_Boost algorithm

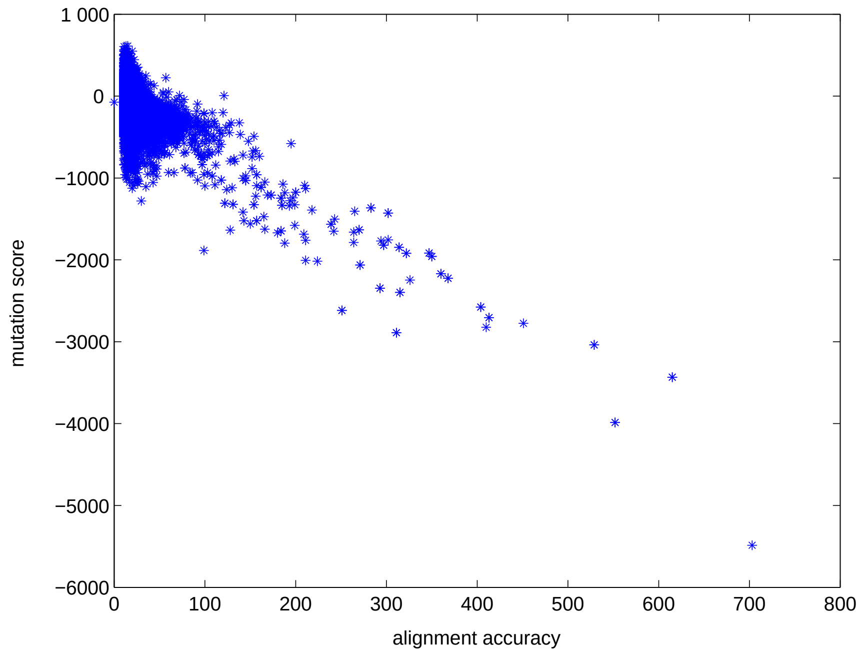

Figure 8.3: The relation between alignment accuracy and mutation score.

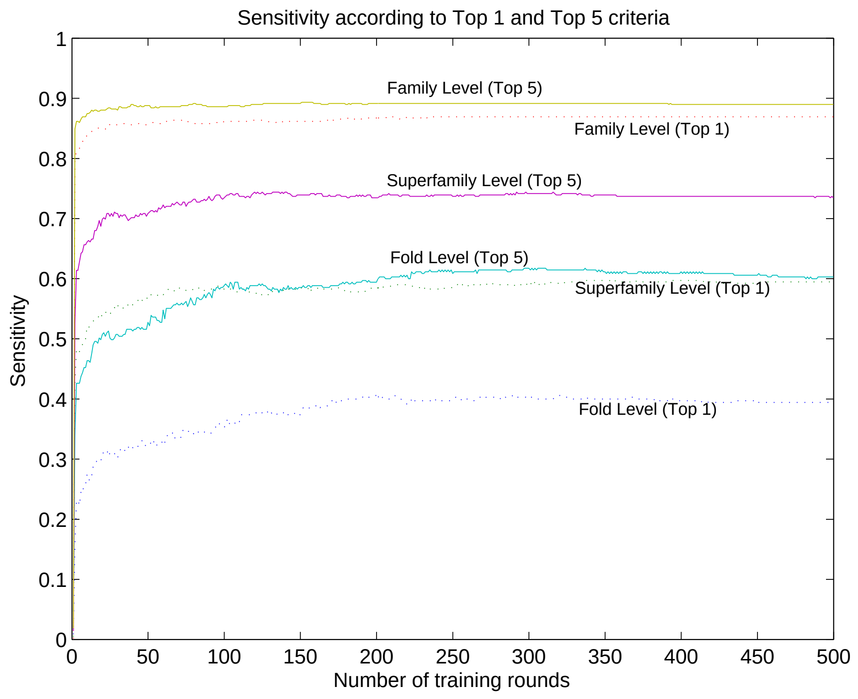

Figure 8.4: Sensitivity curves during the training process.

