Continuous Graphical Models for Static and Dynamic Distributions: Application to Structural Biology

Narges Sharif Razavian

CMU-LTI-13-015

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Christopher James Langmead
Jaime Carbonell
Aarti Singh
Le Song

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Abstract

Generative models of protein structure enable researchers to predict the behavior of proteins under different conditions. Continuous graphical models are powerful and efficient tools for modeling static and dynamic distributions, which can be used for learning generative models of molecular dynamics.

In this thesis, we develop new and improved continuous graphical models, to be used in modeling of protein structure. We first present von Mises graphical models, and develop consistent and efficient algorithms for sparse structure learning and parameter estimation, and inference. We compare our model to sparse Gaussian graphical model and show it outperforms GGMs on synthetic and Engrailed protein molecular dynamics datasets. Next, we develop algorithms to estimate Mixture of von Mises graphical models using Expectation Maximization, and show that these models outperform Von Mises, Gaussian and mixture of Gaussian graphical models in terms of accuracy of prediction in imputation test of non-redundant protein structure datasets. We then use non-paranormal and nonparametric graphical models, which have extensive representation power, and compare several state of the art structure learning methods that can be used prior to nonparametric inference in reproducing kernel Hilbert space embedded graphical models. To be able to take advantage of the nonparametric models, we also propose feature space embedded belief propagation, and use random Fourier based feature approximation in our proposed feature belief propagation, to scale the inference algorithm to larger datasets. To improve the scalability further, we also show the integration of Coreset selection algorithm with the nonparametric inference, and show that the combined model scales to large datasets with very small adverse effect on the quality of predictions. Finally, we present time varying sparse Gaussian graphical models, to learn smoothly varying graphical models of molecular dynamics simulation data, and present results on CypA protein.

Acknowledgments

This thesis would not be possible without the great guidance of my advisor, professor Christopher James Langmead, who supported me and gave me ideas for the past four years. Thanks you! Also I would like to express my utmost gratitude to the members of my thesis committee, professor Jaime Carbonell, professor Aarti Singh, and professor Le Song. I learned a lot from my interactions with you!

I also want to thank all my friends in Pittsburgh, and specially at LTI, for being like a family for me for the past six years. My awesome officemates, Mehrbod(Sharifi), Sukhada(Palkar), Dirk(Hovy), and last but not least, Jahn(Heymann)! Thanks for making the office a place I look forward to come to every day including the weekends! Also my wonderful friends Seza(Dogruoz), Archna(Bhatia), Wang(Ling), Jose(Portelo), Mark(Erhardt), Manaal(Faruqui), Nathan(Schneider), Laleh(Helal), Derry(Wijaya), Meghana(Kshirsagar), Bhavna(Dalvi), Jeff(Flanigan), Prasanna(Kumar), Pallavi(Baljekar), Avner(Maiberg), Nisarga(Markandaiah), Reyyan(Yeniterzi), Subho (Moitra), Hetu(Kamisetty), Arvind(Ramanathan),Sumit(Kumar Jha), Rumi(Naik), Linh(Nguyen), Andreas(Zollmann), Viji(Manoharan), Oznur(Tastan), Reza(Zadeh), Wei(Chen), Yi-Chia(Wang), Sanjika(Hewavitharana), Matthias(Eck), Joy(Zhang), Amr(Ahmed), and many many others, thanks for being source of so much excitement and fun and kindness and happiness and inspiration all the time. Also, a special thanks goes to my mentors and professors in LTI, including Stephan Vogel, who is one of the best mentors I've ever had, Maxine Eskenazi , Bob Frederking, Noah Smith, Karen Thickman, Roni Rosenfeld, Chris Dyre, Jaime Carbonell(again!), Alex Smola, also Stacey Young, and Mary Jo Bensasi for being someone I could count on on every aspect of life. Additionally, Amir(Moghimi), Akram(Kamrani), Shaghayegh(Sahebi), Behnaz(Esmaili), Aida(Rad), Haleh(Moezi), Mahin(Mahmoudi): you guys have always had my back here and I will forever remember this and feel so lucky to have you as friends.

Finally I would like to specially thank my family, who kept giving me love and happiness even through Skype, to remind me that in good or bad times, if you have loved ones who have your back, you have everything :) I love you guys, and I can not be more thankful to have you have my back! This thesis is dedicated to you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many application domains, such as structural biology, deal with large and complicated probabilistic distributions. Understanding the dynamics of proteins, and predicting their behavior under certain conditions, for instance, requires the discovery of the underlying distributions that govern the fluctuations of atoms collectively, in the large protein complex. The size of these models can range from tens to several thousands of amino acids. Currently, probabilistic graphical models have emerged as one of the most successful approaches that can model complicated and large multivariate distributions, in structural biology, as well as other application domains such as genomics, natural language processing, and finance. Graphical models are probabilistic tools that represent complicated multivariate probabilistic distributions in a compact form, and allow efficient parameter estimation and inference.

There are two main families of graphical models: Bayesian Networks, and Markov Random Fields. Bayesian Networks(BNs) are directed acyclic graphical models that describe the casual relationships between individual variables in a multivariate distribution. Markov Random Fields(MRFs), are undirected graphs defined over cliques of interacting variables without the need to set specific casual link between them. In both of these families, the main idea is to $factorize$ the complicated distribution into a normalized product of a set of conditional probabilities, or $factors$, while leveraging the conditional independences reflected in the application.

This factorization leads to more compact and robust estimated models from the data. In this thesis we focus on MRFs, because the structures we deal with are not acyclic and only MRFs can model such structures.

Markov random fields allow a diverse set of factors to be specified. However there's a trade-off between the representational power of the model, and computational complexity of calculating the probability of each query term. One group of graphical models try to make the learning and inference more efficient, by using parametric forms which have fewer parameters to estimate, and sometimes have closed form analytical solutions to certain queries. Gaussian graphical models [24],[5] are an example of such models. A problem with Gaussian graphical models is that the data is not usually Gaussian, and so as we will see in this thesis, we propose models to learn another parametric graphical model, based on von-Mises distribution[23], which as we will see, is a more accurate model of protein structure when the protein structure is specified via its torsion angles. These parametric models have the shortcoming that they lead to unimodal marginals for each variable, which is not realistic. Mixture models, Semi-parametric[45],[38], and non-parametric [78],[79] graphical models can model complex distributions with multi-modal and arbitrary marginals. In this thesis we will also focus on using, developing and improving these models for discovering the generative model of protein structure. Specifically, we will focus on improving the structure learning, inference, and scalability of these graphical models.

In this thesis, our main application is computational structural biology. We focus on developing graphical models that can model the protein structure from Molecular Dynamics(MD) simulation data. MD simulations are often used to characterize conformational dynamics of proteins[34], and the simulations are performed by numerically integrating Newton's laws of motion for a set of atoms. These simulations generate several snapshots of protein structures, as it fluctuates within certain chemical and biological conditions. Several options are available for representing the generated protein structures: Atom coordinates are the most straightforward method, however they are not translation invariant, and require alignment of structure before further analysis can be done. Pairwise distances and internal dihedral angles are alternative methods

of representation which are translation and rotation invariant, and among the two, dihedral angle method represents the structure more compactly. Because of this, most of this thesis focuses on graphical models which can handle angular variables. The specific challenge that need to be addressed in this setting is the ability to model large complex set of variables, whose distribution can be angular and potentially multi-modal. Also, the interactions between the variables can be very complex, and can include several conditional independencies. So the models developed should be able to estimate and take advantage of the inherent dependency structure. In this thesis, we present a collection of probabilistic graphical models, which can handle datasets with these challenges. We first present von Mises and mixture of von Mises graphical models, which allow us to handle protein's dihedral angle representation. Then we focus on nonparametric models, which not only are able to handle angular variables, but also have the power to combine multiple types of variables within same model. Additionally, a particular property of the molecular dynamics simulation datasets is that the data is not identically distributed in longer simulation time spans. So, as part of our thesis, we also focus on modeling non-iid datasets, using time-varying graphical models. There are currently existing time-varying Gaussian graphical models and time-varying discrete graphical models. In this thesis. we develop the smoothly varying $Sparse$ Gaussian graphical model, which can model molecular dynamics simulation data.

Specifically, Part I includes related work and our contributions in parametric group of graphical models for continuous angular variables. In chapter 3 we will focus on structure and parameter learning and inference in von-Mises graphical models, and in chapter 4, we will present mixture of von Mises graphical models, which can handle multi-modality. Part II of the thesis includes the related work and our contributions in semi-parametric and non-parametric group of graphical models. In chapter 6, we will focus on structure learning and inference in non-parametric graphical models, specifically for reproducing kernel Hilbert space embedded graphical models. And chapter 7 includes our solutions for the problem of scalability in the nonparametric graphical models. Finally in Part III of the thesis, we focus on time-varying graphical models for non-iid samples. In chapter 8, we present results of learning sparse time-varying

Gaussian graphical models and its application to CypA enzyme dynamics modeling. Finally in chapter 9 we will present our concluding remarks and directions for future work.

1.1 Thesis Statement

We focus on development of graphical models to learn improved generative models of continuous variables, in particular useful for modeling angular protein structure data. Our contribution includes:

(1) Algorithms to perform sparse structure learning, parameter estimation, and inference, in novel von-Mises graphical models, and apply them to develop better generative models of protein structure,

(2) Algorithms to perform learning for Mixture of von mises, capable of handling weighted input training data,

(3) Comparision of available methods to perform structure learning for non-parametric (reproducing kernel Hilbert space embedded) graphical models,

(4) Developement of novel algorithm of Belief Propagation in Feature Space, and using Fourier Random Projection feature approximation to scale nonparametric Belief Propagation,

(5) Using Coreset Subsampling method to scale Kernel Belief Propagation,

(6) Developing Time-Varying $sparse$ Gaussian graphical model, to estimate better generative models of MD simulation data.

Part I

Parametric Continuous Graphical Models for Structure Modeling

Chapter 2

Background and Related Work in Parametric Graphical Models of Continuous variables

A protein is a linear chain of smaller molecules known as amino acids. The three dimensional structure of a protein can be defined in terms of the Cartesian coordinates of the constituent atoms, or equivalently (and with fewer parameters), in terms of a series of dihedral angles. For example, Figure 2.1 depicts a toy protein consisting of two amino acids (real proteins have dozens to thousands of amino acids). The dihedral angles in this figure are denoted using the conventional names used in biochemistry: ϕ, ψ, ω, χ_1, χ_2, χ_3, and χ_4. Unlike the figure, a protein's structure isn't static. Rather, each protein samples from an underlying distribution over configurations (known as the Boltzmann distribution) according to the laws of physics. Characterizing these distributions is very important for understanding the biological function of these complex molecules. In this thesis we will use this domain as our application, because all challenges associated with large complex continuous systems of variables are reflected in this application domain.

Representing protein structures as a sequence of dihedral angles is desirable, since this repre-

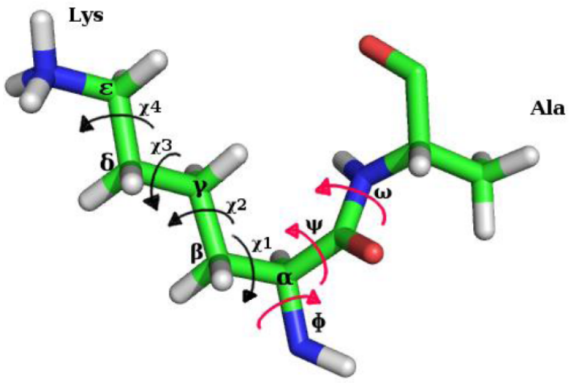

Figure 2.1: Backbone and side-chain dihedral angles of a di-peptide Lys-Ala protein. Picture from [68]

sentation is compact and translation and rotation invariant. However, dealing with large complex sets of angular variables, with potentially multi-modal distributions is an unsolved problem. In this chapter, we cover the existing approaches in parametric graphical model domain, which provide some solutions to this challenge. We will start by providing basic background about undirected graphical models, and then review existing graphical models that can deal with the multivariate complex angular distributions.

2.1 Background on Undirected Graphical Models

Undirected Graphical Models (UGMs), also known as Markov random fields, represent a multivariate distribution as a normalized product of factors, each of which is defined over a clique of the variables on the graph structure. Usually in a UGM, the size of the cliques is limited to two, so the factors are defined over single and pairs of variable (called node and edge potentials respectively).

Figure 2.2 shows an example of a general multivariate distribution and the corresponding undirected graphical model. In this example, a distribution over random variable X, is factorized into a normalized set of node and edge factors, f_i and f_{ij}. The normalization term, sometimes

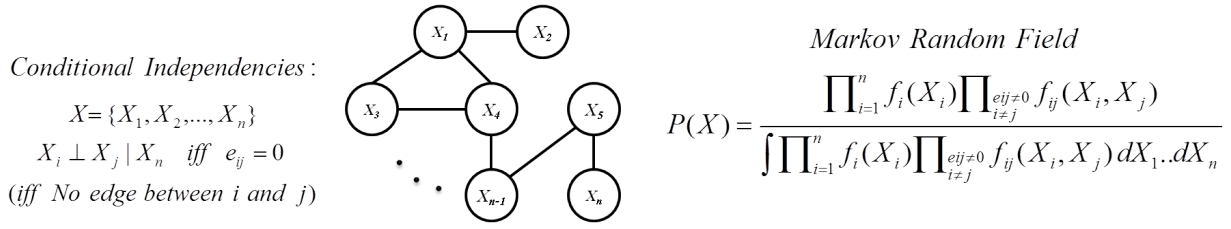

Figure 2.2: Markov Random Field Example

also called the partition function, is computationally very expensive to calculate, since we have to sum over all possible values of all variables. A procedure called $inference$ usually calculates the partition function, as well as the marginal and conditional probabilities of interest related to our graphical model.

Inference in UGMs corresponds to evaluating the probability of any query term, given some observations. There are several algorithms for performing the inference. Belief Propagation (BP) [63] is currently the most common approach to calculate the marginal probabilities and approximate the partition function. Belief Propagation, also known as sum-product algorithm, is a message passing algorithm, in which variables send $messages$ to their neighbors, where each $message$ contains the conditional probability of the neighbor given the observed source variable, after marginalizing all other nodes from the leaf nodes up to that neighboring node. In a tree-graphical model, BP guarantees to find the correct partition function, as well as any marginal and conditional distribution, very efficiently.

A variant of this algorithm, loopy belief propagation [57], is used on loopy UGMs. While loopy belief propagation is missing the convergence guaranties associated with BP, Yedida et. al. [90] showed that loopy belief propagation converges to Bethe approximation of the free energy of the UGM, and presented Generalized BP, which performs message passing between clusters of nodes instead of individual nodes, and approximates the free energy more accurately. Other variants of the BP exist, such as Expectation Propagation [56], and Gaussian mixture belief propagation[83], and particle belief propagation[31].

In this thesis we specifically focus on methods available for learning and inference in continuous graphical models, since the application that we focus on (computational structural biology) uses contentious variables for representation of the data. In the next section we will see the problem formation for computational structural biology, and then review the state of the art in continuous undirected graphical models.

2.2 Learning and Inference in Discrete and Parametric Graphical Models

2.2.1 Discrete and Parametric Graphical Models

The first solution that comes to mind when dealing with continuous variables is the discretization of the value into distinct categories, and then applying all existing methods of structure prediction in discrete domain to the data. In protein structure modeling, side chain conformations are usually categorized into some specific discrete states called Rotamers.[32][28].

In practice, this approach has several shortcomings: Discretization of a continuous value into very large bins introduces inaccuracies into the data. One can avoid this by increasing the granularity of the discretization. However, as the number of discrete categories increases, the graphical model faces the scalability problem.

For instance, in a pairwise graphical model, given an average p discrete states for each node, and average degree d of the graph, the calculations required for messages passing between each node is is $O(p^{d+1})$, which in reasonably large graphs such as protein models, or gene expression networks, quickly becomes intractable. On the other hand, high granularity results in increased number of parameters to estimate, and consequently, the model has a higher change of overfitting to the training data.

At least two solutions to these problems are available: First, using $parametric$ families such as Gaussian, Von Mises, or other continuous families that allow the model to be specified with

fewer parameters, that leads to more data efficiency, and less over-fitting. Second solution is $nonparametric$ and $semiparametric$ kernel based methods, which allows us to smooth the distribution around the samples, and increase model complexity based on the availability of the data. In the rest of this chapter we will cover existing work related to the parametric solutions.

2.2.2 Gaussian Graphical Models

Gaussian graphical models are one of the most commonly used parametric models, which assume that the marginal distribution of every variable in the model is Gaussian. In practice, data is very rarely distributed as Gaussian. However, since Gaussian graphical models(GGMs) have analytical and closed form solutions for all inference queries, and are computationally very efficient to estimate and use, they are very popular in continuous domains.

Parameter estimation in GGM models is equivalent to estimating inverse covariance matrix from the training data. Friedman et. al.[24] proposed a coordinate descent algorithm to estimate sparse covariance matrix using graph-lasso. Banerjee et. al. [5], formulate the sparse inverse covariance estimation estimation in GGM as a convex optimization problem, which they solve using block coordinate descent algorithm. They used L1-regularization penalty over the elements of the inverse covariance matrix. We use this model as one of the baselines in our experiments.

2.2.3 Von-Mises Graphical Models

Gaussian Graphical model make the assumption that the variables are all distributed as Gaussian. In certain domains, such as protein angle data, this assumption is incorrect, since angular variables are limited between $-\pi$ and π, and special techniques are required for taking their average or standard deviation. Von Mises is a distribution which provides a more direct formulation of angular data.

Von-Mises distribution is used in directional statistics to model angles and other circularly-distributed variables [23]. It closely approximates the wrapped normal distribution[12], but has

the advantage that it is more tractable, mathematically [52]. Additionally, von Mises distribution can be generalized to distributions over the $(p-1)$-dimensional sphere in $\Re^p$, where it is known as the von Mises-Fisher distribution.

Wrapped normal distribution for angle $\theta \in [0, 2\pi]$ is defined as an infinite sum of the wrappings of a Gaussian distribution around the unit circle:

$$f_{WN}(\theta; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \sum_{k=-\infty}^{\infty} \exp\left[\frac{-(\theta - \mu + 2\pi k)^2}{2\sigma^2}\right],$$

where μ and σ are the mean and standard deviation of the unwrapped distribution, respectively. The von Mises distribution, which is also known as the circular normal distribution, has a more compact representation given by:

$$f_{VM}(\theta; \mu, \kappa) = \frac{\exp\{\kappa\cos(\theta - \mu)\}}{2\pi I_0(\kappa)}$$

where $I_0(\kappa)$ is the modified Bessel function of order 0, and the parameters μ and $1/\kappa$ are analogous to μ and σ^2 (the mean and variance) in the normal distribution. We note that κ is known as the *concentration* of the variable, and so high concentration implies low variance.

Unlike the wrapped normal distribution, the von Mises distribution belongs to the exponential family and can be extended to higher dimension. The bivariate von Mises distribution [49] over θ_1 and θ_2, for example, can be defined as:

$$f(\theta_1, \theta_2) = \frac{\exp\{\left[\sum_{i=1}^{2} \kappa_i \cos(\theta_i - \mu_i)\right] + \vec{K}_1 M \vec{K}_2^T\}}{Z_c(\mu_1, \mu_2, \kappa_1, \kappa_2, M)},$$

where μ_1 and μ_2 are the means of θ_1 and θ_2, respectively, κ_1 and κ_2 are their corresponding concentrations, $\vec{K}_1 = [\cos(\theta_1 - \mu_1), \sin(\theta_1 - \mu_1)]$, $\vec{K}_2 = [\cos(\theta_2 - \mu_2), \sin(\theta_2 - \mu_2)]$, M is a 2×2 matrix corresponding to their correlation, and $Z_c(\cdot)$ is the normalization constant.

The bivariate von Mises probability density can also be defined as:

$$f(\theta_1, \theta_2) = \frac{\exp\left\{\left[\sum_{i=1}^{2} \kappa_i \cos(\theta_i - \mu_i)\right] + \lambda g(\theta_1, \theta_2)\right\}}{Z_s(\mu_1, \mu_2, \kappa_1, \kappa_2, \lambda)},$$

where μ_1, μ_2, κ_1, and κ_2 are as previously defined, $g(\theta_1, \theta_2) = \sin(\theta_1 - \mu_1)\sin(\theta_2 - \mu_2)$, and λ is a measure of the dependence between θ_1 and θ_2. This formulation, known as the *sine variant*, is generally preferred because it only requires five parameters and is easily expandable to more than 2 variables, as will be demonstrated in the next section.