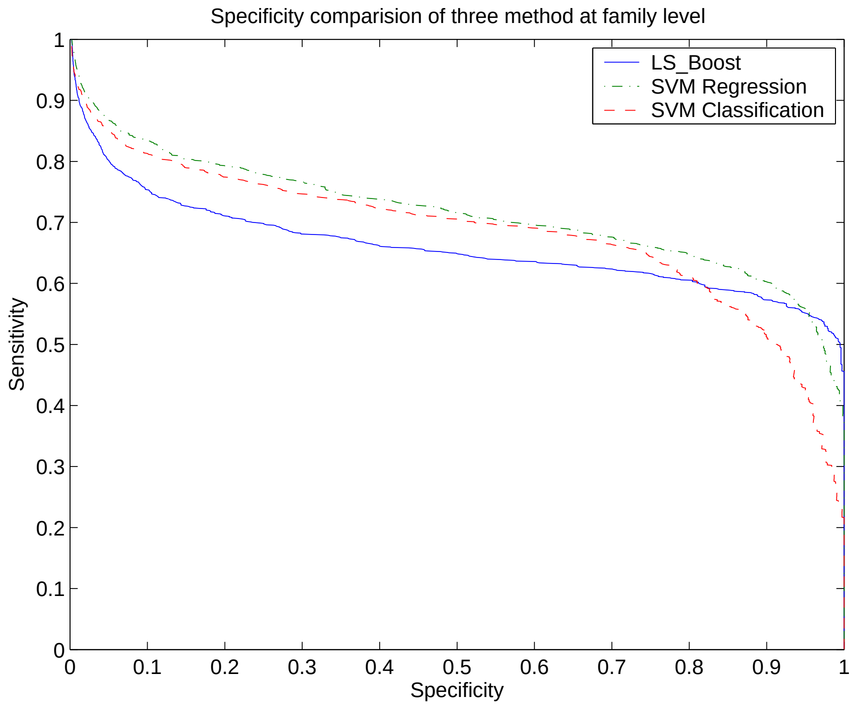

Figure 8.5: Family-level specificity-sensitivity curves on Lindahl's benchmark set. The three methods LS_Boost, SVM regression and SVM classification are compared.

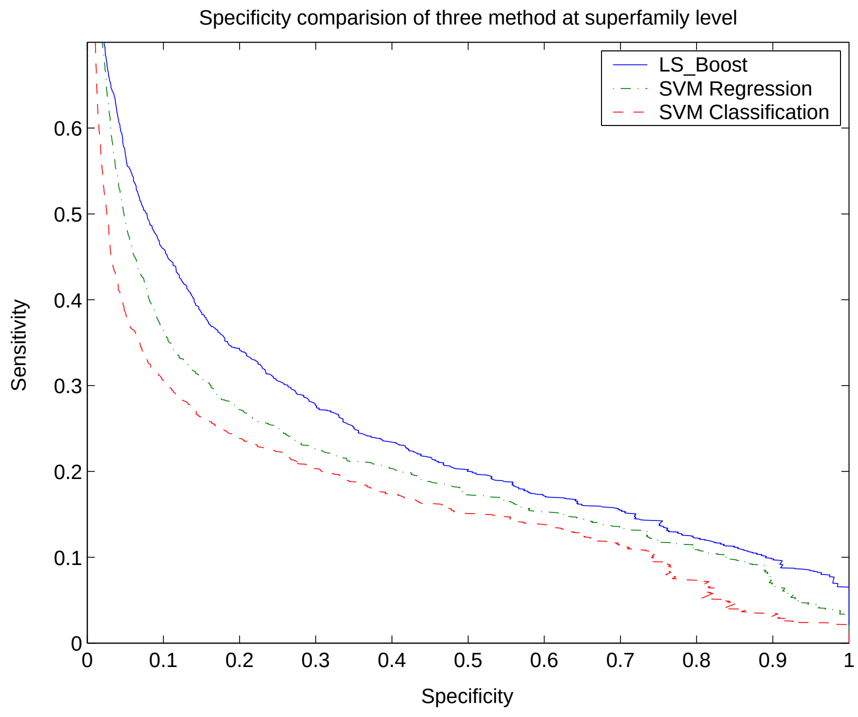

Figure 8.6: Superfamily-level specificity-sensitivity curves on Lindahl's benchmark set. The three methods LS_Boost, SVM regression and SVM classification are compared.