Mardia et. al. provide bivariate[50] and also multivariate[51] von Mises models applied to protein angles, and provide an algorithm based on full pseudo-likelihood to perform parameter estimation.[29] Their formulation of pseudo-likelihood is based on the fact that univariate marginals of the multivariate von-Mises distribution have closed form, and thus can be optimized using gradient descent. They provide experiments over a tri-variate model and use Gibbs sampling for inference. All these models assume a fully connected graph structure. They also perform Gibbs sampling for inference, which becomes slow for larger models.

Boomsma et. al. [8], model protein sequence as a dynamic Bayesian network, in which hidden states generate backbone angle pairs (ϕ and ψ, for each residue) from a bivariate von Mises distribution. Figure 2.3 shows the generative model of this system. They use Expectation Maximization, and a modified version of forward backward algorithm to perform parameter estimation.

This model assumes a simple chain structure, with dependencies only between immediate neighbor residues, which is incorrect due to the 3D structure of the protein.

In the chapter 4, we develop a general sparse structure learning method, in addition to parameter estimation, and fast inference based on Message Expectation Propagation for von Mises graphical models.

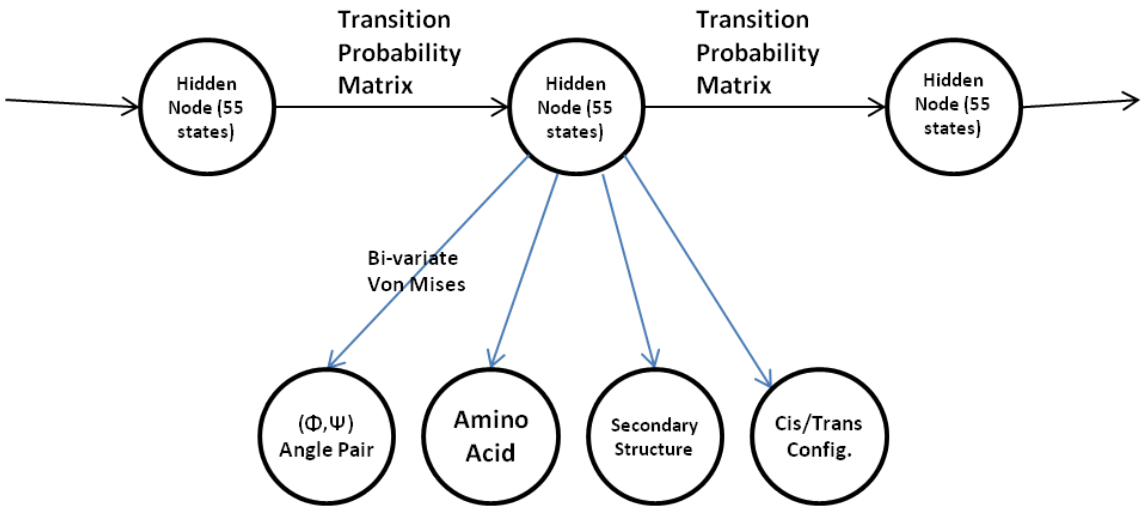

Figure 2.3: Generative model for Dynamic Bayesian Network with von Mises emissions

2.2.4 Gaussian Mixtures Graphical Model

Both of GGM and von-Mises graphical methods have a major shortcoming, which is the uni-modality of the marginals. In most applications, marginal distributions have several modes, and by fitting a unimodal distribution to these variables, the mode of the result often is far from either of the two modes. Figure 2.4 shows an example of the marginal of ϕ angle of fourth residue (Prolin4) in $Engrailed$ protein data, during an unfolding process. We see that a unimodal Gaussian distribution does not provide a reasonable estimation of the data.

Some groups have tackled this multi-modality by replacing unimodal factor functions with mixtures of Gaussian. In [83], Sudderth et. al. estimate each edge potential by mixture of Gaussians, and then performs Gaussian mixture belief propagation for inference. Belief propagation in these models becomes intractable, so they truncate the messages to have a limited number of mixture components. In practice the number of components is chosen through cross-validation.

We develop see in chapter 4 how we can estimate the parameters of a mixture of von Mises graphical model, which we hope is able to handle multi-modality of the angular data. We will describe how we used Expectation Maximization to train the model.

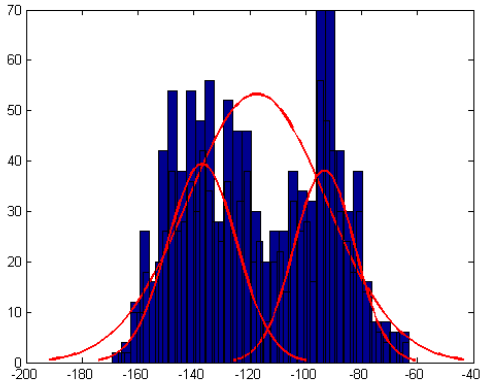

Figure 2.4: Histogram and Gaussian distribution fitting for Pro4 amino acid in Engrailed Protein, after fitting single and mixture of two Gaussians.

Chapter 3

Von-Mises Graphical Models: Sparse Structure Learning, Parameter Estimation, and Inference

3.1 Introduction

Von Mises distribution is a continuous probability distribution defined on a circle, and is used in directional statistics. In this chapter, we introduce the undirected von-Mises $graphical\ model$, which can be used to model a large set of angular variables. We present algorithms for performing parameter estimation and structure learning using L_1 regularization, and show that the learning algorithm is both consistent and efficient. We also introduce an efficient inference algorithm based on Nonparametric Belief Propagation and Expectation Propagation. We compare and contrast the von Mises Graphical Model (vMMs) with a Gaussian Graphical Model (GGM) on both synthetic data and on data from protein structure Molecular Dynamic(MD) simulations, and demonstrate that the vMM achieves higher accuracy than the GGM. We also show that the proposed inference algorithm converges faster than Gibbs sampling without hurting the performance.

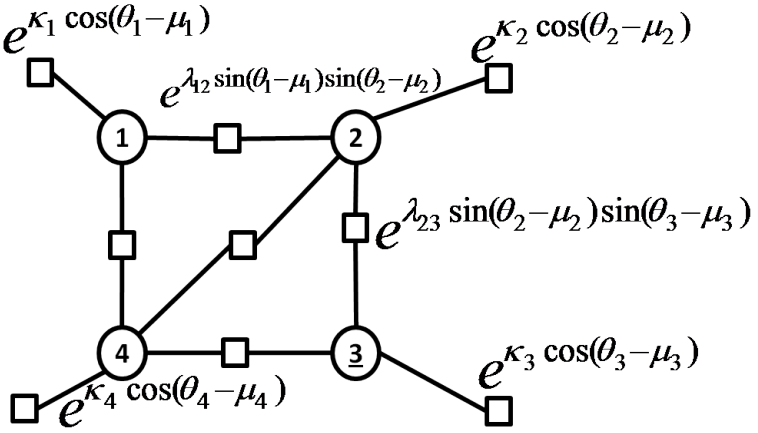

Figure 3.1: Factor Graph Representation for multivariate von Mises distribution. Each circular node is a variable, and the square nodes are factors.

3.2 The von Mises Graphical Model (vGM)

Let $\Theta = (\theta_1, \theta_2, ..., \theta_p)$, where $\theta_i \in [-\pi, \pi)$. The multivariate von Mises distribution [49] with parameters $\vec{\mu}, \vec{\kappa}$, and Λ is given by:

$$f(\Theta) = \frac{\exp\{\vec{\kappa}^T \vec{C} + \frac{1}{2}\vec{S}\Lambda\vec{S}^T\}}{Z(\vec{\mu}, \vec{\kappa}, \Lambda)},$$

where $\vec{\mu} = [\mu_1, \mu_2, \cdots, \mu_p]$, $\vec{\kappa} = [\kappa_1, \kappa_2, \cdots, \kappa_p]$, $\vec{C} = [\cos(\theta_1 - \mu_1), \cos(\theta_2 - \mu_2), \cdots, \cos(\theta_p - \mu_p)]$, $\vec{S} = [\sin(\theta_1 - \mu_1), \sin(\theta_2 - \mu_2), \cdots, \sin(\theta_p - \mu_p)]$, Λ is a $p \times p$ matrix such that $\Lambda_{ii} = 0$, $\Lambda_{ij} = \lambda_{ij} = \lambda_{ji}$, and $Z(\vec{\mu}, \vec{\kappa}, \Lambda)$ is the normalization constant.

It is known that the multivariate von Mises distribution can be closely approximated with a multivariate Gaussian distribution — provided that each of the variables has low variance (i.e., for large values of κ) [29]. This is significant because learning and inference can be performed analytically for multivariate Gaussian distributions. However, we will show in Section 3.7 that the Gaussian approximation introduces significant error when the variance is high (i.e., for small values of κ_i). We address this problem by encoding the multivariate von Mises distribution as a graphical model over von Mises-distributed random variables. Figure 3.1 shows the factor graph representation of the graphical mode for four variables. Under this representation the node

factors are defined as $f_i = \kappa_i cos(\theta_i - \mu_i)$ and the edge factors are defined as $f_{ij} = \lambda_{ij} sin(\theta_i - \mu_i) sin(\theta_j - \mu_j)$. Like all factor graphs, the model encodes the joint distribution as the normalized product of all factors:

$$P(\Theta = \theta) = \frac{1}{Z} \prod_{a \in A} f_a(\theta_{ne(a)}),$$

where A is the set of factors and $\theta_{ne(a)}$ are the neighbors of f_a (factor a) in the factor graph.

3.3 Sampling in vGM

The evaluation of the joint von Mises distribution requires the calculations of the normalization constant, Z. Unfortunately, Z does not have a closed form solution and must therefore be calculated by inference. The easiest way to perform inference is via Gibbs sampling, which has been done in [29].

Univariate von-Mises conditionals are univariate von Mises distributions themselves, and this makes Gibbs sampling mathematically easy. In particular

$$f(\theta_p | \theta_1, \theta_2, \dots \theta_{p-1}) \propto \exp\{\kappa_p cos(\theta_p - \mu_p) + \sum_{j=1}^{p-1} \lambda_{jp} sin(\theta_j - \mu_j) sin(\theta_p - \mu_p)\}$$

$$= \exp\{\kappa^* cos(\theta_p - \mu^*)\},$$

where

$$\kappa^* = \sqrt{\kappa_p^2 + (\sum_{j=1}^{p-1} \lambda_{jp} sin(\theta_j - \mu_j))^2} \tag{3.1}$$

$$\mu^* = \mu_p + \arctan(\frac{1}{\kappa_p} \sum_{j=1}^{p-1} \lambda_{jp} sin(\theta_j - \mu_j)) \tag{3.2}$$

This univariate conditional is sufficient for implementing a Gibbs sampler to generate samples from the vGM and perform inference.

3.4 Sparse Structure Learning and Parameter Estimation in vGM

We next consider the problem of learning the parameters of the model from data. Let $(\vec{\mu}, \vec{\kappa}, \Lambda)$ be the parameters of the vGM, as defined in Section 3.2. Given a set of $i.i.d.$ training samples, $D = \{\Theta_1, \Theta_2, \cdots \Theta_n\}$, the likelihood function is:

$$\mathrm{L}(D|\vec{\mu}, \vec{\kappa}, \Lambda) = \prod_{i=1}^{n} \frac{e^{\vec{\kappa}\vec{C}_i(\Theta,\vec{\mu}) + \frac{1}{2}\vec{S}_i(\Theta,\vec{\mu})^T \Lambda \vec{S}_i(\Theta,\vec{\mu})}}{Z_p(\vec{\mu}, \vec{\kappa}, \Lambda)}$$

where $\vec{C}(\Theta_i, \vec{\mu}) = [\cos(\theta_{i,1} - \mu_1), \cdots \cos(\theta_{i,n} - \mu_p)]$, and $\vec{S}(\Theta_i, \vec{\mu}) = [\sin(\theta_{i,1} - \mu_1), \cdots \sin(\theta_{i,n} - \mu_p)]$. In theory, a maximum likelihood estimate MLE for the parameters can be obtained by maximizing the likelihood of the data. Unfortunately, computing the normalization constant is NP-hard, so computing a MLE estimate for the vGM is intractable. We will therefore maximize the full $pseudo$-likelihood instead.

3.4.1 Full pseudo-likelihood for von Mises Graphical Model

The full pseudo likelihood for the multivariate von Mises is defined as follows:

$$\mathrm{PL}(\Theta|\vec{\mu}, \vec{\kappa}, \Lambda) = (2\pi)^{-pn} \prod_{i=1}^{n} \prod_{j=1}^{p} P_{vm}(\theta_{i,j}|\theta_{i,1}, ..., \theta_{i,j-1}, \theta_{i,j+1}...\theta_{i,p})$$

As discussed in section 3.3, each univariate conditional term for the vGM is itself a univariate

von Mises distribution. Thus, the full pseudo likelihood can be re-written as:

$$\text{PL}(\Theta|\vec{\mu},\vec{\kappa},\Lambda) = (2\pi)^{-pn}\prod_{j=1}^{p}\prod_{i=1}^{n}[I_0(\kappa_{\backslash\!\!\rfloor}^{(i)})]^{-1}e^{\kappa_{\backslash\!\!\rfloor}^{(i)}cos(\theta_{i,j}-\mu_{\backslash\!\!\rfloor}^{(i)})},$$

where

$$\mu_{\backslash\!\!\rfloor}^{(i)} = \mu_j + tan^{-1}\left(\frac{\sum_{l\neq j}\lambda_{j,l}sin(\theta_{i,l}-\mu_l)}{\kappa_j}\right), \text{and} \kappa_{\backslash\!\!\rfloor}^{(i)} = \sqrt{\kappa_j^2 + (\sum_{l\neq j}\lambda_{j,l}sin(\theta_{i,l}-\mu_l))^2}.$$

3.4.2 Consistency of the pseudo likelihood estimator

Dillon and Lebanon show that a maximum pseudo likelihood estimator is consistent provided that the mapping between conditional probabilities and joint probability is $injective$, i.e. the joint probability can be uniquely specified by the set of conditionals [19]. This property does hold true for von Mises.

Proof: Consider two conditionals with different parameters ($\vec{\kappa}_1^*$ and $\vec{\kappa}_2^*$, and $\vec{\mu}_1^*$ and $\vec{\mu}_2^*$), which have the same conditional distributions.

$$[I_0(\kappa_1^*)]^{-1}e^{\kappa_1^*cos(\theta-\mu_1^*)} = [I_0(\kappa_2^*)]^{-1}e^{\kappa_2^*cos(\theta-\mu_2^*)}$$

By taking the derivative of the two conditionals based on θ, and equating the two derivatives, and setting those equal, we get the system of equations:

$$\kappa_1^*cos(\theta-\mu_1^*) = \kappa_2^*cos(\theta-\mu_2^*)$$

$$\kappa_1^*sin(\theta-\mu_1^*) = \kappa_2^*sin(\theta-\mu_2^*)$$

From which we conclude κ_1^*=κ_2^*, and μ_1^*=μ_2^*.

So far we have shown that the conditional probability equality results in equality of the hyper parameters, κ^*s and μ^*s. These parameters are defined in equation (1) and (2), so now we have to

show individual parameters are equal as well. (i.e. For each i and j, $\kappa_{1i} = \kappa_{2i}$ and $\lambda_{1ij} = \lambda_{2ij}$.)

Because the equalities κ_1^*=κ_2^* are held true for any θ value, we can set $\theta_i = \mu_i^*$ in equation (1). This decision eliminates the Sin term, and directly results in $\kappa_{1i}^2 = \kappa_{2i}^2$. And since κ is positive by definition, we conclude that for all i, $\kappa_{1i} = \kappa_{2i}$.

On the other hand, we can also set $\theta_i = \mu_i^* + \frac{\pi}{2}$ in equation (2), which results in the following system of equations. For all i and j,

$$\sum_{l \neq j} \lambda_{1jl} = \sum_{l' \neq j} \lambda_{2jl'}$$

This system has only one solution, which is, for all i and j, $\lambda_{1ij} = \lambda_{2ij}$. And with this conclusion, we have shown that knowing the conditional distributions for von Mises is enough to specify the whole probability distribution, and consequently, the theorem discussed in [19] proves that the Full Pseudo Likelihood is a $consistent$ estimator for the vGM.

3.4.3 Structure learning for vGM

When the topology of the graph is not given or known, we must also learn the structure of the model, as well as the parameters. The study of the so-called structure learning problem has received considerable attention recently (e.g.[39],[30], [73],[87]). Structure learning algorithms based on L_1 regularization are particularly interesting because they exhibit consistency and high statistical efficiency (see [85] for a review). We use an algorithm introduced by Schmidt et.al. [73] that solves the L_1-regularized maximum likelihood estimation optimization problem using gradient projection. Their algorithm can be applied to any twice-differentiable continuous loss function, without any specific functional forms assumed. In particular, for $x = (x_1, x_2, ..., x_n)$

and loss function L, their algorithm minimizes functions of the form:

$$min_x f(x) \equiv L(x) + \rho\|x\|_1$$
$$\text{where } \|x\|_1 = \sum_{i=1}^{n} |x_i|$$

Here, ρ corresponds to regularization parameter. The L_1-Projection method reformulates this problem as a constrained optimization problem. Schmidt et. al. [73] rewrite the absolute value as a differentiable function:

$$|x| \approx \frac{1}{\alpha}[log(1 + e^{-\alpha x}) + log(1 + e^{\alpha x})]$$

As α goes to infinity, the approximation error goes to zero.

If the objective function is differentiable, the whole L1 regularized term can be optimized using projected gradients. We note that methods based on projected gradients are guaranteed to converge to a stationary point [11].

We use this method to learn the structure and parameters of the vGM . We define the loss function L as the negative log of full pseudo likelihood, as defined in Section 3.4.1:

$$L(\Theta|\vec{\mu}, \vec{\kappa}, \Lambda) = -log(\text{PL}(\Theta|\vec{\mu}, \vec{\kappa}, \Lambda))$$

$$log(\text{PL}(\Theta|\vec{\mu}, \vec{\kappa}, \Lambda)) = -(np)log(2\pi) + \sum_{j=1}^{p}\sum_{i=1}^{n} -log(I_0(\kappa_{\backslash j}^{(i)})) + \kappa_{\backslash j}^{(i)} cos(\theta_{i,j} - \mu_{\backslash j}^{(i)}).$$

The sub-gradients of the loss function are calculated as follows. For each element of $\vec{\kappa}$, κ_R

we have:

$$\frac{\partial log(\text{PL}(\Theta|\vec{\mu},\vec{\kappa},\Lambda))}{\partial \kappa_R} = \kappa_R \sum_{i=1}^{n} \left(\frac{cos(\theta_{i,R} - \mu_{\setminus R}^{(i)}) - A_0(\kappa_{\setminus R}^{(i)})}{\kappa_{\setminus R}^{(i)}} + \frac{sin(\theta_{i,R} - \mu_{\setminus R}^{(i)}) \sum_{l \neq R} \lambda_{Rl} sin(\theta_{il} - \mu_l)}{\kappa_{\setminus R}^{(i)}} \right)$$

Here, $A_0(\kappa)$ is defined as $\frac{I_1(\kappa)}{I_0(\kappa)}$ as described in [49].

Taking derivative of the pseudo likelihood with respect to each element of Λ matrix, $\lambda_{R,S}$, is also as follows:

$$\frac{\partial log(\text{PL}(\Theta|\vec{\mu},\vec{\kappa},\Lambda))}{\partial \lambda_{R,S}} = \sum_{j=1}^{p} \sum_{i=1}^{n} \left(\frac{\partial \kappa_{\setminus j}^{(i)}}{\partial \lambda_{R,S}} [-A_0(\kappa_{\setminus j}^{(i)}) + cos(\theta_{i,j} - \mu_{\setminus j}^{(i)}] + \frac{\partial \mu_{\setminus j}^{(i)}}{\partial \lambda_{R,S}} \kappa_{\setminus j}^{(i)} sin(\theta_{i,j} - \mu_{\setminus j}^{(i)}) \right)$$

such that

$$\frac{\partial \kappa_{\setminus j}^{(i)}}{\partial \lambda_{R,S}} = \delta(R,J) * \frac{\sum_{l \neq j} \lambda_{j,l} sin(\theta_{i,l} - \mu l) * sin(\theta_{i,s} - \mu_s)}{\kappa_{\setminus j}^{(i)}}$$

$$\frac{\partial \mu_{\setminus j}^{(i)}}{\partial \lambda_{R,S}} = \delta(R,J) * \frac{sin(\theta_{i,s} - \mu_s)}{\kappa_j * (1 + [\frac{\sum_{l \neq j} \lambda_{j,l} * sin(\theta_{i,l} - \mu l)}{\kappa_j}]^2)}$$

These gradients are then used in the projected gradient method to solve the maximum pseudo likelihood estimation for the parameters of the von Mises graphical model.