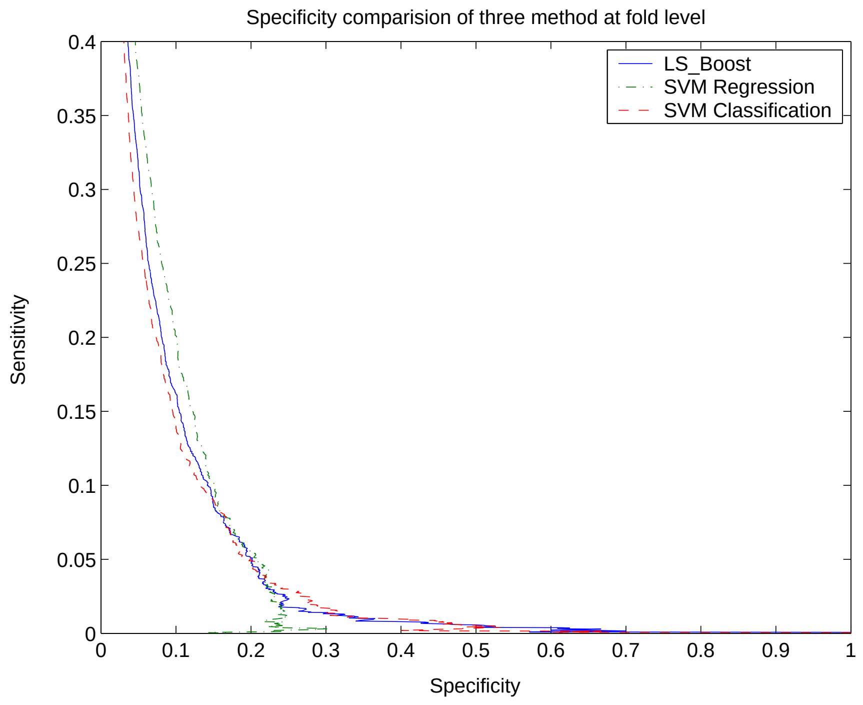

Figure 8.7: Fold-level specificity-sensitivity curves on Lindahl's benchmark set. The three methods LS_Boost, SVM regression and SVM classification are compared.

Chapter 9

Conclusions and Future work

9.1 Conclusions

In this thesis, I focused on graphical models and algorithms for protein problems. For graphical models I presented some of the research on semi-supervised methods on conditional random fields. For protein problems, I proposed a tree decomposition framework to solve both protein structure prediction and protein structure alignment problems. I proposed a computational framework for detection of similar structures of a target protein with sparse NMR data. Also I proposed a new machine learning approach—LS_Boost—to solve the protein fold recognition problem.

In Chapter 4, I present a new semi-supervised training procedure for conditional random fields (CRFs) that can be used to train sequence segmentors and labelers from a combination of labeled and unlabeled training data. My approach is based on extending the minimum entropy regularization framework to the structured prediction case, yielding a training objective that combines unlabeled conditional entropy with labeled conditional likelihood. I apply the new training algorithm to the problem of identifying gene and protein mentions in biological texts, and show that incorporating unlabeled data improves the performance of the supervised CRFs in this case.

In Chapter 5 and 6, I propose a framework to solve both protein structure prediction and structure alignment problems. The method is based on the tree decomposition method. We reveal that the low tree width of protein contact graph determines tree decomposition method is very suitable for protein problems.

In Chapter 7, I presented a computational framework for detection of similar structures of a target protein with sparse NMR data. The model and algorithms proposed here can be used for protein structure prediction even if a limited number of NMR data is available.

In Chapter 8, I proposed a new machine learning approach—LS_Boost—to solve the protein fold recognition problem. I used a regression approach that proved to be both more accurate and efficient than classification based approaches. The algorithm achieves strong sensitivity results compared to other fold recognition methods, including both machine learning methods and z-score based methods. Moreover, it is significantly more efficient for both the training and testing phases that may allow genome-scale scale structure prediction.

9.2 Future work

For semi-supervised CRFs, choosing the regularization parameter γ is a very difficult problem. Currently I just train batch models using different γ values, which might lead to overfitting problems and also is not efficient. In [48], Hastie proposed a method to estimate the regularization parameter for an SVM, by fitting the entire path of an SVM solution for every value of the regularization parameter with essentially the same computational cost as fitting a single SVM model. I would like to adapt the same technique to semi-supervised CRFs.

I would also like to conduct research on how to choose a suitable amount of unlabeled data to combine with labeled data for training CRFs. The size of the unlabeled data set affects both the training time and the accuracy, and a suitable choice of unlabeled training set size can have a significant impact on performance. If the size is too small, training time will be fast but the unlabeled data will have no effect. Conversely, if the size is too big, the unlabeled data can overwhelm the labeled data, leading to overfitting in addition to high computational cost. I would like to consider some of the existing work on generalization bounds, which might provide some theoretical support for the approach I have proposed. Also, I would like to consider the prospects of combining semi-supervised training with boosting.