3.5 Inference in von Mises Graphical Models

One of the most widely used Inference techniques on Graphical Models is Belief Propagation [63]. Belief Propagation algorithm works by passing real valued functions called $messages$ along the edges between the nodes. A message from node v to node u contains the probability factor which results from eliminating all other variables up to u, to variable u. There are two types of messages:

A message from a variable node "v" to a factor node "u" is the product of the messages from

all other neighboring factor nodes.

$$\mu_{v\to u}(x_u) = \prod_{u^*\in N(v)\setminus\{u\}} \mu_{u^*\to v}(x_v)$$

where $N(v)$ is the set of neighboring (factor) nodes to v.

A message from a factor node u to a variable node v is the product of the factor with messages from all other nodes, marginalized over x_v.

$$\mu_{u\to v}(x_v) = \int_{\mathbf{x}'_u:x'_v=x_v} f_u(\mathbf{x}'_u) \prod_{v^*\in N(u)\setminus\{v\}} \mu_{v^*\to u}(x'_u)dx'_u$$

where $N(u)$ is the set of neighboring (variable) nodes to u.

In von Mises graphical models, the integration for computing the message from a bivariate factor to a node does not have a closed form solution, and Belief propagation algorithm can not be represented by a single set of μ and κ and λ anymore, so we can not perform belief propagation on these graphs directly. To solve this problem, we follow the solution provided by Sudderth et.al.[83], in which the outgoing message is approximated to fall in a certain family. Our proposed method differs from theirs such that we approximate the outgoing message to be in von Mises family, rather than Gaussian. Our method is also similar to Expectation Propagation, developed by Minka[56], but the distiction is that while in Expectation Propagation the approximation is based on moment matching on the posterior, our algorithm performs this via moment matching of the messages themselves.

In our model, the graphical model marginal probabilities, $P(x_i)$ are the product of the factors, and can be calculated by the messages passed to the variable node v_i.. In the next section we will perform the moment matching step (i.e. The KL-divergence minimization step) for the special case of the Von Mises model.

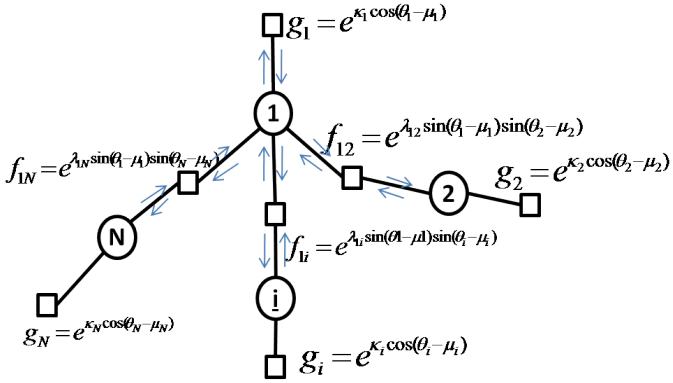

Figure 3.2: Von Mises Factor graph representation, and the messages exchanged between the factors and variable x_1.

3.6 Message Expectation propagation for von Mises Graphical Models

Recall from section 3.2, that a von Mises distribution is factorized as uni and bivariate factors:

$$P(x|\mu, \kappa, \lambda) = \frac{1}{Z(\kappa, \lambda)} \prod_{i=1}^{n} e^{\kappa_i \cos(x_i - \mu_i)} \prod_{i,j=1 i \neq j}^{n} e^{\lambda_{ij} \sin(x_i - \mu_i) \sin(x_j - \mu_j)}$$

Figure 3.2 shows the factor graph representation of Von Mises and the messages that pass between variable x_1 and the related factors.

In approximated belief propagation for von Mises, each message going to the variable will be approximated by a $univariate$ Von Mises and we used moment matching to calculate the messages. To perform the moment matching there are four types of messages. Message from factor node g_i to the variable x_i, and vice versa; And message from factor node f_ij to the variable x_i.

Messages from factor G to the variables - Factors Gi are univariate factors with $e^{\kappa_i cos(x_i - \mu_i)}$ as their value. The message they will send to the variable x_i is simply $e^{\kappa_i cos(x_i - \mu_i)}$, already in the desired form.

Messages from factor F to the variables - Factors F_{ij} are bivariate, and they first have to

receive message from x_j, then multiply the factor of the form $e^{\lambda_{ij} sin(x_i-\mu_i)sin(x_j-\mu_j)}$, and approximate the integration, by a message of the form $e^{\kappa^* cos(x_i-\mu^*)}$.

Let's assume that the message $m_{j\to F_{ij}} = e^{\kappa_j^* cos(x_j-\mu_j^*)}$ is the message that x_j sends to the factor F_{ij}. The message that F_{ij} sends to the variable x_i is then calculated as follows:

$$m_{fij\to x_i}(x_i) = \int_{x_j} F_{ij}(x_i, x_j) m_{j\to F_{ij}}(x_j) dx_j = \int_{x_j} e^{\lambda_{ij} sin(x_i-\mu_i)sin(x_j-\mu_j)} e^{\kappa_j^* cos(x_j-\mu_j^*)} dx_j$$

We want to approximate this integral by the form:

$$m_{fij\to x_i}(x_i) \approx e^{\kappa^* cos(x_i-\mu^*)}$$

We will use moment matching technique to find κ^* and μ^* . However direct moment matching does not have a closed form solution in this case and is computationally very expensive. Instead, we use *Trigonometric Moments*, to perform moment matching and approximate the message. [52] defines two set of moments, $\alpha_0, \alpha_1, ...$ and $\beta_0, \beta_1, ...$, that are used to identify a function with desired level of accuracy. These moments are defined as follows.

$$\alpha_p(f_x) = \int_x \cos(px) f(x) dx$$

$$\beta_p(f_x) = \int_x \sin(px) f(x) dx$$

The series α and β specify the Fourier coefficients of a function, which uniquely specifies the function with desired accuracy. We match α_0 , α_1 , β_0 and β_1 of the two functions: The actual message that should be sent, $\int_{x_j} e^{\lambda_{ij} sin(x_i-\mu_i)sin(x_j-\mu_j)} e^{\kappa_j^* cos(x_j-\mu_j^*)} dx_j$, and the approximation of it, $e^{\kappa^* cos(x_i-\mu^*)}$.

For simplicity and without loss of generality, we can assume that the μ_i for data was first

calculated and subtracted from the data.

$$\alpha_0^{real} = \iint_{x_i,x_j=-\pi}^{\pi} e^{\lambda_{ij} sin(x_i) sin(x_j) + \kappa_j^* cos(x_j - \mu_j^*)} dx_j dx_i$$
$$\beta_0^{real} = 0.$$
$$\alpha_1^{real} = \iint_{x_i,x_j=-\pi}^{\pi} \cos(x_i) e^{\lambda_{ij} sin(x_i) sin(x_j) + \kappa_j^* cos(x_j - \mu_j^*)} dx_j dx_i$$
$$\beta_1^{real} = \iint_{x_i,x_j=-\pi}^{\pi} \sin(x_i) e^{\lambda_{ij} sin(x_i) sin(x_j) + \kappa_j^* cos(x_j - \mu_j^*)} dx_j dx_i$$

The trigonometric moments for $e^{\kappa^* cos(x_i - \mu^*)}$ are also calculated as follows:

$$\alpha_0^{ep} = \int_{x_i=-\pi}^{\pi} e^{\kappa^* cos(x_i - \mu^*)} dx_i = 2\pi I_0(\kappa^*)$$
$$\beta_0^{ep} = 0.$$

On the other hand, for the higher degree moments we have:

$$\alpha_1^{ep} = \int_{x_i-\pi}^{\pi} \cos(x_i) e^{\kappa^* \cos(x_i - \mu^*)} dx_i = \int_{x_i-\pi}^{\pi} \cos(x_i) e^{\kappa^* \cos(x_i)\cos(\mu^*) - \kappa^* sin(x_i) sin(\mu^*)} dx_i$$
$$\beta_1^{ep} = \int_{x_i-\pi}^{\pi} \sin(x_i) e^{\kappa^* cos(x_i - \mu^*)} dx_i = \int_{x_i-\pi}^{\pi} \sin(x_i) e^{\kappa^* \cos(x_i)\cos(\mu^*) - \kappa^* sin(x_i) sin(\mu^*)} dx_i$$

Each individual integral does not have a closed form on its own, and thus would not let us estimate the κ^* and μ^*. However we can combine the α_1^{ep} and β_1^{ep}s, and get the relationship between κ^*, μ^*, α_1^{real} and β_1^{real}:

$$\kappa^* \cos(\mu^*)\beta_1^{ep} + \kappa^* \sin(\mu^*)\alpha_1^{ep} = -e^{\kappa^* \cos(x_i - \mu^*)}\Big|_{x_i=-\pi}^{\pi} = 0$$

So, since after moment matching, $\alpha_p^{real} = \alpha_p^{ep}$ and $\beta_p^{real} = \beta_p^{ep}$, we can get the parameters of the

Expectation Propagation message as follows:

$$\kappa^* = \frac{I_0^{-1}(\alpha_0^{real})}{2\pi} \;\; and \;\; \mu^* = -tan^{-1}(\frac{\beta_1^{real}}{\alpha_1^{real}})$$

Messages from variable x_i to the factors - Messages sent from factors are all approximated to be of the form $e^{\kappa_j^* cos(x_i - \mu_j^*)}$. The messages that variable sends to the factors are also of the same family, however they are exact. A message from variable x_i to factor G_i is a product of all messages received excluding the message received from factor G_i:

$$m_{i \to G_i}(x_i) = \prod_{j=1..N, j \neq i} m_{f_{ij} \to i}(x_i) = \prod_{j=1..N, j \neq i} e^{\kappa_j^* \cos(x_i - \mu_j^*)} = e^{\kappa_i^G \cos(x_i - \mu_i^G)}$$

So the message parameters κ_i^G and μ_i^G can be calculated as follows:

$$\kappa_i^G = \sqrt{(\sum_{l \neq i} \kappa_l^* \cos(\mu_l^*))^2 + (\sum_{l \neq i} \kappa_l^* \sin(\mu_l^*))^2} \; ; \mu_i^G = \tan^{-1} \frac{\sum_{l \neq i} \kappa_l^* \sin(\mu_l^*)}{\sum_{l \neq i} \kappa_l^* \cos(\mu_l^*)}$$

Similarly, messages from variable x_i to factor F_{ij} can be calculated as:

$$m_{i \to F_{ij}}(x_i) = e^{\kappa_{ij}^F \cos(x_i - \mu_{ij}^G)}$$

Such that:

$$\kappa_{ij}^F = \sqrt{(\sum_{l \neq j} \kappa_l^* \cos(\mu_l^*))^2 + (\sum_{l \neq j} \kappa_l^* \sin(\mu_l^*))^2} \; ; \mu_{ij}^F = \tan^{-1} \frac{\sum_{l \neq j} \kappa_l^* \sin(\mu_l^*)}{\sum_{l \neq j} \kappa_l^* \cos(\mu_l^*)}$$

Messages from observed variables to their factors - If we have any observed variables, we can simply replace the observation in the edge factors and treat them like a node potential, and continue inference.

Partition Function after Convergence - Once the messages passing has converged, we can

calculate the final partition function as the integration of beliefs of any of the variables.

$$Z = \int_{x_i} \prod_{j=1..N} e^{\kappa_j^* \cos(x_i - \mu_j^*)} dx_i = \int_{x_i} e^{\sum_{j=1..N} \kappa_j^* \cos(x_i - \mu_j^*)} dx_i = \int_{x_i} e^{\kappa_z^* \cos(x_i - \mu_z^*)} dx_i$$

$$where$$

$$\kappa_z^* = \sqrt{(\sum_{l=1..N} \kappa_l^* \cos(\mu_l^*))^2 + (\sum_{l=1..N} \kappa_l^* \sin(\mu_l^*))^2}$$

$$\mu_z^* = \tan^{-1} \frac{\sum_{l=1..N} \kappa_l^* \sin(\mu_l^*)}{\sum_{l=1..N} \kappa_l^* \cos(\mu_l^*)}$$

So finally, the partition function can be calculated as $Z = \frac{1}{2\pi} I_0(\kappa_z^*)$.

3.7 Experiments

In this section, we will present the results of our experiments, related to structure and parameter learning, and inference. Our experiments are performed on synthetic data, generated via Gibbs sampling, and also on Engrailed protein molecular dynamics simulation data. We compare our model to Gaussian graphical models(GGM) and we use Root Mean Squared Error (RMSE) as our evaluation metric.

3.7.1 Parameter Learning and Inference on Synthetic Data

We generated random vGM graphs for different parameter configurations by systematically varying the followings: (a) the number of nodes of graph; (b) the density of edges of the graph; and (c) the von Mises parameters $\vec{\kappa}$ and Λ. We generated $\vec{\kappa}$ using a uniform distribution on $U[0, S_\kappa]$. Elements of the Λ matrix were drawn from a Gaussian distribution $\mathcal{N}(0, S_\Lambda)$. In these synthetic datasets, the mean values for the marginal distributions, $\vec{\mu}$, were held fixed at zero.

We then used our Gibbs sampler (Sec. 3.3) to randomly generate 10 data set from each of the

randomly generated VGM configurations. Each dataset contained 1000 fully observed samples.

Next, we used our structure learning algorithm (Sec3.4) to learn a VGM from each data set. For comparison, we also used the structure learning algorithm presented in [72] to learn a sparse GGM from the same data.

Cross Validation and Evaluation Metric In each experiment, we performed leave-one-out cross validation, where for each test data, we assumed 50% of the variables to be unobserved, and performed inference to infer the values of the other 50%, given the observations and the model learned from the training data. We repeated the experiment for 10 different random 50% subsets, each time.

After the inference, we computed the RMSE (root mean squared error) between the predicted values and the true values.

Model Selection The structure learning algorithm has one free parameter — the regularization penalty for adding edges. We selected the optimal value for this parameter by first randomly shuffling each column of the samples (columns correspond to variables), to remove all effects of correlation between the variables. Then we learned a VGM for many values of regularization penalty on this shuffled data, and selected the lowest penalty that did not capture any dependencies on the data. This regularization penalty was then used on the actual samples for the learning.

Results Figures 3.3 and 3.4 present the RMSE of the estimated Von Mises graphical models, after the inference via our approximate inference, for two different graph sizes. In each figure, we show the effect of κ values, and λ values on RMSE, under different edge densities. Based on the results, we see as expected, that when the strength of the coupling between variables is small, (i.e. λs are drawn from $N(0, 0.1)$), density of the edges does not has much effect on the accuracy of the model. As the couplings get stronger, the accuracy of the estimations increases

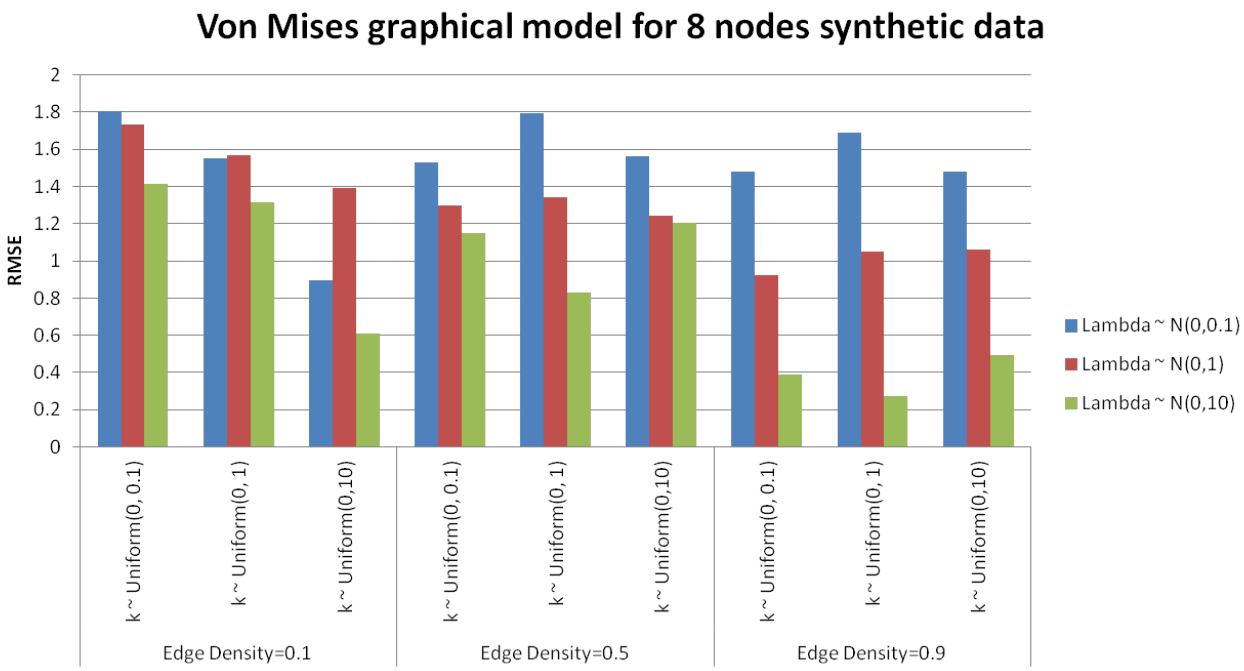

Figure 3.3: RMSE of estimated von-Mises graphical models on synthetic data with 8 nodes

in almost all cases.

As shown in the plot, when λs are drawn from $N(0, 10)$, RMSE is significantly lower in all cases. The reason for that is, higher lambda values correspond to stronger coupling between variables, and as the couplings get stronger it imposes tighter constraint on the variables, which in turn makes the variance of the samples smaller and the estimation of the parameters becomes easier.

When the couplings are strong, if the variable concentrations κs, are large, then we expect to see better accuracies because the entropy of the model will be lower. Indeed we observe this in our results. Also, the more connections we have, (i.e. higher edge density), the stronger we expect the predictive power to be, due to lower entropy. We can see that indeed, with denser graphs, regardless of the values of κs, we observe low RMSEs.

Figures 3.5 and 3.6 show the comparison of Von-Mises graphical model to sparse Gaussian graphical models on the synthetic data of sizes 8 and 16, for different configurations. We observe

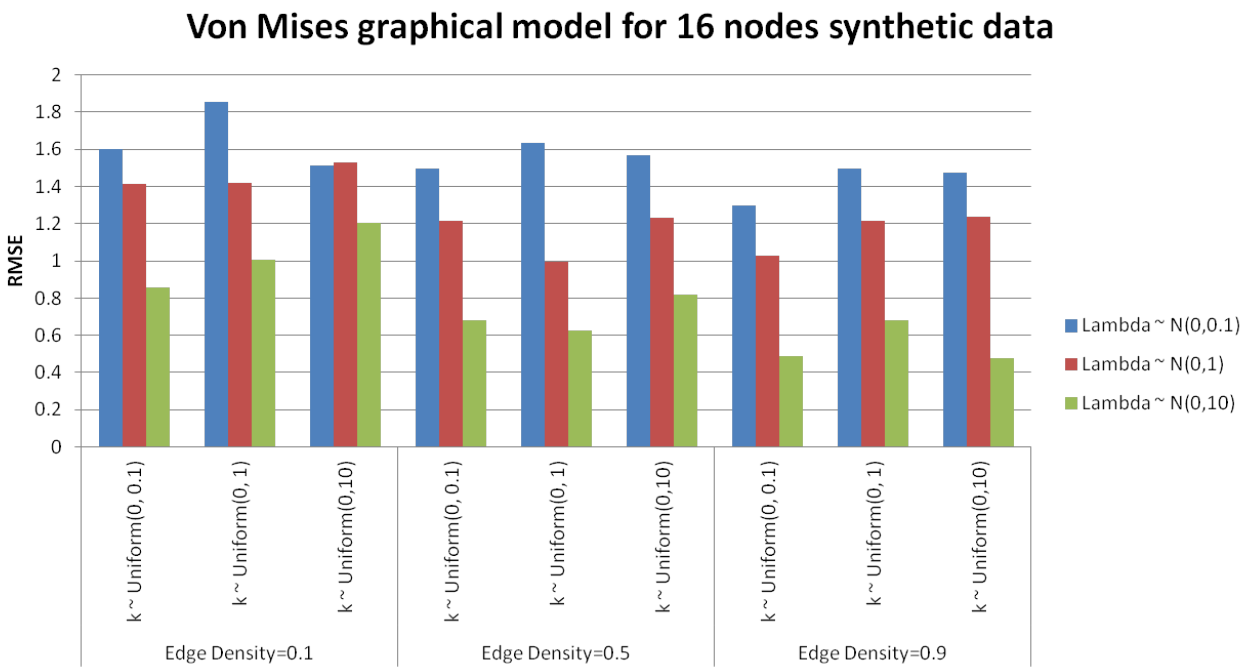

Figure 3.4: RMSE of estimated von-Mises graphical models on synthetic data with 16 nodes

that when κs are large (i.e. $\kappa \sim Uniform(0, 10)$), vGM and GGM perform similarly in most cases. However, when the concentrations are lower, we see larger improvements from using vGM graphical models. This result is expected because as previously mentioned (Sec. 3.2), a vGM can be well-approximated with a GGM when the variables have low variance (i.e., high κ). Below we describe how exactly this approximation is done via Taylor expansion.