CRFs have been used in many areas in computational biology. In [79], Segmentation

Conditional Random Fields (SCRFs) were proposed for fold recognition. Compared with traditional graphical models such as Hidden Markov Model (HMM), SCRFs follow a discriminative approach. It has the flexibility to include overlapping or long-range interaction features over the whole sequence, as well as global optimally solutions for the parameters. On the other hand, the segmentation setting in SCRFs makes its graphical structures intuitively similar to the protein 3-D structures and more importantly, provides a framework to model the long-range interactions directly [79]. Also in [109],CRFs was used on RNA secondary structural alignment. The approach has specific features compared with previous methods. For instance the parameters for structural alignment are estimated such that the model can most probably discriminate between correct alignments and incorrect alignments, and has the generalization ability so that a satisfiable score matrix can be obtained even with a small number of sample data without overfitting [109]. Since we have demonstrated the superiority of Semi-CRFs over the general CRFs algorithm, it will be of great interest to apply the Semi-CRFs algorithm to these two problems and see how much gain we can obtain.

Currently in protein structure prediction we build a uniform model for all proteins. But since some protein folds may differ significantly, it might be helpful if we build different models for different folds, both in threading and in fold recognition steps. We can build some typical individual models for different proteins and a probability can be calculated for it. In fact we need to predict the second structure and the accuracy of it is limited (80we can combine it with 3D structure prediction and build a uniform propositional model. $P(3d|sequence) = P(3d|2d) * P(2d|sequence) = P(3d|fold) * P(fold|2d) * P(2d|sequence)$. The unified graphical model can be shown as Figure 9.1. Further research will be done in the future for this model. Machine learning and bioinformatics are the two most promising and developing areas. There are still many mysterious and attractive research topics in these two fields, which will surely lead to some promising and fruitful results.

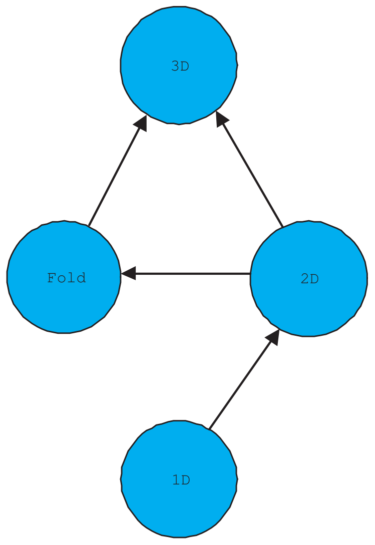

Figure 9.1: A graphical model for protein structure prediction

References

[1] J. Skolnick A. Godzik, A. Kolinski. Topology fingerprint approach to the inverse protein folding problem. *Journal of Molecular Biology*, 227(1):227–238, 1992.

[2] S. Abney. Understanding the yarowsky algorithm. pages 365–395, 2004.

[3] Vincent J.Della Pietra Adam Berger, Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computation Linguistics*, pages 37–91, 1996.

[4] T. Akutsu, M. Hayashida, E. Tomita, and J. Suzuki. protein threading with profiles and constraints. In *The Fourth IEEE Symposium on Bioinformatics and Bioengineering (BIBE)*, pages 537–544, May 2004.

[5] T. Akutsu and S. Miyano. On the approximation of protein threading. *Theoretical Computer Science*, 210:261–275, 1999.

[6] M. Albrecht, D. Hanisch, R. Zimmer, and T. Lengauer. Improving fold recognition of protein threading by experimental distance constraints. *In Silico Biol*, 2(3):325–337, 2002.

[7] N. Alexandrov. SARFing the PDB. *Protein Engineering*, 9:727–732, 1996.

[8] Y. Altun, D. McAllester, and M. Belkin. Maximum margin semi-supervised learning for structured variables. *Advances in Neural Information Processing Systems 18*, 2005.

[9] S. Arnborg, D.G. Corneil, and A. Proskurowski. Complexity of finding embedding in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

[10] F. Bach and M. Jordan. Thin junction trees. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 14. 2002.

[11] C. Bailey-Kellogg, S. Chainraj, and G. Pandurangan. A random graph approach to NMR sequential assignment. In *RECOMB '04: Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*, pages 58–67, New York, NY, USA, 2004. ACM Press.

[12] C. Bailey-Kellogg, A. Widge, J. J. Kelley, M. J. Berardi, J. H. Bushweller, and B. R. Donald. The noesy jigsaw: automated protein secondary structure and main-chain assignment from sparse, unassigned nmr data. *J Comput Biol*, 7(3-4):537–558, 2000.

[13] S. Balev. Solving the protein threading problem by lagrangian relaxation. In *Workshop on Algorithms in Bioinformatics*, pages 182–193, 2004.

[14] Protein Data Bank. *http://www.rcsb.org/pdb/experimental_methods.htm*, 2001.

[15] A. Berger. The improved iterative scaling algorithm: A general introduction. 1997.

[16] A. Berry, P. Heggernes, and G. Simonet. The minimum degree heuristic and the minimal triangulation process. In *Lecture Notes in Computer Science, Vol. 2880*, pages 58–70. Springer Verlag, June 2003.

[17] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of the Workshop on Computational Learning Theory*, pages 92–100, 1998.

[18] P. M. Bowers, C. E. Strauss, and D. Baker. De novo protein structure determination using sparse NMR data. *J Biomol NMR*, 18(4):311–318, December 2000.

[19] S. Boyd and L. Vandenberghe. Convex optimization. *Cambridge University Press*, 2004.

[20] S. Bryant and C. Lawrence. An empirical energy function for threading protein sequence through the folding motif. *Proteins*, 16,1:92–112, 1993.

[21] S.H. Bryant and S.F. Altschul. Statistics of sequence-structure threading. *Current Opinions in Structural Biology*, 5:236–244, 1995.

[22] A. Caprara, R. Carr, S. Istrail, G. Lancia, and B. Walenz. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. *Journal of Computational Biology*, 11(1):27–52, 2004.

[23] A. Caprara and G. Lancia. Structural alignment of largesize proteins via Lagrangian relaxation. In *RECOMB 2002*, pages 100–108. ACM Press, 2002.

[24] V. Castelli and T. Cover. The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Trans. on Information Theory*, 42(6)(3):2102–2117, 1996.

[25] G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14:315–332, 1992.

[26] L. Cheng, F. Jiao, D. Schuurmans, and S. Wang. Variational bayesian image modelling. In *International Conference on Principles of Knowledge Representation and Reasoning*, 2005.

[27] G. M. Clore, A. M. Gronenborn, and A. Bax. A robust method for determining the magnitude of the fully asymmetric alignment tensor of oriented macromolecules in the absence of structural information. *J Magn Reson*, 133(1):216–221, July 1998.

[28] B. E. Coggins and P. Zhou. Paces: Protein sequential assignment by computer-assisted exhaustive search. *J Biomol NMR*, 26(2):93–111, June 2003.

[29] I. Cohen and F. Cozman. Risks of semi-supervised learning. *Semi-Supervised Learning, MIT Press*, pages 55–70, 2006.

[30] A. Corduneanu and T. Jaakkola. Data dependent regularization. *Semi-Supervised Learning*, 2006.

[31] G. Cornilescu, F. Delaglio, and A. Bax. Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *J Biomol NMR*, 13(3):289–302, March 1999.

[32] J. Huang D. Zhou and B. Schölkopf. learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning*, pages 1041–1048, 2005.

[33] T. Navin Lal J. Weston D. Zhou, O. Bousquet and B. Schölkopf. Learning with local and global consistency. *Advances in Neural Information Processing Systems*, pages 321–328, 2004.

[34] J. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The annuals of Mathematics statistics*, 43:1470–1480, 1972.

[35] O. Dror, H. Benyamini, R. Nussinov, and H. Wolfson. MASS: Multiple structural alignment by secondary structures. *Bioinformatics*, 19(Suppl. 1):95–104, 2003.

[36] R. Duda and P. Hart. Understanding the yarowsky algorithm. *Pattern Classification and Scene Analysis, John Wiley & Sons*, pages 365–395, 1973.

[37] Amir E. and McIlraith S. Partition-based logical reasoning. In *International Conference on Principles of Knowledge Representation and Reasoning*, 2000.

[38] A. Erdmann and S. Rule. Rapid protein structure detection and assignment using residual dipolar coupling. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, December 2002.

[39] F. Forbes and N. Peyrard. Hidden markov random field model selection criteria based on mean field-like approximation. *IEEE Transaction on Pattern Recognition and Machine Intelligence*, 25(9):1089–1101, 2003.

[40] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[41] J.H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annuals of Statistics*, 29(5), October 2001.