Using the definition of the multivariate von Mises model:

$$f_{VMM}(\vec{\mu}, \vec{\kappa}, \Lambda) \propto \exp\{\vec{\kappa}^T \vec{C} + \frac{1}{2}\vec{S}\Lambda\vec{S}^T\}$$

where $\vec{C} = [\cos(\theta_1 - \mu_1), \cos(\theta_2 - \mu_2), \cdots, \cos(\theta_p - \mu_p)]$ and $\vec{S} = [\sin(\theta_1 - \mu_1), \sin(\theta_2 - \mu_2), \cdots, \sin(\theta_p - \mu_p)]$, we can use the Taylor expansion for $\cos(x-\mu)$ and $\sin(x-\mu)$ as follows:

$$\cos(x - \mu) = \sum_{n=0}^{\inf} \frac{(-1)^n}{(2n)!}(x - \mu)^{2n}$$

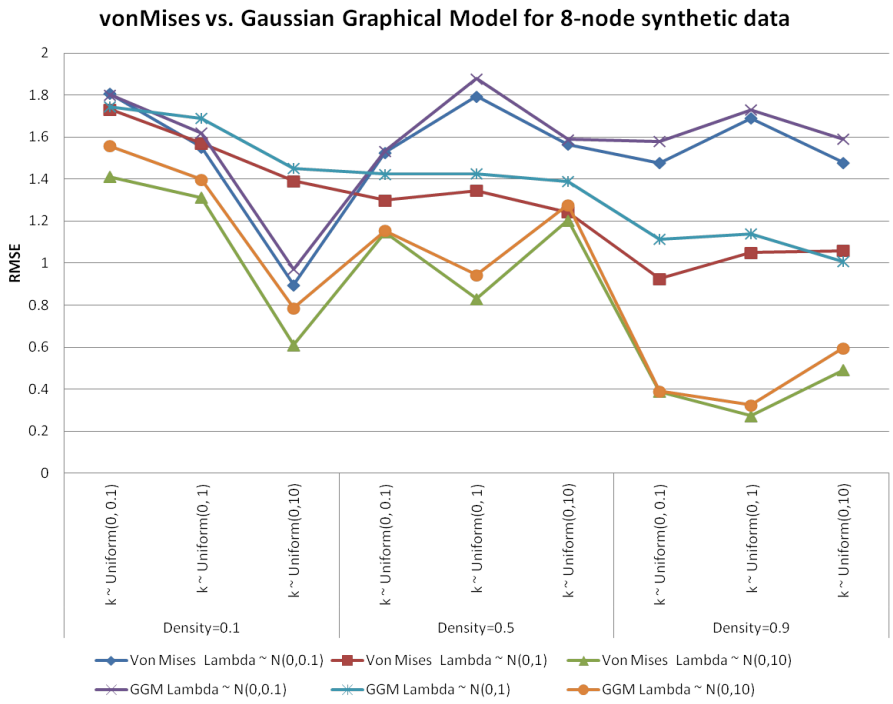

Figure 3.5: Comparing von-Mises graphical model with Gaussian graphical model on synthetic data with 8 nodes

$$\sin(x-\mu) = \sum_{n=0}^{\inf} \frac{(-1)^n}{(2n+1)!}(x-\mu)^{2n+1}$$

When $(x-\mu)$ is close to zero, these series can be approximated with:

$$\cos(x-\mu) \propto 1 - \frac{(x-\mu)^2}{2}$$

$$\sin(x-\mu) \propto x-\mu$$

Thus, under the condition where $(x-\mu)$ approaches zero (i.e., when the marginal variance of each variable is sufficiently small), a vGM can be approximated with a multivariate Gaussian distribution:

$$f_{VMM}(\vec{\mu}, \vec{\kappa}, \Lambda) \propto f_{GGM}(\mu, \Sigma)$$

where $(\Sigma^{-1})_{ii} = \kappa_i$ and $(\Sigma^{-1})_{ij} = -\Lambda_{ij}$.

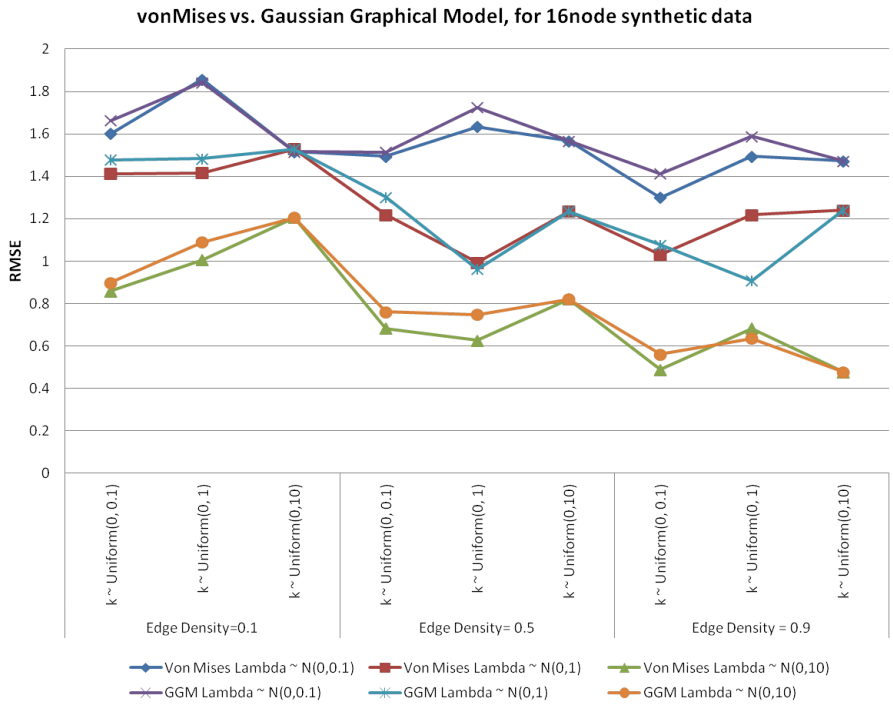

Figure 3.6: Comparing von-Mises graphical model with Gaussian graphical model on synthetic data with 16 nodes

Tables 3.1 and 3.2 show the $time$ and $RMSE$ of our proposed inference method, compared to Gibbs sampling, for different graph size with fixed $\kappa = 0.1$,$\lambda = 0.1$ and $density = 0.5$.

Time	8 nodes	16 nodes	32 nodes	64 nodes
Gibbs	11.21	43.78	186.81	826.16
VM Approximate Belief Propagation	0.34	0.37	0.79	1.34

Table 3.1: Comparing computational time(seconds) for Gibbs vs. von Mises approximate belief propagation inference for different nodes

As we observe, our proposed inference method outperforms Gibbs sampling in terms of both speed and accuracy of predictions.

3.7.2 Parameter learning and Inference on Engrailed Protein data

In addition to synthetic dataset, we also performed our experiments over engrailed homeodomain (Protein ID: 1ENH) MD simulations data, which is a 54-residue DNA binding domain(Figure

RMSE	8 nodes	16 nodes	32 nodes	64 nodes
Gibbs	1.95	2.12	1.89	1.83
VM Approximate Belief Propagation	1.52	1.51	1.40	1.31

Table 3.2: Comparing RMSE(degrees) for Gibbs vs. von Mises approximate belief propagation inference for different nodes

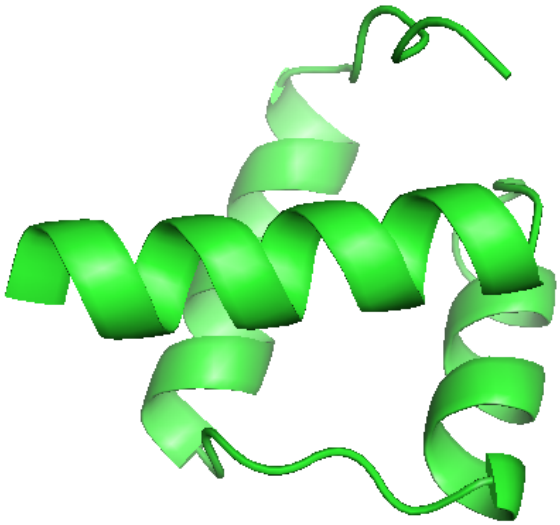

Figure 3.7: Engrailed Homeodomain

3.7. Homeodomains are the DNA-binding domains of homeotic proteins, which have a major role in the development of metazoans [26]. Certain mutations of the Homeodomains can be causes of diseases in humans[17]. Homeodomains fold into a highly conserved structure consisting of three alpha-helices, and the C-terminal helix makes sequence-specific contacts in the major groove of DNA [27]. The Engrailed Homeodomain is an ultra-fast folding protein that is predicted to exhibit significant amounts of helical structure in the denatured state ensemble [54]. Moreover, the experimentally determined unfolding rate is of 1.1E + 03/sec [53], which is also fast. Taken together, these observations suggest that the protein may exhibit substantial conformational fluctuations.

We performed three 50-microsecond simulations of the protein at 300, 330, and 350 degrees Kelvin. These simulations were performed on ANTON[74], a special-purpose supercomputer designed to perform long-timescale simulations. Each simulation had more than 500,000 frames.

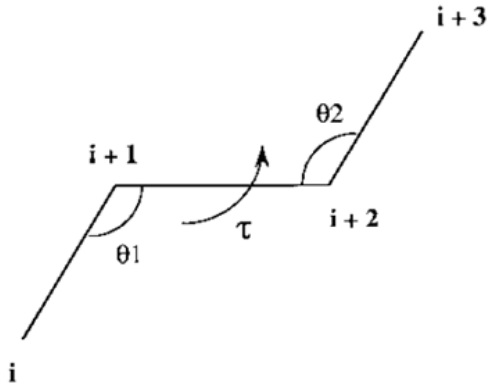

Figure 3.8: Theta and Tau angles representing a Protein Structure

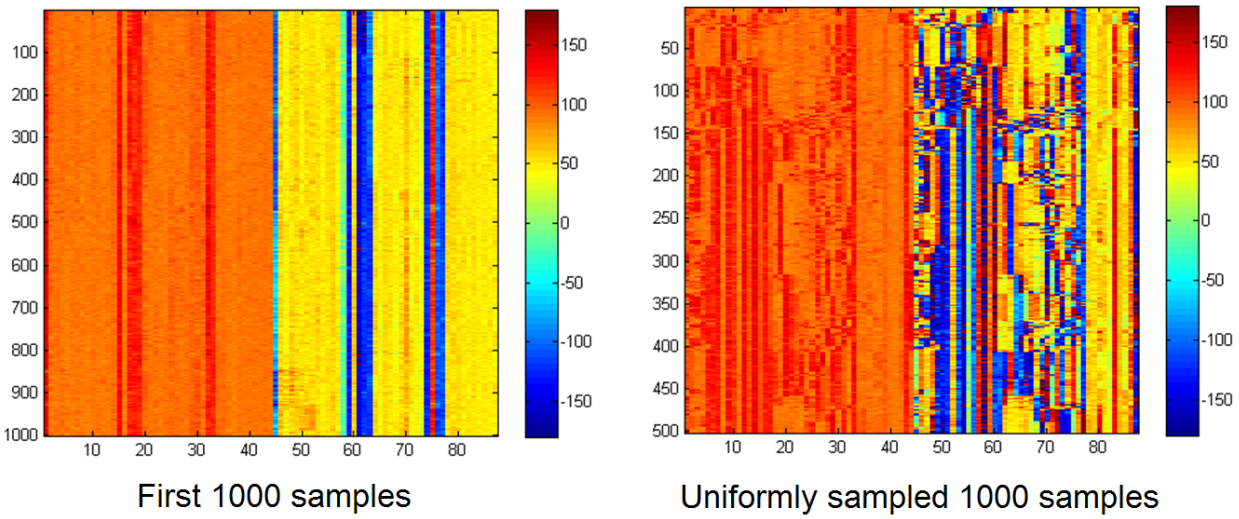

Figure 3.9: Engrailed protein structure frames

We use $angular$ sequence of (θ, τ) to represent each frame. θs are the bond angle of the $C - \alpha$ atoms sequentially, and τ is the dihedral angle specified between C_α atoms or i, $i + 1$, $i + 2$ and $i + 3$. Figure 3.8 shows these angles on a section of an imaginary protein sequence.

Figure 3.9 shows the two sub-sampled data that we have used: The first 1000 samples of the protein, and the uniformly sampled 1000 samples that cover the whole unfolding trajectory. The first data has lower entropy, and we expect lower prediction errors for the first sample set, compared to the uniformly sub-sampled data.

Table 3.3 shows the results of running the full cross validation experiment, on von-Mises

Data	von Mises graphical model	Gaussian graphical model	Wilcoxon rank test p-value
First 1000 samples	6.93	8.46	9.03e-20
Uniformly sampled 1000 samples	44.75	59.40	2.48e-17

Table 3.3: RMSE result comparison for von-Mises graphical models, vs. Gaussian graphical models

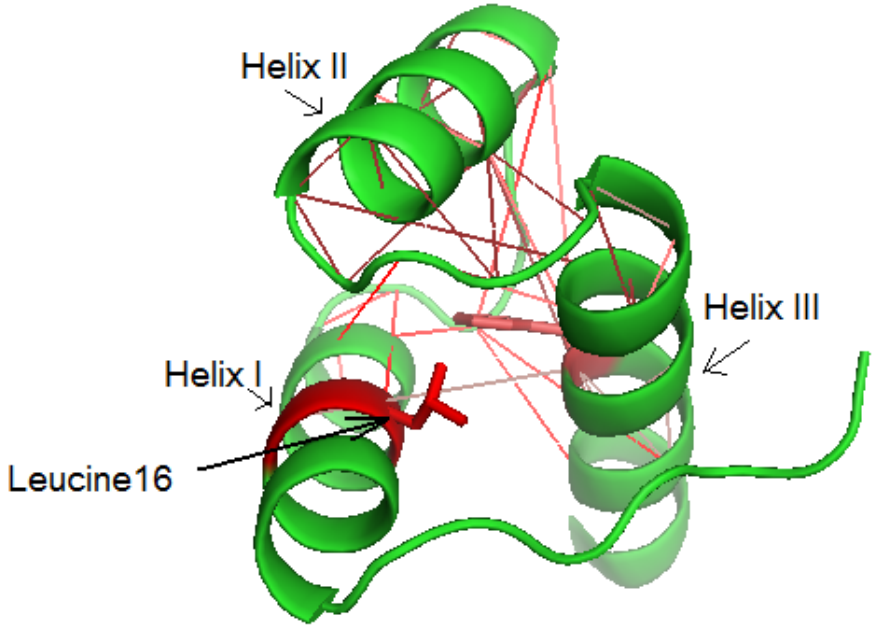

Figure 3.10: Structure Learned for Engrailed Homeodomain, via von Mises graphical models

graphical model and Gaussian graphical models. In both cases we see the RMSE (measured in degrees) of the predicted hidden variables conditioned on the observed variables, compared with their actual values. As we can see, the von Mises graphical model outperforms Gaussian graphical model based on out leave one out cross validation experiments.

Figure 3.10 shows the angular coupling over the whole data, learned via the von Mises structure learning model.

Mayor et. al. [53] showed that during the unfolding process, helix I and III show higher stability and we observe that most couplings are between the helix II and the loop region with

the rest of the structure. In another publication, Mayor et. al. [54] identified Leucine16, as one of the core hydrophobic residues which plays important part in the stabilization of the protein. As indicated in Figure 3.10, we also see that this residue couples strongly with Helix III region.

3.8 Summary

Von Mises Graphical models provide a unified framework to model a network of circular variables, but due to previously unsolved theoretical challenges, imposed by the particular form of the probability formula, these models had not yet been used. In this chapter, we used a gradient based algorithm to estimate sparse structure and parameters of such graphical models from data, and we showed the consistency of the maximum full pseudo likelihood estimator for these graphical models.

We also presented a novel inference method, based on nonparametric belief propagation developed specifically for Von-Mises graphical models. We used special techniques, including trigonometric moment matching, to derive the estimated parameters of the messages, and demonstrated that our developed inference method outperforms Gibbs sampling inference, in terms speed of convergence and RMSE error.

We tested the quality of our estimator on a set of synthetic data created by the Von-Mises sampler, and compared our estimator to the regularized Gaussian Graphical Model estimator, and observed that the Von Mises model has a better accuracy compared to Gaussian Graphical Models across a fairly large range of parameter combinations. We also applied our model to the dihedral angles of the engrailed homeodomain. Comparing the conditional probabilities of subsets of variables conditioned on the rest, showed us that Von Mises is a better fit for the protein data, and can recover long distance dependencies between the movements of residues.

However, real world applications of these graphical models are typically of multi-modal nature, while presented von Mises graphical model is a unimodal system. In the next chapter, we will focus on one of our solutions to this issue: mixture of von Mises graphical models.

Chapter 4

Mixtures of von Mises Graphical Models

4.1 Introduction

While von Mises graphical models are capable of handling angular distributions, they assume that the marginal distributions of each variable is uni-modal. In practice however, it is often the case that the marginal distributions of variables are multi-modal. To solve this shortcoming, in this chapter we propose a Mixture of von-Mises graphical models, which can handle multi-modal angular distributions. We develop an estimation algorithm based on Expectation Maximization (EM) [18], and present our results on modeling side-chain angles of Argenine amino acid in a non-redundant set of protein structures.

4.2 Mixtures of von Mises graphical models

A mixture of von Mises with k component can be described as:

$$P(\theta|\mu_1, ..., \mu_k, \kappa_1, ..., \kappa_k, \Lambda_1, ..., \Lambda_k) = \sum_{k=1}^{K} \pi(k) P_{vm}(\theta|\mu_k, \kappa_k, \Lambda_k)$$

where $\pi(i)$ is the mixture weight (prior probability of selecting component i), and μ, κ and

Λ are the parameters of each von Mises component.

To estimate the parameters of the mixture components, we alternate between (M step) approximating the k von Mises parameters, and (E step) computing partial assignment (i.e. probability of each observation according to each von Mises component). For estimation of the parameters of each component, we follow the same process as maximum log pseudo-likelihood estimation, as we discussed in section 3.4.1.

The only modification is that we now have to incorporate the partial assignment (i.e. weight η_i for observation θ_i) of each observation in the formulation of the pseudo likelihood, for estimating each von Mises component, which is straight forward:

$$\text{Log PL}(\theta|\vec{\mu},\vec{\kappa},\Lambda) = (-pn)log((2\pi))\sum_{i=1}^{n}\eta_i\sum_{j=1}^{p}log(P_{vm}(\theta_{ij}|\theta_{i1},...,\theta_{i(j-1)},\theta_{i(j+1)}...\theta_{ip}))$$

Note that we optimize the L1-regularized log-pseudo-likelihood for each mixture component now, using the gradient descent algorithm we discussed in section 3.4.3. To compute the partial assignment in the E step, we use the inference algorithm we proposed in section 3.6.

In practical situations, it is often possible that the training data consists of millions of data points. In order to handle such large datasets, we develop our algorithm such that it can handle $weighted$ training data. Many algorithms, such as $coreset\ selection$[22] exist which can compresses the training data into weighted samples. The EM estimation method for mixture of von Mises developed in this chapter can easily handle such datasets.