[42] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[43] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In *Proceedings of International Conference on Intelligent Systems in Molecular Biology*, pages 59–67, 1996.

[44] J. Gibrat, T. Madej, and S. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, pages 377–385, 1996.

[45] D. Goldman, C. Papadimitriou, and S. Istrail. Algorithmic aspects of protein structure similarity. In *FOCS 99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 512–522. IEEE Computer Society, 1999.

[46] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems*, 2004.

[47] T. Haliloglu, A. Kolinski, and J. Skolnick. Use of residual dipolar couplings as restraints in ab initio protein structure prediction. *Biopolymers*, 70(4):548–562, December 2003.

[48] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. In *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2005. MIT Press.

[49] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233:123–138, 1993.

[50] L. Holm and C. Sander. Decision support system for the evolutionary classification of protein structures. 5:140–146, 1997.

[51] L. Hung and R. Samudrala. Accurate and automated classification of protein secondary structure with psicsi. *Protein Sci*, 12(2):288–295, 2003.

[52] Y. Weiss J. Yedidia, W. Freeman. Generalized belief propagation. In *Neural Information Processing Systems*, pages 689–695, 2000.

[53] Y. Weiss J. Yedidia, W. Freeman. Constructing free energy ap- proximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, July 2005.

[54] F. Jiao, S. Wang, C. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *International Committee on Computational Linguistics and the Association for Computational Linguistics*, 2006.

[55] T. Joachims. *Making Large-scale SVM Learning Practical.* MIT Press, 1999.

[56] S. Jokisch and H. Müller. Inter-point-distance-dependent approximate point set matching. Technical Report Research Report No. 653, July 1997.

[57] D. Jones. GenTHREADER: An efficient and reliable protein fold recognition method for genomic sequences. *Journal of Molecular Biology*, 287:797–815, 1999.

[58] D. Jones and J. Thornton W. Taylor. A new approach to protein fold recognition. *Nature*, 358:86–98, 1992.

[59] D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, 1999.

[60] M. Jordan and C. Bishop. An introduction to graphical models. manuscript, 2005.

[61] M. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[62] M. Jordan K. Murphy, Y. Weiss. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*, pages 467–475, 1999.

[63] S. Thrun K. Nigam, A. McCallum and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2/3):135–167, 2000.

[64] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.

[65] H. Kamisetty, C. Bailey-Kellogg, and G. Pandurangan. An efficient randomized algorithm for contact-based NMR backbone resonance assignment. *Bioinformatics*, 22(2):172–180, January 2006.

[66] D. Kim, D. Xu, J. Guo, K. Ellrott, and Y. Xu. PROSPECT II: Protein structure prediction method for genome-scale applications. *Protein Engineering*, 16(9):641–650, 2003.

[67] R. Kolodny and N. Linial. Approximate protein structural alignment in polynomial time. *PNAS*, 101(33):12201–12206, 2004.

[68] A.M.C.A. Koster, S.P.M. van Hoesel, and A.W.J. Kolen. Solving frequency assignment problems via tree-decomposition. Research Memoranda 036, Maastricht : METEOR, Maastricht Research School of Economics of Technology and Organization, 1999. available at http://ideas.repec.org/p/dgr/umamet/1999036.html.

[69] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[70] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *Proceedings of the Fifth Annual International Conference on Research in Computational Molecular Biology*, pages 193–202. ACM Press, 2001.

[71] G. Lancia and S. Istrail. Protein structure comparison: Algorithms and applications. In *Mathematical Methods for Protein Structure Analysis and Design*, volume 2666 of *Lecture Notes in Computer Science*, pages 1–33, 2003.

[72] C. J. Langmead, A. Yan, R. Lilien, L. Wang, and B. R. Donald. A polynomial-time nuclear vector replacement algorithm for automated NMR resonance assignments. *J Comput Biol*, 11(2-3):277–298, 2004.

[73] R. Lathrop. The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Engineering*, 7:1059–1068, 1994.

[74] R. Lathrop. An anytime local-to-global optimization algorithm for protein threading in theta (m2n2) space. *J Comput Biol*, 6(3-4):405–418, 1999.

[75] R. Lathrop and T. Smith. A branch-and-bound algorithm for optimal protein threading with pairwise (contact potential) amino acid interactions. In *Proceedings of the 27th Hawaii International Conference on System Sciences*. IEEE Computer Society Press, 1994.

[76] C. Lemmen and T. Lengauer. Computational methods for the structural alignment of molecules. *Journal of Computer-Aided Molecular Design*, 14:215–232, 2000.