Our algorithm is summarized in figure 4.1.

We note that our method for learning mixtures of vGMs is different than the method introduced by Boomsma *et al* [8]. Their method uses a functional approximation of the normalization factor during the maximum likelihood estimation and partial assignment stages in the EM algorithm. By approximating the von Mises as a Gaussian, the method can only model highly concentrated variables, which are not common in practice. In contrast, we perform consistent pseudo-likelihood estimation, and also, we compute the partial assignment of observation to the

Algorithm: Weighted EM for Mixture of Von Mises Graphical Models

Input: Samples D={$x_1,\ldots x_N$} with weights, W={$w_1,\ldots w_N$}, number of components K, Convergence threshold τ;

Output: $\mu_1,\ldots, \mu_K, \kappa_1,\ldots, \kappa_K, \Lambda_1,\ldots, \Lambda_K, \pi_1,\ldots, \pi_K$

Initialization: Sample $\mu_1,\ldots, \mu_k$, by sampling K points from D, according to weights W; Initialize $\kappa_1,\ldots, \kappa_K$ to be 1_{dx1}; Initialize $\Lambda_1,\ldots, \Lambda_K$ to be I_{dxd}; and initialize $\pi_1,\ldots, \pi_K$ to be equal to 1/K;

While $L_{vmm}(D|\ \mu_1,\ldots, \mu_K, \kappa_1,\ldots, \kappa_K, \Lambda_1,\ldots, \Lambda_K, \pi_1,\ldots, \pi_K) > L_{old} - \tau$ **do**

$L_{old} = L_{vmm}(D|\ \mu_1,\ldots, \mu_K, \kappa_1,\ldots, \kappa_K, \Lambda_1,\ldots, \Lambda_K, \pi_1,\ldots, \pi_K)$;

for all x_i in D , **for** k=1:K **do**

$$\eta_{ki} = w_i \frac{P_{vm}(x_i \mid \mu_k, \kappa_k, \Lambda_k)}{\sum_{l=1:K} P_{vm}(x_i \mid \mu_l, \kappa_l, \Lambda_l)}$$

end for

for k=1:K **do**

$$\pi_k = \frac{\sum_{i=1:N} \eta_{ki} w_i}{\sum_{i,j} \eta_{ij}}$$

$$\mu_k^*, \kappa_k^*, \Lambda_k^* = \arg\max_{\mu,\kappa,\Lambda} (-pn)\log(2\pi) \sum_{i=1}^{N} w_i \eta_{ki} \sum_{j=1}^{d} \log P_{vm}(x_{ij} \mid x_{i1},..x_{i(j-1)} x_{i(j+1)},...x_{id}, \mu, \kappa, \Lambda)$$

end for

end while

Figure 4.1: Weighted Expectation Maximization for learning mixture of von Mises graphical models.

mixture components directly using von mises graphical model framework we have developed.

4.3 Experiments

4.3.1 Dataset

We performed our experiments on modeling side-chain angles of Arginine amino acid. We collected the arginine data from Astral SCOP 1.75B dataset[13], which includes all sequences available in PDB SEQRES with less than 40% identity to each other.

We collected the PDB files for each sequence from RSCB data bank, and calculated torsion angles using MMTSB[21] software. Our dataset includes 7919 PDB structures, and is available o line at: http://www.cs.cmu.edu/ nsharifr/dataset/nrnpdb.tgz.

We then separated all instances of Arginine amino acid, and ended up with the dataset of size 93712 instances of the Arginine amino acid. Arginine amino acid is shown in figure 4.2. We collected a total of 7 variables for each instance: backbone dihedral angles ϕ,ψ,ω, and side-chain dihedral angles $\chi 1$ through $\chi 4$. We selected 5000 randomly selected subsamples as our final test data, which was not used during the development.

4.3.2 Experiment Setup and Evaluation

To select the number of mixing components(K), we used a development set of size 4000, randomly selected from the training data. We then computed the log likelihood of the development dataset, for different values of K, and picked the K that maximized the log likelihood of the development set.

We then performed $imputation$ test (i.e. predicting a subset of variables, conditioned on the model and the other variables as observed) using our unseen test dataset. In particular, we performed imputation for predicting the side chain angles, $\chi 1$ through $\chi 4$, given the backbone angles ϕ and ψ.

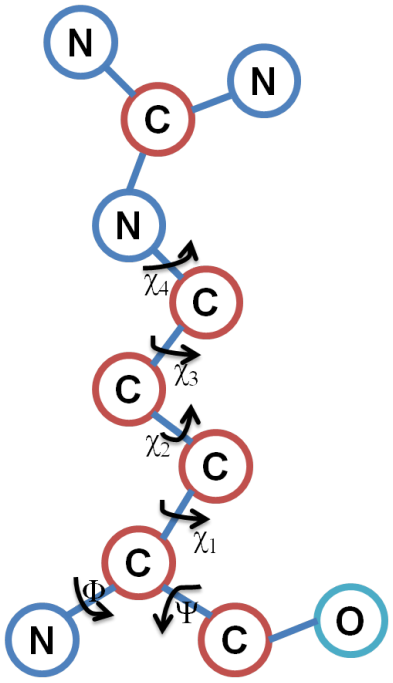

Figure 4.2: Arginine amino acid.

Our evaluation measures were log-likelihood, and Root Mean Squared Error(RMSE) of predicted angles compared to the true values. We note that to calculate the differences between values, we mapped each error to $[-\pi, \pi]$ to ensure fair comparison.

4.3.3 Results

Figure 4.3 shows the pairwise scatter plot of 7 dihedral angles of Arginine in our dataset. As it can be seen the data is clearly multi-modal, with high variance. Also the conditional mean of the angular values is often around $-\pi$ or π, which indicates that using a distribution which can take advantage of the equality of these two values in the unit circle may have a better predictions.

We first measured the log likelihood of the data under the two models and for different number of mixture components. Figures 4.4 and 4.5 show the negative log likelihood of the development set, for different values of K.

As it can be seen mixture of von Mises requires fewer components, since it can wrap the two ends of the unit circle. Also, the likelihood of the development set is significantly lower in

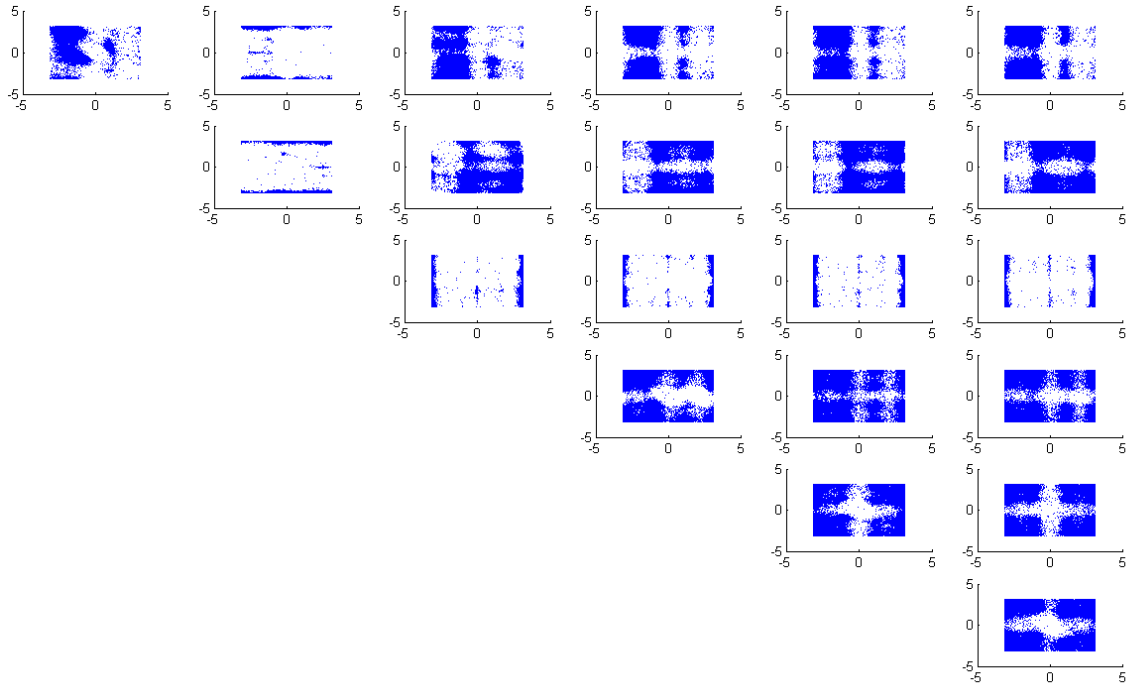

Figure 4.3: Scatter plot of dihedral angles of Arginine amino acid, in non-redundant PDB dataset

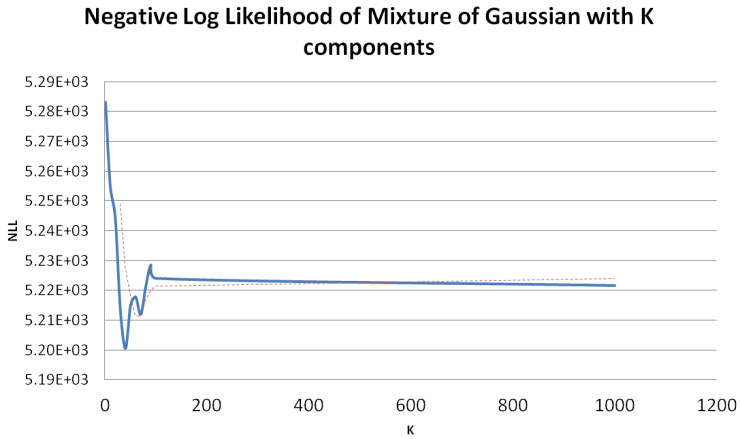

Figure 4.4: Negative Log Likelihood for Mixture of Gaussian model for different number of mixing components

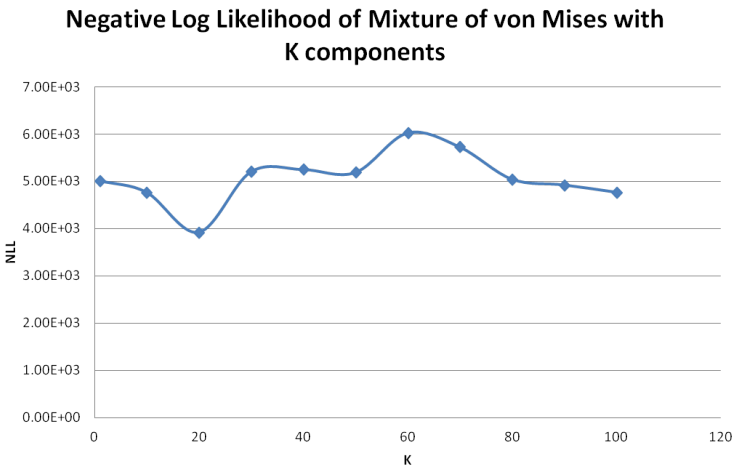

Figure 4.5: Negative Log Likelihood for Mixture of von Mises graphical model for different number of mixing components

mixture of von Mises model, compared to mixture of Gaussian. This indicates that the mixture of von Mises is a more appropriate model of angular data.

Table 4.1 shows the results of predicting the side-chain angles, conditioned on the backbone dihedral angles ϕ, and ψ. We measure the root mean squared error (RMSE) of the prediction. Mixture model results are reported for the number of mixing components(K) which achieved optimal log-likelihood during the cross validation.

As we can see, indeed von Mises model improves the prediction error significantly. Also notice that mixture of von Mises achieves the lower error with fewer parameters than mixture of Gaussian (20 components vs 40).

RMSE	Gaussian	Mixture of Gaussian (K=40)	von Mises	Mixture of von Mises (K= 20)
$\chi 1$	1.1999	0.9312	0.866	0.8130
$\chi 2$	1.1865	1.1048	0.9820	0.9115
$\chi 3$	1.3991	1.2259	1.0376	0.9837
$\chi 4$	1.4775	1.4683	0.9907	0.9815

Table 4.1: Accuracy of Gaussian, Mixture of Gaussian, von Mises, and Mixture of von Mises graphical models

Table 4.2 also shows the log likelihood of the test set, under the four models. As it can be

seen, the log likelihood for the unseen test data is significantly lower for mixture of von Mises graphical models. Although the better likelihood results come with the price of increased run-time. Table 4.3 shows the CPU time that it took for each model to train.

	Gaussian	Mixture of Gaussian (K=40)	von Mises	Mixture of von Mises (K= 20)
LL	-5.29e+03	-5.20e+03	-5.00e+03	-3.81e+03

Table 4.2: Log likelihood of Gaussian, Mixture of Gaussian, von Mises, and Mixture of von Mises graphical models, for unseen test set

	Gaussian	Mixture of Gaussian (K=40)	von Mises	Mixture of von Mises (K= 20)
Time	1.93	28.37	103.22	3732.91

Table 4.3: Run time(in seconds) of EM estimation algorithm for learning Gaussian, Mixture of Gaussian, von Mises, and Mixture of von Mises graphical models

4.4 Summary

In this chapter, we introduced mixture of von Mises graphical models, and developed a novel algorithm based on weighted expectation maximization, to estimate the parameters of the model. Our experiments over side chain prediction of Arginine amino acid showed that the von Mises mixture model outperforms both von Mises and also Gaussian and mixture of Gaussian graphical models, in terms of log likelihood of held out test data, and also the imputation error. The improvements, however, come with a price of orders of magnitude increase in runtime of training and inference steps, so depending on the resources available for training, one must choose the appropriate model.

Part II

Nonparametric Graphical Models

Chapter 5

Background and Related Work for Semi-parametric and Nonparametric graphical models

So far we focused on parametric graphical models. The benefits of these models is that they are compact and faster to estimate. However these benefits come at the cost of the model often being strictly designed for a specific set of distributions. In other words, if the variables of interest change, or if there are multiple types of variables, it is very difficult to use previous algorithms directly, and many times a whole new model needs to be defined. For instance, after designing a model for dihedral angles, if for a part of the calculations atomic coordinates become helpful, it is not possible to transform the algorithms easily to the new setting. Also, some applications such as protein design, involve sequence and structural variables, which have different distribution families. A model that can handle diverse set of variable types is very helpful in those important applications.

Semi-parametric and non-parametric models try to provide sample-based measures, which allow us to handle inhomogeneous variables sets, of arbitrary distributions, within the same model, with very little re-design of the general framework. These models use the data itself

as the source to calculate the required densities and expectations, without imposing a specific parametric form on the variables.

In the second part of this thesis, we focus on the semi-parametric and nonparametric graphical models, and this chapter reviews the background and related work in these families of graphical models.

5.1 Non-paranormal Graphical Models

Non-paranormal graphical models are semi parametric models that define a parametric form over the $transformed$ data, where the transformations are smooth functions around the data points. These models were introduced by Liu et al.[45].

A non-paranormal is a Gaussian copula with non-parametric marginals. In this model, data points, X, are transformed to a new space $f(X)$, and are assumed to form a Gaussian graphical model in that space. A non-paranormal is specified by $X \sim NPN(\mu, \Sigma, f)$, where μ and Σ are parameters of the Gaussian graphical model, and f is the function that transforms space X into the space of $f(X)$. Liu et. al. show that if f is a $monotonic$ function with the following two properties, then there exists a closed form solution to inference for the non-paranormal:

$$\mu_j = E[X_j] = E[f(X_j)] \text{ ,and } \sigma_j^2 = Var[X_j] = Var[f(X_j)]$$

These two properties lead to a specific form for f, which is $f_j(x) = \mu_j + \sigma_j \Phi^{-1}(F_j(x))$ for each dimension j. In this definition, $F_j(x)$ is the cumulative distribution function of X_j.

Structure and parameter learning in non-paranormals are accomplished by maximum likelihood estimation. In order to perform structure learning, after the data is transformed, L1-regularized likelihood term is optimized over the training data to get a sparse structure and the parameters. They use convex optimization formulation presented by Banerjee et. al. [5] to optimize the likelihood and infer the sparse Gaussian graphical model in the f space.

5.2 Nonparametric Forest Graphical Models

Lafferty et. al. recently proposed a non-parametric forest structure learning method[38], which provides an alternative to the non paranormal. This model is based on nonparametric Mutual Information, calculated using kernel density estimation. The forest structure is then learned using maximum spanning tree algorithm[35][65].

In this graphical model, if the number of variables is larger than the number of samples, a fully connected graph leads to high variance and over-fits the training data. To solve this issue, Lafferty et. al. use cross validation to prune the tree to get a forest that optimizes log likelihood over held-out data. This model is strong and efficient, but has a major shortcoming: Not all relationships between the variables are always acyclic, specially in applications such as computational molecular biology. They propose alternative structure learnings based on nonparametric sparse greedy regression [37], which they have not yet tested in this context.

5.3 Nonparametric Kernel Space Embedded Graphical Models

Kernel based methods have a long history in statistics and machine learning. Kernel density estimation is a fundamental nonparametric technique used for estimating smooth density functions given finite data, which has been used by community since 1960s when Prazen provided formulations for it in [62].

A kernel is a positive semidefinite matrix that defines a measure of similarity between any two data points, based on the linear similarity (i.e dot product) of the two points in some feature space, ϕ.

Examples of kernels include Gaussian(RBF) Kernel, $K_\lambda(x, y) = e^{-\lambda||x-y||^2}$ and Laplace kernel $K(x, y) = e^{-\lambda|x-y|}$. In the case of Gaussian kernel, for instance, the corresponding feature space into which the data is projected is an infinite dimensional space based on the Taylor

expansion of the RBF kernel function, $\phi(x) = e^{-\lambda x^2}[1, \sqrt{\frac{2\lambda}{1!}}x, \sqrt{\frac{(2\lambda)^2}{2!}}x^2, \sqrt{\frac{(2\lambda)^3}{3!}}x^3, ...]$ [43], and $k_{RBF}(x, y)$ is equal to the dot product of $\phi(x)$ and $\phi(y)$.

Usually we use such feature spaces in algorithms which only use the $dot\ product$ of the two feature vectors, and never use one feature vector on its own. Since the kernel function is the closed form result for the dot product of the feature vectors, such algorithms will be very efficient and powerful. This technique of replacing dot product in the feature space in the algorithms which use the dot product of the x_is, is usually referred to as the $kernel\ trick$, and is an essential trick to create efficient kernel methods.

5.3.1 Kernel Density Estimation and Kernel Regression

In kernel density estimation, given a dataset $X = x_1, x_2, ...x_n$, and a Kernel function K, the density function $f(x)$ can be estimated as:

$$\hat{f}_\lambda(x) = \frac{1}{n}\sum_{i=1}^{n} K_\lambda(x - x_i)$$

This formulation allows a smooth and differentiable density function instead of a histogram, and is extensively used in signal processing and econometrics. Figure 5.1 shows an example of kernel density estimation for a sample dataset $X = \{-2.1, -1.3, -0.4, 1.9, 5.1, 6.9\}$, using Gaussian kernel with $\lambda = 2.25$.

In addition to density estimation, kernel methods have been used for nonlinear regression[58], [88], as well. Roth [70] proposed sparse kernel regression, which uses support vector method to solve the regression problem.

Given a data set $D = \{(x_1, y_1), (x_2, y_2), .., (x_N, y_N)\}$, $linear$ regression tries to minimize the squared error, $\sum_{i=1}^{N}(y_i - w^T x_i)^2 + \lambda||w||^2$. Taking derivative with respect to the regression coefficient w and setting it to zero results in $w = (\lambda I + \sum_i x_i x_i^T)^{-}1(\sum_i y_i x_i)$. Since this formulation only deals with the dot product of the x_is, we can use the kernel trick to replace

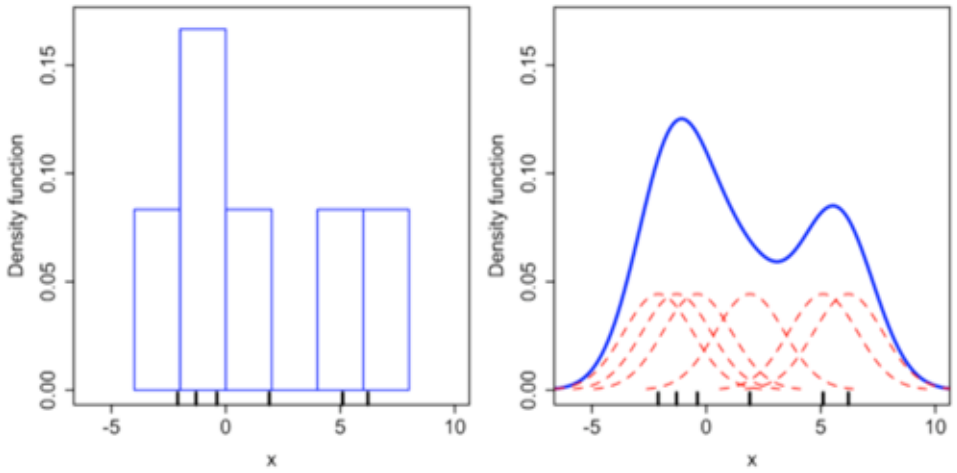

Figure 5.1: Kernel Density Estimation Example

this dot product with a suitable kernel such as Gaussian kernel, and this enables us to perform nonlinear regression.