[77] W. Li and A. McCallum. Semi-supervised sequence modeling with syntactic topic models. *Proceedings of Twentieth National Conference on Artificial Intelligence*, pages 813–818, 2005.

[78] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *Journal of Molecular Biology*, 295:613–625, 2000.

[79] Y. Liu, J. Carbonell, P. Weigele, and V. Gopalakrishnan. Segmentation conditional random fields (scrfs): A new approach for protein fold recognition. In *RECOMB '05: Proceedings of the eighth annual international conference on Resaerch in computational molecular biology*. ACM Press, 2005.

[80] ED M. Jordan. *Learning in graphical models*. Cambridge MA: MIT Press, 1999.

[81] D. MacKay. Introduction to Monte Carlo methods. In M. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer Academic Press, 1998.

[82] A. McCallum. Mallet: A machine learning for language toolkit. 2002.

[83] A. McCallum, D. Freitag, and F. Perera. Maximum entropy markov models for information extraction and segmentation. In *International Conference on Machine Learning*, 2000.

[84] R. McDonald and F. Pereira. Identifying gene and protein mentions in text using conditional random fields. In *BMC Bioinformatics 2005*, 2005.

[85] G. McLachlan and D. Peel. *Finite Mixture Models.* Wiley, 2000.

[86] J. Meiler and D. Baker. Rapid protein fold determination using unassigned NMR data. *Proc Natl Acad Sci U S A*, 100(26):15404–15409, December 2003.

[87] J. Meiler, W. Peti, and C. Griesinger. Dipocoup: A versatile program for 3d-structure homology comparison based on residual dipolar couplings and pseudocontact shifts. *J Biomol NMR*, 17(4):283–294, August 2000.

[88] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine learning, neural and statistical classification,(edit collection).* Elllis Horwood, 1994.

[89] G. L. Miller, S. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of ACM*, 44(1):1–29, 1997.

[90] T. Minka. A comparison of numerical optimizers for logistic regression. 2003.

[91] A. Mordecai. *Nonlinear Programming: Analysis and Methods.* Dover Publishing, 2003.

[92] G. A. Mueller, W. Y. Choy, D. Yang, J. D. Forman-Kay, R. A. Venters, and L. E. Kay. Global folds of proteins with low densities of noes using residual dipolar couplings: application to the 370-residue maltodextrin-binding protein. *J Mol Biol*, 300(1):197–212, June 2000.

[93] A. Murzin, S. Brenner, T. Hubbard, and C. Chothia. SCOP:a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[94] R. Neal. Probabilistic inference using markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[95] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000.

[96] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[97] S. Pietra, V. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–393, 1997.

[98] Y. Qu, J. Guo, V. Olman, and Y. Xu. Protein structure prediction using sparse dipolar coupling data. *Nucl. Acids Res.*, 32(2):551–561, 2004.

[99] R. Bucy R. Kalman. New results in linear filtering and prediction theory. *Transactions of the ASME - Journal of Basic Engineering*, 83:95–107, 1961.

[100] J. Laurie Snell R. Kindermann. *Markov Random Fields and Their Applications*. Amer Mathematical Society, 1980.

[101] T. Smith R. Lathrop. Global optimum protein threading with gapped alignment and empirical pair score functions. *Journal of Molecular Biology*, 255:641–665, 1996.

[102] T. Lenguaer R. Thiele, R. Zimmer. Protein threading by recursive dynamic programming. *Journal of Molecular Biology*, 290:757–779, 1999.

[103] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, Feb. 1989.

[104] Adwait Ratnaparkhi. A simple introduction to maximum entropy model for natual language processing. *Techonical Report 97-08, Institude for Research in Cognitive Science, University of Pennsylvania*, 1997.

[105] S. Roberts, R. Everson, and I. Rezek. Maximum certainty data partitioning. *Pattern Recognition*, pages 833–839, 2000.

[106] N. Robertson and P.D. Seymour. Graph minors. II. algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[107] C. A. Rohl and D. Baker. De novo determination of protein backbone structure from residual dipolar couplings using rosetta. *J Am Chem Soc*, 124(11):2723–2729, March 2002.

[108] L. Rychlewski, L. Jaroszewski, W. Li, and A. Godzik. Comparison of sequence profiles. strategies for structural predictions using sequence information. *Protein Science*, 9(2):232–241, 2000.

[109] K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 21(2):237–242, 2005.