5.3.2 Reproducing Kernel Hilbert Space embedding of graphical models

A $Hilbert$ space is a complete vector space, endowed with a dot product operation. When elements of H are vectors, each with elements from some space, F, a Hilbert space requires that the result of the dot product be in F as well. For example, the space of vectors in $\Re^n$ is a Hilbert space, since the dot product of any two elements is in $\Re$. [16].

Reproducing kernel Hilbert space is a Hilbert space defined over a $reproducing\ kernel\ function$. Reproducing kernels are the family of kernels that define a dot product function space, which allows any new function, $f(x)$, to be evaluated as a dot product of the feature vector of x, $\phi(x)$, and the f function. In other words,

$$f(x) = \langle K(x, .), f(.) \rangle$$

And Consequently, $k(x, x') = \langle K(x, .), K(x', .) \rangle$

This reproducing property is essential to define operations required for calculating $expected$

$values$ of functions and belief propagation messages in kernel space.

In order to define embedding of graphical model in RKH space, we will first review how a simple probability distribution is embedded in this space. and then look at how conditional probabilities can be represented in this space. And then we have all the building blocks to represent and embed our graphical model in kernel Hilbert space. Finally we'll review how the belief propagation is performed non-parametrically in this space. In the rest of this section we will briefly mention each of these steps.

Smola et. al.[75] provided the formulations to non-parametrically embed probability distributions into RKH spaces. Given an iid dataset $X = \{x_1, ..., x_m\}$, they define two main mappings:

$$\mu[P_x] = E_x[k(x, .)]$$

$$\mu[x] = 1/m \sum_{i=1}^{m} k(x_i, .)$$

Using the reproducing property of the RKH space, we can then write the expectations and empirical mean of any arbitrary functions f as:

$$E_x[f(x)] = \langle \mu[P_x], f \rangle$$

$$\langle \mu[X], f \rangle = 1/m \sum_{i=1}^{m} f(x_i)$$

The authors prove that if the kernel is from a universal kernel family[81] then these mappings are injective, and the empirical estimation of the expectations converges to the expectation under the true probability, with error rate going down with rate of $O(m^{-1/2})$, where m is the size of the training data. Figure 5.2 shows an example of the transformation from the variable space into feature space defined by the reproducing kernel, and the RKHS mappings defined for empirical and true expectations.

To embed conditional distributions in RKH space, Song et. al. [77] define covariance operator, on pairs of variables (i.e. $D_{XY} = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$), as:

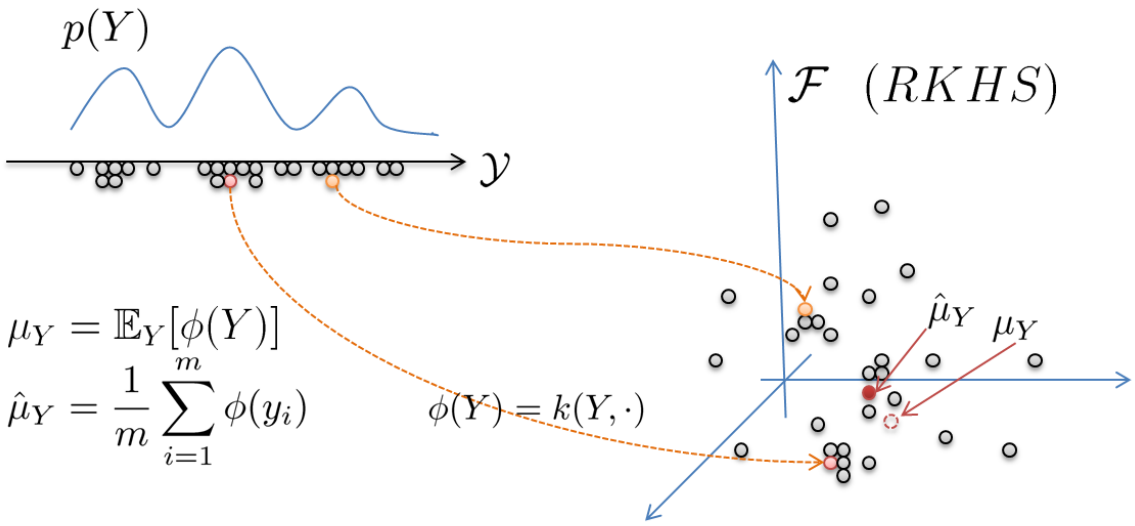

Figure 5.2: Reproducing Kernel Hilbert Space embedding of a probability distribution

$$C_{X,Y} = E_{X,Y}[\phi(X) \otimes \phi(Y)] - \mu_X \otimes \mu_Y$$

where $\otimes$ is the tensor product, the generalization or the product in the variable space.

This allows the covariance of any two functions to be estimated from the data:

$$C_{f(x),g(y)} = E_{X,Y}[f(x)g(y)]$$

$$\hat{C}_{f(x),g(y)} = 1/m \sum_{i=1}^{m} f(x_i)g(y_i)$$

Using covariance operator, we can then define the conditional-mean mapping. The main requirement for a conditional mean mapping is that one should be able to use reproducing property to take conditional expectations, $E_{Y|x}[g(Y)] = \langle g, \mu_{Y|x} \rangle_G$. It turns out that the following definition satisfies this requirement:

$$\mu_{Y|x} = U_{Y|X}\phi(x) = C_{Y,X}C_{X,X}^{-1}\phi(x)$$

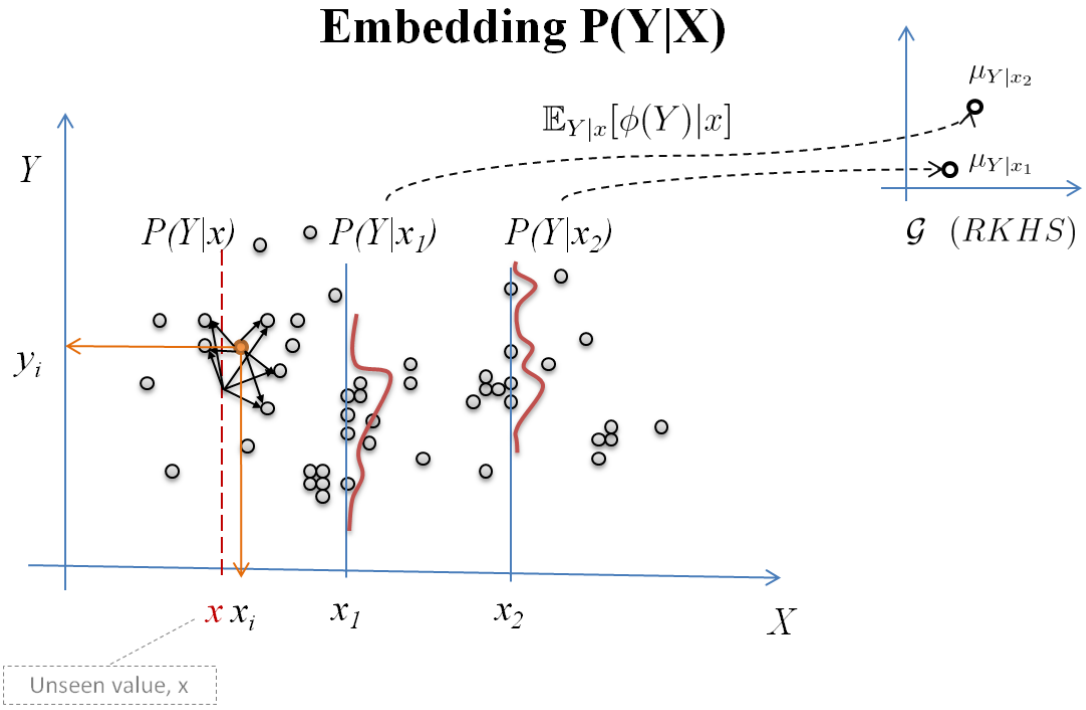

Figure 5.3: Reproducing Kernel Hilbert Space embedding of conditional distributions

Where $U_{Y|X}$ can be estimated from the data, as $\hat{U}_{Y|X} = \Phi(K + \lambda m I)^{-1}\Upsilon^T$, where K is the kernel matrix over samples X, and Φ and Υ are feature matrices over X and Y, respectively.

Based on the definitions above, and the reproducing property, for any new value of x, $\hat{\mu}_{Y|x}$ can now be estimated as $\langle \hat{U}_{Y|X}, \phi(x) \rangle$, which with some re-arrangements can be rewritten as $\sum_{i=1}^{m} \beta_x(y_i)\phi(y_i)$ with $\beta_x(y_i) \in \Re$.

We note that $\hat{\mu}_{Y|x}$ resembles $\hat{\mu}_Y$, except that we have replaced the $1/m$ with $\beta_x(y_i)$s, where $\beta_x(y_i) = \sum_{j=1}^{m} K(x, x_j)K(x_i, x_j)$. This means that we now weight each feature function $\phi(y_i)$ by how similar x is to the corresponding x_i. Figure 5.3 shows an example of a two dimensional data and the conditional mean mapping in the RKHS.

Now that we reviewed how to represent conditional means and marginals in the RKH space, we can represent a graphical model as a set of conditional and marginal factors. In [78], Song et. al. represent a Tree graphical model in RKHS, and provide formulations to perform belief propagation on this tree graphical model, non-parametrically. In [79], Song et. al. provide the belief

propagation on the $loopy$ graphical models, non-parametrically. In both of these models and methods, it is assumed that the structure of the graph is previously known. This is an assumption that is impractical for our purposes, and we will focus in our thesis to use sparse structure learning methods, such as neighborhood selection[55], that have been successful in other contexts, to learn the structure in the RKH space, and perform nonparametric inference.

5.3.3 Belief Propagation in RKHS

Belief propagation in RKHS requires the beliefs and messages to be represented non-parametrically. There are three major operations that is performed during the belief propagation inference, which needs to be non-parametrically modeled. First: Messages from $observed$ variables are sent to their unobserved neighbors. Second: Incoming messages to an unobserved node are combined to create an outgoing message to other unobserved nodes. Third: All incoming messages are combined to create the marginal beliefs at the root node, after the convergence. In [78] and [79], the following formulations are presented:

First: A message from observed variable is simply the conditional probability of the target node, given the observed node. In RKHS, we can simple represent it as $m_{ts}(x_s) = P(x_t|x_s)$, which is estimated through conditional mean mapping as:

$$\hat{m}_{ts} = \Upsilon_s \beta_{ts}$$

$$\beta_{ts} := ((L_t + \lambda I)(L_s + \lambda I))^{-1} \Upsilon_t^T \phi(x_t)$$

Second: Assuming that all incoming messages into node t, are of the form $\hat{m}_{ut} = \Upsilon_t \beta_{ut}$, then the outgoing message is the tensor product of the incoming messages, which can take advantage of the reproducing property and be simplified by using element-wise product of kernels instead:

$$\hat{m}_{ts}(x_s) = [\bigodot_{u \in \Gamma_t \backslash s} (K_t^{(u)} \beta_{ut})]^T (K_s + \lambda m I)^{-1} \Upsilon_s^T \phi(x_s)$$

Where $\bigodot$ is the element-wise vector product. Again, if we define $\beta_{ts} := (L+\lambda mI)^{-1}(\bigodot_{u\setminus s} K\beta_{ut})$ we can write the outgoing message as $\hat{m}_{ts} = \Upsilon\beta_{ts}$ and this allows for iterative message passing until convergence.

Third: Finally once the message passing converges, the beliefs can be computed similarly at any root node r as:

$$B_r = E_{X_R}[\phi(X_r) \prod_{s\in\Gamma_r} m_{sR}(X_r)]$$

And empirically, if each incoming message to belief node is of the form $\hat{m}_{sr} = \Upsilon_r\beta_{sr}$ then the beliefs are estimated as:

$$B_r = \Upsilon_r(\bigodot_{s\in\Gamma_r} K_r^{(s)}\beta_{sr})$$

where $K_r^{(s)} = \Upsilon_r^T\Upsilon_r^{(s)}$. The (s) indicates that that this feature vector is calculated from available samples that have both r and s, which means the method can take advantage of all samples even if the data is missing the values for some variables in each sample.

In this formulation of the belief propagation, every iteration costs on the order of $O(m^2 d_{max})$ operations, with m being the number of samples, and d_{max} the maximum degree in the graph. In molecular dynamic simulation modeling, where we sometimes have a few thousand samples, there is an scalability issue which we discuss in the next chapter and introduce our solutions.

5.3.4 Tree Structure Learning for RKHS Tree Graphical Models

Recently in [80], Song et. al. proposed a method to perform structure learning for tree graphical models in RKH space. Their method is based on the structure learning method proposed by Choi et. al . [14], where they use a $tree\ metric$ to estimate a distance measure between node pairs, and use that value to select a tree via a minimum spanning tree algorithm [35][65].

According to [14], if there exists a distance measure on the graph such that for every two

nodes, s and t, $d_{st} = \sum_{(u,v)\in Path(s,t)} d_{uv}$, then a minimum spanning tree algorithm based on this distance measure can recover the optimum tree, if the latent structure is indeed a tree. Choi. et. al. propose a distance based on the correlation coefficient, ρ .

$$\rho_{ij} := \frac{Cov(X_i, X_j)}{\sqrt{Var(X_i)Var(X_j)}}$$

For Gaussian graphical models, the information distance associated with the pair or variables X_i and X_j is defined as $d_{ij} := -log|\rho_{ij}|$.

To learn the hidden structure in RKHS, Song et. al. write this measure non-parametrically:

$$d_{ij} = -\frac{1}{2}log|C_{st}C_{st}^T| + \frac{1}{4}log|C_{ss}C_{ss}^T| + \frac{1}{4}log|C_{tt}C_{tt}^T|$$

where C_{ij} is the nonparametric covariance operator between X_i and X_j which can be estimated from the data directly. Using this metric, it is then possible to perform minimum spanning tree algorithm[35][65] with this distance measure, and learn an optimum tree structure non-parametrically in RKHS. In the next chapter, we use this model as well as other structure learning methods, to learn sparse network structure prior to RKH inference.

With this background, in the next two chapters we focus on solutions for two of the challenges of the RKHS embedding of graphical models: In chapter 6 we evaluate several solutions for sparse structure learning, and in chapter 7, we provide two solutions to deal with scalability issue of the kernel embedded graphical models.

Chapter 6

Sparse Structure learning for Graphical Models in Reproducing Kernel Hilbert Space

As we discussed in previous chapter, a powerful model for handling multi-modal complex distributions, and potentially inhomogeneous variable sets is nonparametric graphical models, and in particular, reproducing kernel Hilbert space embedded graphical models. Structure learning in reproducing kernel Hilbert space currently only exists for tree structured graphical models[80]. For applications such as structural biology, where the structure is potentially loopy, tree structures are not reasonable. On the other hand, the space complexity of each message update in Hilbert space inference is $O(N^2 d_{max})$ with N being the number of samples, and d_{max} being the degree of the variable, so it is crucial that we perform $sparse$ structure learning prior to any inference to decrease the maximum degree of the graph in large networks.

For general graph structures, there are already several techniques and measures of conditional independence are available, none of which has been tested in the context of structure learning for reproducing kernel Hilbert space belief propagation. Among the most successful algorithms are Neighborhood Selection [55], Kernel Measures of Covariance[14], Nonparametric Greedy

Sparse Regression(Rodeo)[37], Kernel based Mutual Information[25], and Kernel based Conditional Independence test[92]. In this chapter, we will review these techniques for sparse structure learning in reproducing kernel Hilbert space, and compare them in the context of prediction of protein structure, and in the larger context of full cross-validation and inference.

6.1 Sparse Structure Learning in Kernel Space by Neighborhood Selection

The problem of structure learning in Markov random fields is NP-Hard, which, for real world applications such as protein structure prediction, becomes infeasible to solve as a single global optimization problem.

This problem stems from the fact that the partition function in undirected graphical models needs an integration of all variables. Neighborhood selection method, as proposed by Meinshausen et. al. [55], tries to break this optimization into a set of smaller optimization problems: By maximizing $Pseudo-likelihood$, instead of the full likelihood. Meinshausen et. al. show that each optimization term in the pseudo likelihood becomes equivalent to a sparse regression problem, and can be solved efficiently with the Lasso regression[84]. They prove that this method is consistent, and with enough training data, recovers the true structure with probability 1. According to Zhao et.al. [93] and Ravikumar et.al[67], neighborhood selection method has the better sample complexity for structure learning, compared to other methods including graph lasso.

Given the training dataset $D = x_1, x_2, ...x_N$, where each $x_i = (x_{i1}, x_{i2}, ..., x_{id})$ is a $d-$dimensional sample. For variable a, we optimize the lasso regression objective function:

$$\theta^{a,\lambda} = argmin_\theta ||x_{.a} - x_{.\hat{a}}\theta||_2^2 + \lambda||\theta||_1$$

where $\theta^{a\lambda}$ is the vector of regression coefficients, $x_{.a}$ is the column vector of a element of all

samples, $x_{.\hat{a}}$ is the matrix of all but a column of all samples, and λ is the regularization penalty. This optimization problem can then be solved using multiple algorithms. We use an algorithm based on Active set construction[60], and implemented by Schmidt [71]. In this algorithm, iteratively, the coefficient which has the highest absolute effect on the regressors is added to the list of selected variables. This algorithm is widely popular for solving regularized least square optimization problem, because it operates on d variables only, without needing to double the number of variables or generating exponential number of constraints during the optimization process.

This algorithm has exponential convergence rate, and can be executed in parallel, and has the same runtime complexity as the least squared solution. So it has with $O(nd)$ memory requirement, and $O(d^2(n+d))$ runtime complexity, which makes the method among the most scalable methods available for our purposes. The downside of this algorithm is that the method assumes the relationship between variables to be linear with Gaussian noise, and this may be a limiting assumption. In next section we present stronger method that can avoid these assumptions.

6.2 Tree Structure Learning via Kernel Space Embedded Correlation Coefficient

Song et. al. [80] presented the embedding of latent tree structure learning, based on correlation coefficient ρ:

$$\rho_{ij} := \frac{Cov(X_i, X_j)}{\sqrt{Var(X_i)Var(X_j)}}$$

A secondary measure based on this coefficient, $d_{ij} := -log|\rho_{ij}|$, has been used for the tree structure learning, by Choi et. al. [14]. Song et. al. used kernel embedded covariance operator to approximate the d distance measure in kernel space, and subsequently used:

$$d_{ij} = -\frac{1}{2}log|\hat{C}_{st}\hat{C}_{st}^T| + \frac{1}{4}log|\hat{C}_{ss}\hat{C}_{ss}^T| + \frac{1}{4}log|\hat{C}_{tt}\hat{C}_{tt}^T|$$

Where the $\hat{C}$ are the covariance operator, estimated via the Hilbert space embedding of variables.

As a technical note, Song et.al. mention that this measure definition has a restriction, by which the variables have to have the same number of dimensions (i.e. same number of states in discrete variables, and same dimensionality for Gaussian variables). This limitation can be removed by using pseudo-determinant, which is defined simply as the product of top k singular values of the matrix. By using this measure we can now compute pairwise distances between variables, and then perform minimum spanning tree to uncover the tree structure over the variables.

The runtime complexity of this algorithm is $O(N^3d^2)$, and it has $O(d^2N^2)$ memory complexity. Also while the algorithm has consistency guarantees when the true structure is a tree, this assumption doesn't hold for many applications. We next focus on a more general nonparametric structure learning, based on kernel measures of conditional dependence.

6.3 Structure Learning via Normalized Cross Covariance Operator in Kernel Space

Kernel measures of independence has been proposed in the literature before. However the conditional independence these measures have all been dependent not only on the data, but also on the choice of kernel parameter. Fukumizu et.al.[25] proposed a kernel based measure of conditional independence, which is independent of the choice of kernel, and is only based on the densities of the variables.

This measure is based on kernel embedded conditional covariance operator:

$$\Sigma_{YX|Z} = \Sigma_{YX} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZX}$$

Note that Σ is the nonlinear extension of covariance matrix, and Fukumizu et.al. prove that if extended variables $\ddot{X} = (X, Z)$ and $\ddot{Y} = (Y, Z)$ are used, there is an equivalence between the two conditions: $X \perp Y|Z$ is true, if and only if $\Sigma_{\ddot{Y}\ddot{X}|Z} = 0$.

Based on this definition, Fukumizu et.al. write define the $normalized$ cross covariance, which is independent of choice of kernel:

$$V_{YX|Z} = \Sigma_{YY}^{-1/2}(\Sigma_{YX} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZX})\Sigma_{XX}^{-1/2}$$

They show that while both $\Sigma_{YX|Z}$ and $V_{YX|Z}$ encode the conditional dependency, the normalized measure ($V_{YX|Z}$) removes the effect of the marginals and encodes the existing conditional dependency more directly.

And now we can use the Hilbert Schmidt(HS) norm (HS norm $||A||_{HS} := trace(A * A)$) of the normalized cross covariance, as our measure of conditional independence:

$$I^{NOCCO}(X, Y|Z) = ||V_{\ddot{Y}\ddot{X}|Z}||_{HS}^2$$

Given kernel matrices defined for variables $K_{\ddot{X}}$, $K_{\ddot{Y}}$ and K_Z, we can then compute the $I^{NOCCO}(X, Y|Z)$ empirically as:

$$\hat{I}^{NOCCO}(X, Y|Z) = Tr[K_{\ddot{Y}}K_{\ddot{X}} - 2K_{\ddot{Y}}K_{\ddot{X}}K_Z + K_{\ddot{Y}}K_ZK_{\ddot{X}}K_Z]$$

We use this normalized kernel measure of conditional independence for structure learning prior to RKHS inference. In order to discover the network, we perform this conditional independence test for all pairs of variables:

$$X \perp Y | [Z = \text{all variables except X and Y}]$$

If the $I^{NOCCO}(X, Y|Z) == 0$ we consider the two variables X and Y to be unconnected in the network.

This method, without any adjustments, has $O(d^3N^2)$ memory requirement(for N samples each in d dimension), and the runtime is $O(d^3N^4)$, which can be too expensive for practical purposes. As we will see in the next chapter, there are possible solutions to approximate kernel matrices with low rank components, but with very high impact on accuracy.

6.4 Other relevant structure learning methods

Many other relevant structure learning solutions exists, which we will review briefly in this section. Following Fukumizu et.al.[25], in a recent work , Zhang et. al.[92] propose a new conditional dependence test, in which one does not directly estimate the conditional or joint densities, and computes the null hypothesis probability directly from the kernel matrices of the variables. This method is less sensitive to dimensionality of the conditioning variable, however unfortunately the scalability of the algorithm is weak, and the method was not scalable to variable and sample sizes reasonable in our applications.

Another method, based on sparse non-parametric regression, is Rodeo [37]. In this method, we can perform variable selection by testing the sensitivity of the estimator function to the bandwidth of the kernel defined over that variable. The relevant variables are then selected by applying a threshold on the bandwidth of all variables and selecting the variables which have lowest bandwidth. In theory the method is fast, however the calculation of gradient requires multiplication of kernel methods for all variables, which in effect cause numerical instabilities and underflows, and is not scalable to high dimensions without significant modification, so we were not able to take advantage of this method.

Finally we investigated Lin et.al.'s component selection in multivariate nonparametric regression (COSSO) method[44]. COSSO is based on approximating the estimator with sum of splines of different orders. Typically additive splines are most commonly used. The COSSO method performs nonparametric variable selection by optimizing the spline approximation of the estimator, modified by sum of L1 norm of the spline components. This measure is closely related to Lasso regression, but can be extended to incorporate nonlinear functions as well, specially using kernel estimation. Currently the model is developed and used in space of linear functions, so we did not take advantage of this method. An interesting direction for future work is to develop the model for more general family of functions, and use it for structure learning in nonparametric models. The next section will cover our experimental results.

6.5 Experiments

In this section we perform our experiment on synthetic and protein structure molecular simulation data. We first describe the results on the synthetic data, and then describe our results on the protein simulation data.

6.5.1 Experiments on Synthetic Data

To generate our synthetic data, we first draw the binary edge structure randomly with a desired edge density ρ. The strength of the dependency was then drawn from a Gaussian distribution $N(0, \lambda_{edge})$. We also draw the variance of variables from $Uniform(1, \kappa_{variable})$, and checked to ensure the structure is symmetric and positive definite. In our dataset, we set $\lambda_{edge} = 10$, $\kappa_{variable} = 10$, and $\rho = 0.3$.

After we sampled graph structure, Σ^{-1}, we draw 10,000 independent samples from a Gaussian multivariate distribution $G(0, \Sigma)$. In order to add non-linearity to the samples, we then randomly selected 50% of variables and replaced them with the square of their values. This

Sample Size	50	100	1000	5000	10000
Neighborhood Selection (λ=0.1) AUC	**0.5528**	0.5552	0.5528	0.551	0.5468
Neighborhood Selection (λ= 1) AUC	**0.5673**	**0.5802**	0.6758	0.7022	0.7067
Neighborhood Selection (λ= 10) AUC	0.5525	**0.6185**	**0.731**	**0.7592**	**0.7598**
Neighborhood Selection (λ= 100) AUC	0.5501	0.5288	**0.695**	**0.7238**	**0.7260**
Neighborhood Selection (λ= 1000) AUC	0.5375	0.5072	0.695	0.7235	0.7256
Nonparametric Tree Structure Learning AUC	0.5504	0.5459	0.5591	0.5628	Not Scalable
Normalized Cross Covariance Operator AUC	0.5401	0.5578	0.6021	Not Scalable	Not Scalable

Table 6.1: Area Under ROC curve for structure learning methods in 100 dimensional space.

caused a subset of relationships to be nonlinear.

We then experimented with structure learning methods with different sizes of training data: Very small training data of size=50 to large training data of size=10,000.

We now report the quality and runtime of structure estimation methods for all three methods of Neighborhood Selection, Nonparametric Tree learning, and Normalized Cross Covariance Operator.

In table 6.1, we compares the Area Under ROC Curve (AUC), for Neighborhood selection structure learning with different regularization penalty; the Tree structure learning, and Normalized Cross Covariance Operator(NOCCO) and for different sample sizes, for 100 dimensions. We used Gaussian kernel $K = e^{-\lambda||x-y||^2}$ with $\lambda = 0.3$ for the NOCCO method. Table 6.2 shows the AUC results for dataset of dimension of 1000.

Table 6.3 shows the computational time for the same methods.

Based on our experiments, we observe that Neighborhood selection significantly outperforms other structure learning methods, and is also more scalable and faster, despite the variables'

Sample Size	50	100	1000	5000	10000
Neighborhood Selection (λ=0.1) AUC	0.5054	0.5053	0.5059	0.5067	0.5066
Neighborhood Selection (λ= 1) AUC	0.5061	0.5057	0.5113	0.5167	0.5186
Neighborhood Selection (λ= 10) AUC	0.5251	0.5257	0.5589	0.5222	0.5617
Neighborhood Selection (λ= 100) AUC	0.5776	**0.6123**	**0.6223**	0.6279	**0.6793**
Neighborhood Selection (λ= 1000) AUC	**0.5812**	**0.6166**	**0.6301**	**0.6701**	**0.679**
Nonparametric Tree Structure Learning AUC	0.5124	0.5292	Not Scalable	Not Scalable	Not Scalable
Normalized Cross Covariance Operator AUC	0.5092	0.5245	Not Scalable	Not Scalable	Not Scalable

Table 6.2: Area Under ROC curve for structure learning methods in 1000 dimensional space.

Sample Size	100	500	1000	5000	10000
Neighborhood Selection (λ=0.1) CPU	2.92	4.66	6.31	14.81	17.91
Neighborhood Selection (λ= 1)	16.13	27.11	35.78	75.05	125.31
Neighborhood Selection (λ= 10)	102.32	67.20	92.01	157.97	295.41
Neighborhood Selection (λ= 100)	160.43	84.15	92.39	164.03	247.66
Neighborhood Selection (λ= 1000)	214.26	88.69	94.50	167.88	252.92
Nonparametric Tree Structure Learning AUC	12.41	568.23	4832.89	32.0972e04	Not Scalable
Normalized Cross Covariance Operator AUC	16.21	619.39	5.5609e03	Not Scalable	Not Scalable

Table 6.3: CPU time(in seconds) taken for structure learning methods for 100 dimensions.

relationship to be nonlinear. These results imply that for structure learning, linear methods may be a reasonable solutions. In fact the simplicity of the linear models add to the robustness of the estimation, which in high dimensional setting becomes an advantage. Better scalability and speed of model estimation for these models also becomes an important advantage in applications with high dimensional data points.

6.5.2 Experiments on Protein Molecular Dynamics Simulation Data

We also performed our experiments on real protein simulation data, to evaluate the structure learning methods for the purpose of structure imputation and modeling. Similar to the von Mises experiments, here we performed our experiments over Engrailed protein dataset again. The characteristics of the data was covered in section 3.7.2. We used the same two sub-sampled datasets, first1000, and uniformly sampled 1000, as described in figure 3.9.

We performed leave-one-out cross validation. For each test frame, we assumed randomly selected 50% of the variables of the frame are observed, and predicted the rest of the variables, given these observations and the training data. For each frame we repeated this 50% subset selection 20 times.

In all the experiments, we first normalized the data before the learning, then performed structure learning and prediction, and finally rescaled the predictions before computing the RMSE score.

Table 6.4 shows the results of running the full cross validation experiment, on RKHS inference after Neighborhood selection as the structure learning method, with two different kernels, versus the Non-paranormal and Sparse Gaussian Graphical model. In all cases, we see the RMSE (measured in degrees) of the predicted hidden variables conditioned on the observed variables, and the RMSE is calculated from the difference of predicted and actual values of the hidden variables.

We note that in this particular case, where our data is angular, we try two different kernels:

Model	Gaussian Kernel for RKHS	Triangular kernel for RKHS	Non-paranormal	Gaussian Graphical Model
First 1000 samples	8.42	7.30	8.43	8.46
Uniformly sampled 1000 samples	54.76	51.34	63	59.4

Table 6.4: RMSE result comparison for RKHS with neighborhood selection(using two kernels), compared with non-Paranormal and Gaussian graphical models. (Pairwise Wilcoxon rank test P-values of nonparametric vs. NPR and GGM are smaller than 7.5e-007 for all cases.)

Neighborhood selection with Triangular kernel	Tree structure learning with Triangular kernel	NOCCO structure learning with Triangular kernel
7.30	7.41	7.39

Table 6.5: RMSE result for RKHS using Neighborhood selection, versus Tree structure learning versus Normalized Cross Covariance Operator on First1000 dataset. All methods used triangular kernel.

Gaussian kernel, $K_1 = e^{-\lambda||x-y||^2}$, and triangular kernel, $K_2 = e^{-\lambda(sin(||x-y||))^2}$.

Table 6.5 shows the RMSE results of tree structure learning, versus the neighborhood selection method, versus the nonparametric Normalized Cross Covariance Operator(NOCCO) on the first 1000 sample dataset, and using the Triangular kernel in both cases.

As you can see from these results, without a relevant kernel, RKHS models do not outperform the non-paranormal and Gaussian graphical models. However, if the kernel is well suited for the problem(i.e. triangular kernel function for angular data), we see significant improvement in RMSE score of neighborhood selection for structure learning in RKH space, over non-paranormal and Gaussian graphical models.

Also we observe that neighborhood selection outperforms the tree-based nonparametric structure learning, and NOCCO method. However we should note that neighborhood selection with Gaussian kernel over angular data does worse than tree structured method and the NOCCO method with triangular kernel. This proves the importance of Kernel selection and learning, and we discuss the possibilities in the proposed future work in section **??**. We also observe that NOCCO method, does not outperform the kernel based tree structure learning method. Based on

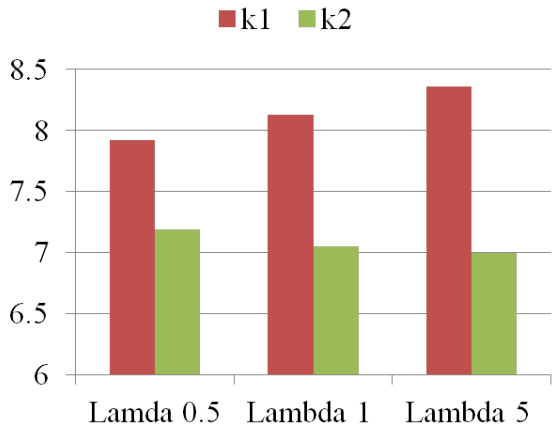

Figure 6.1: Effect of structure sparsity in neighborhood selection on RMSE in RKHS inference

the experiments on the synthetic data, we attribute this lack of performance to the inability of the method to be robust when number of variables increases.

We also investigated the effect of the density of the estimated structure learned by neighborhood selection on the RMSE of the predictions. Using different regularization penalties in Lasso regression, results in different levels of sparsity. Figure 6.1 shows the RMSE for different values of the regularization penalty, measured with both Gaussian and Triangular kernels, when modeling the first1000 dataset. As the graph becomes denser, the Triangular kernel performs better. The Gaussian kernel, on the other hand, does not benefit from denser graphs.

Finally we compared the best result achieved by RKHS, with all the methods previously presented including von Mises and mixture of von Mises model. Table 6.6 shows the results, for the leave-one-out cross validation experiment over First1000 samples Engrailed angle dataset.

Our results indicate that on this dataset where the distribution exhibits fewer complexity (i.e. Figure 3.9), and when we deal with only angular variables, mixture of von Mises graphical model is the most suitable model and handles the multi-modality and angularity of the data. We also see that mixture of Gaussian and mixture of Nonparanormal, while outperforming the single Gaussian and single Nonparanormal models, still can not outperform models designed for angles. RKHS models show good performance compared to Gaussian and Nonparanormal

Model	RKHS with Triangular kernel	Non-paranormal	Gaussian	Von Mises	Mixture of Gaussian (k = 50)	Mixture of non-paranormal (k = 50)	mixture of von Mises (k=30)
RMSE (degree)	7.30	8.43	8.46	6.93	8.21	7.63	5.92

Table 6.6: RMSE result comparison for RKHS with Nonparanormal, Gaussian, von Mises, Mixture of Gaussian, Mixture of Nonparanormal and Mixture of von Mises models, on First1000 dataset. (All differences are significant at wilcoxon rank p-value of 1e-3 level)

Model	Gaussian Kernel lambda=1 (dense)	Gaussian Kernel lambda=0.5 (sparse)	Non-paranormal	Gaussian Graphical Model
First 1000 samples	-	0.72	1.16	1.15
Uniformly sampled 1000 samples	2.48	2.36	4.91	5.07

Table 6.7: RMSE result comparison for RKHS, non-Paranormal and Gaussian graphical models over Distance variables

models, however. And since these models have the benefit that they easily extent to all variable types, we still focus on them and try to improve the scalability of them in the next chapters.

6.5.3 Experiments on Pairwise Distances Network

We also experimented with a non-angular representation of the Protein structure, which is based on pairwise distances. For a protein of length N amino acids, the C_α of each amino acid can pinpoint its location given its distance to 4 other amino acid C_αs, so we used $4N$ pairwise distances to represent Engrailed protein structure. As before, we used sub-sampled data.

Figure 6.2 shows a collection of univariate marginals in both of the sub-sampled data set, and we see that multi-modal and asymmetric distributions are very common in both datasets.

We calculated the RMSE error(measured in Angstrom) using Gaussian kernel, with two different graph densities, and also calculated the RMSEs using non-Paranormal and Gaussian graphical models as well. Table 6.7 shows the result of the RMSE calculations:

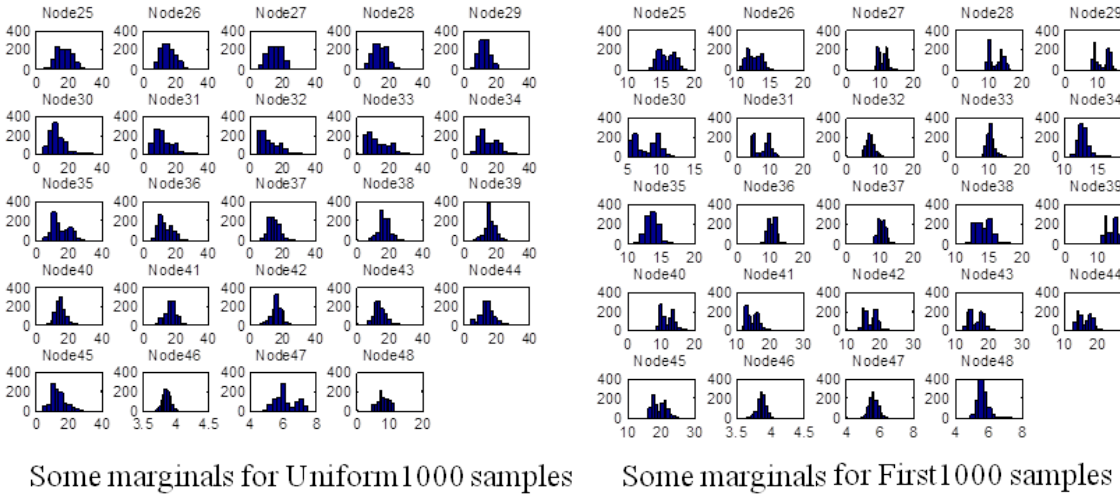

Figure 6.2: Marginal Distributions for a subset of pairwise distance variables in two sample sets

Based on these results, we see that RKHS with Neighborhood selection outperforms other semi-parametric and Gaussian methods in predicting the distances, as well as the angles.

6.6 Summary

In this chapter, we evaluated several sparse structure learning methods, to use prior to reproducing kernel Hilbert space inference.

We compared neighborhood selection, nonparametric tree structure learning, and kernel based normalized cross covariance operator on two different data types: Synthetic data, and Protein molecular dynamics simulation data. We showed that a relevant kernel is important for inference, and through our experiments, showed that Neighborhood selection with Triangular kernel outperforms other structure learning methods, and also showed that the neighborhood selection structure learning along with inference in RKH space outperforms non-paranormal and Gaussian graphical models.

While inference in the RKHS is very promising, there are currently several issues left to tackle. The main disadvantage of the RKHS nonparametric models is their scalability issue.

Both non-Paranormal and Gaussian graphical models are easily scalable to very large datasets, and computations are extremely efficient, once the learning is complete, whereas RKHS model can not scale beyond a few thousand samples with any reasonable size of variables. In the next chapter, we provide some solutions to this problem.

Chapter 7

Scaling Reproducing Kernel Hilbert Space Models: Moving from Kernel space back to Feature Space

As we saw in previous chapters, taking advantage of full power of nonparametric models comes at the price of intense memory and computation requirements. In particular, the space complexity of a RKHS-embedded graphical models is $\Omega(N^2 d d_{max})$, where N is the number of training samples, d is the number of dimensions and d_{max} is the maximum degree of the graph. This complexity is prohibitive in the contexts where we have large complex datasets.

Many techniques have been proposed to increase the scalability of kernel machines. Representative examples include: low rank approximations of the kernel matrix (e.g., [76],[89],[59],[42]) and feature-space methods (e.g., [66]). Song et.al. have studied the use of low rank approximations on RKHS-embedded graphical models [79], but the advantages and disadvantages of feature-space approximations have not been studied previously for this class of model.

In this chapter, we derive a feature-space version of Kernel Belief Propagation, and investigate the scalability and accuracy of the algorithm using the random feature selection method presented in [66] as a basis. Additionally, we explore the use of different strategy for improving

scalability by adapting the Coreset selection algorithm [22] to identify an optimal subset of the data from which the model can be built.

7.1 Background

Kernel Belief Propagation relies on kernel matrices, which require $O(N^2 d d_{max})$ space, where N is the number of samples, and d is the number of random variables, and d_{max} is the maximum degree of the graph. During the belief propagation, each message update costs $O(N^2 d_{max})$. Song et.al. show that these update costs can be reduced to $O(l^2 d_{max})$, where $l \ll N$, by approximating the feature matrix, Φ, using a set of l orthonormal basis vectors obtained via Gram-Schmidt orthogonalization [79]. They also show that the updates can be further reduced to constant time by approximating the tensor product.