[110] J. Shi, T. Blundell, and K. Mizuguchi. FUGUE: Sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *Journal of Molecular Biology*, 310:243–257, 2001.

[111] I. Shindyalov and P. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11(9):739–747, 1998.

[112] A. Singh and D. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proceedings of International Conference on Intelligent Systems in Molecular Biology*, pages 284–93, 1997.

[113] J. Skolnick and D. Kihara. Defrosting the frozen approximation: Prospector–a new approach to threading. *Proteins*, 42(3):319–331, 2001.

[114] J. Gibrat T. Madej and S. Bryant. Threading a database of protein cores. *Proteins: Structure, Function and Genetics*, 23:356–369, 1995.

[115] F. Tian, H. Valafar, and J. H. Prestegard. A dipolar coupling based strategy for simultaneous resonance assignment and structure determination of protein backbones. *J Am Chem Soc*, 123(47):11791–11796, November 2001.

[116] D. Titterington, A. Smith, and U.Makov. *Statistical Analysis of Finite Mixture Distributions*. Chichester,UK:John Wiley and Sons, 1985.

[117] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[118] O. Verbitsky. On the largest common subgraph problem, 1994. Unpublished manuscript.

[119] S. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated training of conditional random fields with stochastic meta-descent. In *Proceedings of the 23th International Conference on Machine Learning*, 2006.

[120] B. Wallner and A. Elofsson. *Protein Sci*, 12(5):1073–1086, May 2003.

[121] L. Wang and B. R. Donald. An efficient and accurate algorithm for assigning nuclear overhauser effect restraints using a rotamer library ensemble and residual dipolar couplings. *Proc IEEE Comput Syst Bioinform Conf*, pages 189–202, 2005.

[122] Z. Ghahramani X. Zhu and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *Proceedings of the 20nd International Conference on Machine Learning*, pages 912–919, 2003.

[123] J. Xu. *Protein Structure Prediction by Linear Programming*. PhD thesis, University of Waterloo, Waterloo, Canada, August 2003.

[124] J. Xu. Fold recognition by predicted alignment accuracy. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 2(2):157–165, April 2005.

[125] J. Xu. Protein fold recognition by predicted alignment accuracy. *IEEE Transactions on Computational Biology and Bioinformatics*, 2:157 – 165, 2005.

[126] J. Xu. Rapid protein side-chain packing via tree decomposition. In *Research in Computational Molecular Biology*, 2005.

[127] J. Xu. Rapid side-chain packing via tree decomposition. In *Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005)*. 2005.

[128] J. Xu, F. Jiao, and B. Berger. A tree-decomposition approach to protein structure prediction. In *In IEEE Computational Systems Bioinformatics Conference*, 2005.

[129] J. Xu, F. Jiao, and B. Berger. A parameterized algorithm for protein structure alignment. In *Annual International Conference on Research in Computational Molecular Biology*, 2006.

[130] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.

[131] J. Xu, M. Li, G. Lin, D. Kim, and Y Xu. Protein threading by linear programming. pages 264–275, Hawaii, USA, 2003. Biocomputing: Proceedings of the 2003 Pacific Symposium.

[132] Y. Xu, D. Xu, Oakley H. Crawford, and J. Ralph Einstein. A computational method for NMR-Constrained protein threading. *Journal of Computational Biology*, 7(3-4):449–467, 2000.

[133] Y. Xu, D. Xu, D. Kim, V. Olman, J. Razumovskaya, and T. Jiang. Automated assignment of backbone NMR peaks using constrained bipartite matching. *Computing in Science and Engg.*, 4(1):50–62, January 2002.

[134] Y. Xu, D. Xu, and V. Olman. A practical method for interpretation of threading scores: an application of neural networks. *Statistica Sinica Special Issue on Bioinformatics*, 12:159–177, 2002.

[135] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[136] X. Yuan and C. Bystroff. Non-sequential structure-based alignments reveal topology-independent core packing arrangements in proteins. *Bioinformatics*, 27:1010–1019, 2005.

[137] D. Zheng, Y. J. Huang, H. N. Moseley, R. Xiao, J. Aramini, G. V. Swapna, and G. T. Montelione. Automated protein fold determination using a minimal NMR constraint strategy. *Protein Sci*, 12(6):1232–1246, June 2003.

[138] J. Zhu and Z. Weng. FAST: A novel protein structure alignment algorithm. *Proteins: Structure Function, and Bioinformatics*, 2004. In Press.