7.2 Random Fourier Features for Kernel Belief Propagation

An alternative strategy for increasing scalability is to use feature space approximations. For example, when dealing with continuous variables and Gaussian kernels, the Random Fourier Features method may be used[66]. This method maps the feature vectors of shift-invariant kernel functions (e.g., Gaussian kernel) onto a lower dimensional space. Function evaluations are then approximated as a linear product in that lower dimension space.

The idea behind the method is as follows: It is well known that the kernel trick can be used to approximate any function, f, at a point x in $O(Nd)$ time as: $f(x) = \sum_{i=1}^{N} c_i k(x_i, x)$, where N is the total sample size, and d is the dimension of x in the original space. Alternatively, it is possible to represent the kernel function $k(x, y)$ explicitly as a dot product of feature vectors, and then learn the explicit mapping of the data to a D-dimensional inner product space, using a randomized feature map:

$$z : \Re^d \rightarrow \Re^D$$

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \approx z(x)z(y)$$

Rahimi and co-workers show that for shift invariant kernels, it is possible to approximate the kernel functions to within ϵ error, with only $D = O(d\epsilon^{-2}log(1/\epsilon^2))$ dimensional feature vectors.

Using these approximated feature mappings, functions can now be estimated directly in the feature space as linear hyperplanes (i.e. $f(x) = w'z(x)$). This decreases the cost of evaluating $f(x)$ from $O(Nd)$ to $O(D + d)$

Rahimi et.al's proposed feature mapping is based on transformations of random Fourier features. Briefly, any function (including kernel functions) can be represented exactly by an infinite sum of Fourier components. Thus, by sampling from this infinite dimensional vector, one can approximate the function with any level of accuracy.

To get these samples, we draw $2D$ samples from the projection of x into a random direction ω, drawn from the Fourier transform $p(\omega)$ of the kernel, wrapped around the unit circle. It is shown that after transforming x and y in this way, the inner product will be an unbiased estimator of $k(x, y)$.

For a Gaussian kernel (i.e. $k(x, y) = \exp\left[-\frac{||x-y||_2^2}{2\sigma}\right]$), the Fourier transformation is $p(\omega) = (2\pi)^{-d/2}e^{-\frac{||\omega||_2^2}{2\sigma^{-1}}}$. Given samples of ω, the projection of these samples and wrapping it around the unit circle gives $z(x) = \sqrt{\frac{1}{D}}[cos(\omega_1'x)...cos(\omega_D'x)sin(\omega_1'x)...sin(\omega_D'x)]'$, which can be used to approximate $k(x, y) \approx z(x)z(y)$.

Given this approximate feature mapping, we musts then re-write the kernel belief propagation in the feature space rather than the kernel space. This transformation improves the memory cost of the belief propagation algorithm from $O(dN^2)$ into $O(dND)$.

The exact formulation of belief propagation in feature space is based on several algebraic manipulations of the message passing formulations in Hilbert space, and we will review the details below.

7.2.1 Messages from Observed Variables

As we know, messages from observed variable x_s to unobserved one x_t in belief propagation,$m_{st}(x_t)$ is the conditional probability of the unobserved, given the observation. $m_{st}(x_t) = P(x_t|x_s)$

Following Song et.al.[**?**] we can write these messages as:

$$m_{st}(x_t) = A_{st}\phi(x_t)$$

where $\phi(x_t)$ is the feature map defined for variable x_t and A_{st} matrix is computed via the embedded covariance operators:

$$A_{st} = C_{ss}^{-1}C_{st}C_{tt}^{-1}$$

The C matrices are estimations of the covariance in the kernel space, and when we have explicit feature representations $z(x_i)$ for variable x_i,(i.e. $k(x_i, x_j) \approx z(x_i)z(x_j)$), we can calculate the C_{ij} instead as $C_{ij} = z(x_i)'z(x_j)$ directly, and consequently, compute A_{st} in the $feature$ space.

Now again following the Kernel Belief Propagation formulation, if we want to calculate m_{st} at a particular x_{new}, we can do it via RKHS dot product:

$$m_{st}(x_{new}) = \langle m_{ts}(.), \phi(x_{new})\rangle_F$$

7.2.2 Messages from Unobserved Nodes

To formulate the messages from unobserved variables, now let's assume that each message is in the form

$$m_{ut}(.) = A_{tu}\phi(x_u) := w_{ut}$$

. where $\phi(x_u)$ is the feature map of x_u, and A_{tu} is the embedded correlation matrix, defining the connection between variables x_u and x_t. This formulation is based on Song et.al.[**?**]. We

propose to represent this message function as w_{ut}, a weight vector in the feature space.

Note that this formulation directly represents message functions as linear hyperplanes (specified by the w vector in the feature space). In particular, w_{ut} directly specifies the coefficient matrix of this hyperplane. And since it is defined as a function, it can be evaluated at any new point x_{new} simply as

$$m_{ut}(x_{new}) = w'_{ut} z(x_{new})$$

Where again, $z(x_{new})$ is the approximate feature representation for x_{new}.

Using this, we can transform the message update formula given by Song et.al. into the feature space as follows:

As we reviewed in section 5.3.3, we know:

$$m_{ut}(.) = \Upsilon_t \beta_{ut} = A_{tu} \phi(x_u)$$

where Υ_t is the feature meatrix, containing feature map $\phi(x_t)$ for variable x_t in all the training samples. We can approximate the β_{ut} matrix using the samples:

$$\beta_{ut} = ((L_u + \lambda_m I)(L_t + \lambda_m I))^{-1} \Upsilon_u^T \phi(x_u)$$

Now, using the fact that A_{ut} was also calculated as follows:

$$A_{ut} = m \Upsilon_t ((L_u + \lambda_m I)(L_t + \lambda_m I))^{-1} \Upsilon_u^T$$

we derive the relationship between β_{ut} and A_{tu}:

$$\beta_{ut} = \Upsilon_t^{-1} A_{tu} \phi(x_u)$$

Now, we again as we reviewed in section 5.3.3, know that $m_{ts} = \Upsilon_s (K_s)^{-1} \bigodot_{u \in \Gamma_t \setminus s} K_t \beta_{ut}$,

so replacing β_{ut} with $\Upsilon_t^{-1} A_{tu}\phi(x_u)$ will give us:

$$m_{ts} = \Upsilon_s(K_s)^{-1} \bigodot_{u \in \Gamma_t \setminus s} K_t \Upsilon_t^{-1} A_{tu}\phi(x_u)$$

and since $K_t = \Upsilon_t^T \Upsilon_t$,

$$m_{ts} = \Upsilon_s(K_s)^{-1} \bigodot_{u \in \Gamma_t \setminus s} \Upsilon_t^T \Upsilon_t \Upsilon_t^{-1} A_{tu}\phi(x_u) = \Upsilon_s(K_s)^{-1} \bigodot_{u \in \Gamma_t \setminus s} \Upsilon_t^T A_{tu}\phi(x_u)$$

.

Also, we can write $K_s = \Upsilon_s^T \Upsilon_s$, so $K_s^{-1} = \Upsilon_s^{-1} \Upsilon_s^{T(-1)}$, and use pseudo-inverse to replace $\Upsilon_s(K_s)^{-1}$ with $\Upsilon_s \Upsilon_s^{-1} \Upsilon_s^{T(-1)} = \Upsilon_s^{T(-1)}$.

This simplifies our message calculation as:

$$m_{ts} = \Upsilon_s^{T(-1)} \bigodot_{u \in \Gamma_t \setminus s} \Upsilon_t^T A_{tu}\phi(x_u)$$

Now remember that we defined $A_{tu}\phi(x_u)$ to be w_{ut}, so we the message update above can be written as

$$m_{ts} = \Upsilon_s^{T(-1)} \bigodot_{u \in \Gamma_t \setminus s} \Upsilon_t^T w_{ut}$$

So, in summary, if each incoming message from u to t is represented in the feature space as a hyperplane w_{ut}, the outgoing message from t to s, can be calculated by performing an element-wise product of $\Upsilon_t^T w_{ut}$ for all incoming messages from other neighbors of t, and finally, transforming it via multiplication of pseudo-inverse of Υ_s^T.

With this reformulation, we have transformed our memory requirement from $O(dN^2)$ into $O(dND)$, where d is the original dimension of the data, and D is the random feature dimension which will be decided upon, depending on our desired level of accuracy.

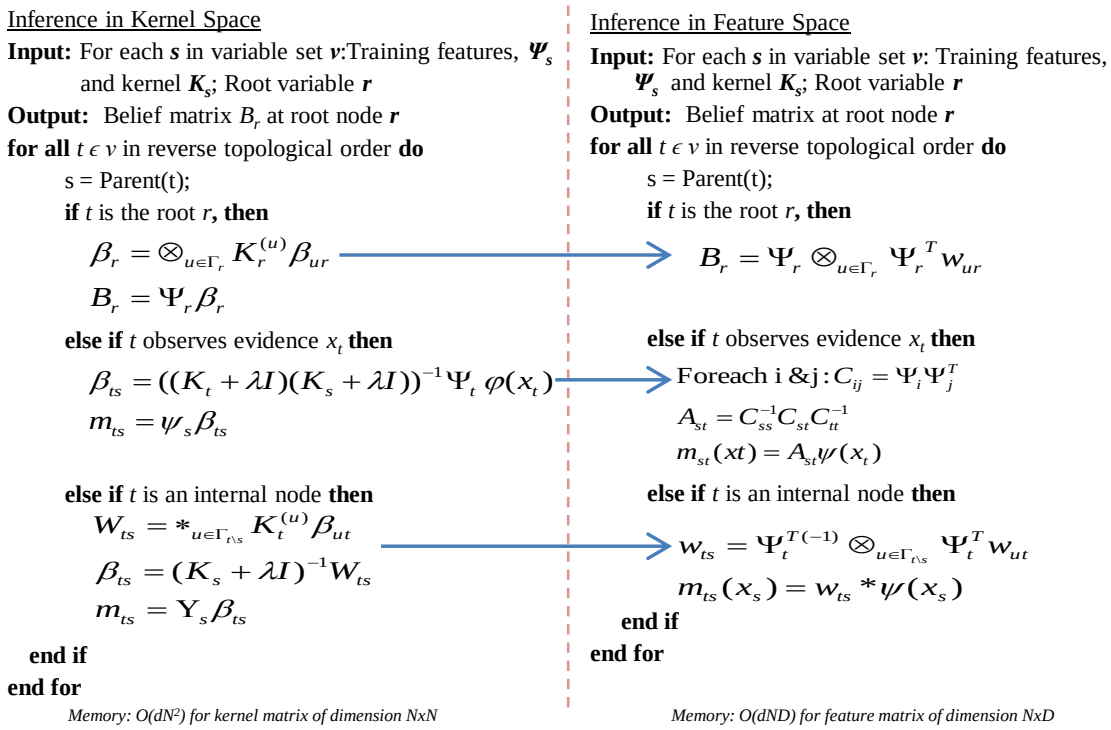

Figure 7.1: Kernel Belief Propagation, and the corresponding steps in explicit feature space. In feature space, each message is represented as a linear weight vector w_{ts}, and the inference and belief calculation is done via calculation of these weights.

7.2.3 Belief at Root node

After the algorithm converges, the final belief at each root node can be calculated similarly:

$$B_r = \Upsilon_r \bigodot_{u \in \Gamma_r} \Upsilon_r^T w_{ur}$$

Figure 7.1 shows the transformed kernel belief propagation algorithm in the explicit D-dimensional feature space. Note that each message from node s to t (i.e. $m_{s \to t}$) is now approximated as a linear function in the new feature space, and is represented by a separating hyperplane with coefficients $w_{s \to t}$.

One of the benefits of our feature belief propagation algorithm is that the messages have more intuitive representation and this improves the interpretability of the intermediate components of

the inference. Our algorithm can also potentially handle any form of feature approximation and not only Fourier based random feature approximation. Although in our experiments we focus on this method, exploring other feature functions is an exciting direction for future work.

7.3 Error analysis of RKHS inference with Random Fourier Features

Rahimi et. al. have shown that the kernel approximation error is bounded as $Pr(|z(x)z(y) - k(x,y)| \geq \epsilon) \leq 2exp(-D\epsilon^2/2)$, for any fixed pair of arguments.

Given this probability, and following [79], the approximation error imposed on each message during the belief propagation will be $O(2(\lambda_m^{-1} + \lambda_m^{-3/2}))$ with probability $2exp(-D\epsilon^2/2)$. Note that λ_m is the matrix regularization term we add to the diagonal of kernel matrix, to ensure the matrix is invertible.

7.4 Sub-sampling for Kernel Belief Propagation

An alternative means for decreasing the cost of kernel methods is to sub-sample the training data. This approach may be a necessary alternative to kernel matrix approximations and feature-space methods when N is large. Sub-sampling can be used in combination other methods. For example, multiple authors have presented variations of the Nystrom method that incorporate sub-sampling [36, 91].

In this thesis, we adapt the $Coreset\ selection$ method introduced by Feldman in the context of learning Gaussian mixture models [22]. A Coreset is a weighted subset of the data, which guarantees that models fitting the Coreset will also provide a a good fit for the original data set. Feldman et.al. show that for Gaussian mixture models, the size of this Coreset is independent of the size of original data. We note that while the Coreset method has not been used previously in

Algorithm: Coreset Construction

Input: Dataset D, ε, δ, k

Output: Coreset $C = \{(x_1, w_1), (x_2, w_2), \ldots (x_{|C|}, w_{|C|}), \}$

$D' \leftarrow D \quad B \leftarrow \phi$

While $|D'| > 10dk\ln(1/\delta)$ **do**

Sample set S of $\beta = 10dk\ln(1/\delta)$ points from D' uniformly
Remove $D'/2$ points $x \in D$ closest to S from D'
Set $B \leftarrow B \cup S$

Set $B \leftarrow B \cup D'$

For each $b \in B$, **do** $D_b \leftarrow$ the points in D whose closest point in B is b;

For each $b \in B$ and $x \in D_b$ **do**

$$m(x) \leftarrow \left\lceil \frac{5}{|D_b|} + \frac{dist(x,B)^2}{\sum_{x' \in D} dist(x',B)^2} \right\rceil$$

Pick a non-uniform random sample C of $10[dk|B|^2 ln(1/\delta)/\varepsilon^2]$ points from D, where for every $x' \in C$ and $x \in D$, we have $x' = x$ with probability $m(x)/\sum_{x' \in D} m(x')$

For each $x' \in C$ **do** $w(x') = \dfrac{\sum_{x \in D} m(x)}{|C|\, m(x')}$

Figure 7.2: Coreset construction algorithm

the context of kernel methods, it does have some similarities to the idea presented in [91]. The key difference is that the Coreset method optimizes the total *distance* of the data points to the selected samples.

Figure 7.2 shows the Coreset selection algorithm. The algorithm starts by constructing a set, B, which includes samples from high and low density sample space. Once the set is constructed, the original points are clustered around elements of B, by some measure of distance. The Coreset is then sampled from the original data, with specific probability, which is defined in such a way to reduce the variance of the log likelihood of the data: Each point's probability is proportional to the linear combination of relative distance to the centroid of its cluster, and the size of the cluster.

Given this sampling algorithm, Feldman et al.[22] prove that with probability $1 - \delta$, the error

of the likelihood of data D, (if data is modeled by mixture of Gaussian) will be bounded as:

$$(1-\epsilon)\phi(D|\theta) \leq \phi(D|\tilde{\theta}) \leq (1-\epsilon)\phi(D|\theta)$$

where $\phi(D|\theta)$ is sum of the data-dependent elements of the log likelihood (i.e. excludes the normalization factor Z, which only depends on the model):

$$\phi(D|\theta) = -(log\_likelihood(D|\theta) - |D| * ln(Z(\theta))$$

One benefit of the method is that it can be performed in an online fashion, over streamed data. This feature improves the scalability of the method. In the online version of the algorithm, the incoming data is compressed in batches independently, and the Coresets are merged and re-compressed in a binary tree structure, and this leads to several layers of compressed data, and at the root of the tree, the error that is imposed by the data compression will be only ($O(log(|D|)\epsilon)$, as opposed to $O(|D|\epsilon)$. [22]

Given our training data, we performed this Coreset selection to sample a core set of data points as our basis points for the purpose of kernel belief propagation. We set $k = log(n)$, in the algorithm, and fixed ϵ to 0.01. While our data is not drawn from a mixture of Gaussian, we will show that the method yields surprisingly good performance and actually outperforms our feature-space KBP algorithm in terms of accuracy on inference tasks. Since the kernel belief propagation method defines the new methods according to its distance to the samples, we believe that having a well represented sample from all regions of input space increases the performance of the model.

7.5 Combining Random Fourier Features and Sub-sampling for Kernel Belief Propagation

It is now natural to consider combinations of feature-space method, and sub-sampling. In this section, we show that the optimal sub-sampling strategy is to sample uniformly.

For KBP, we are primarily interested in estimating RKHS embedded covariance operators, which allows us to compute messages and beliefs. When we switch to explicit Fourier feature space, the covariance can be written simply as $C_{s,r} = \Psi_s \Psi_r'$, assuming the features already have zero means.

Recalling from Section 7.2, one can represent the kernel function as $k(x, y) \approx z(x)z(y)$, where $z(x) = \sqrt{\frac{1}{D}}[cos(\omega_1' x)...cos(\omega_D' x)sin(\omega_1' x)...sin(\omega_D' x)]'$, and the ωs are random samples drown from the Fourier transform of the kernel function. Thus, we can approximate the feature matrix of variable s as $\Psi_s \approx [z(s_1)z(s_2)...z(s_N)]$, which is a $D \times N$ matrix. When explicit feature is employed, based on the analysis in Drineas et.al. [20], we can show that uniform sub-sampling leads to optimal error for approximating covariance matrix in the Fourier feature space.

Theorem 1. Given a dataset $\{x_1, x_2, ..., x_N\}$, kernel function representation $k(x, y) \approx z(x)z(y)$ with $z(x) = \sqrt{\frac{1}{D}}[cos(\omega_1' x)...cos(\omega_D' x)sin(\omega_1' x)...sin(\omega_D' x)]'$, uniform sub-sampling of the data gives *optimal* Hilbert-Schmidt norm for the error of the covariance matrix $\hat{C_{s,r}} = \hat{\Psi}_s \hat{\Psi}_s{}'$, where $\hat{\Psi}_s := [z(s_{i_1})z(s_{i_2})...z(s_{i_l})]$ and $x_{i_1}, x_{i_2}, ..., x_{i_l}$ are uniformly selected random samples from training data.

Proof: For two variables r and s, the error, ϵ, is defined as:

$$\epsilon = E[||C_{s,r_{NN}} - \hat{C}_{s,r_{ll}}||^2_{HS}] = E[||\hat{\Psi}_{s_N}\hat{\Psi}'_{s_N} - \hat{\Psi}_{s_l}\hat{\Psi}'_{s_1}||^2_{HS}]$$

where N is the size of training data, l is the size of sub-sampled data, and s, r_{NN} means the covariance is approximated from N samples of r and N samples of s variables.

According to Drineas and co-workers, the optimal ϵ is achieved when the sub-sampling is performed according to the probability distribution P, where sample k is selected according to [20]:

$$p_k = \frac{|\hat{\Psi}_{s_N}^{(k)}||\hat{\Psi}_{s_N}^{(k)}|}{\sum_{k'=1}^{N}|\hat{\Psi}_{s_N}^{(k')}||\hat{\Psi}_{s_N}^{(k')}|}$$

In case of random Fourier feature set, since $\hat{\Psi}_s = [z(s_{i_1})z(s_{i_2})...z(s_{i_l})]$ with $z(s_{i_1}) = \sqrt{\frac{1}{D}}[cos(\omega_1'x)...cos(\omega_D'x)sin(\omega_1'x)...sin(\omega_D'x)]'$, the norm of the kth column corresponding to the kth sample is simply:

$|\hat{\Psi}_{s_N}^{(k)}|^2 = \frac{1}{D}[cos(\omega_1'x)^2 + ... + cos(\omega_D'x)^2 + sin(\omega_1'x)^2 + ... + sin(\omega_D'x)^2]$

Noting that $cos(\alpha)^2 + sin(\alpha)^2 = 1$: for any choice of α, gives:

$$|\hat{\Psi}_{s_N}^{(k)}|^2 = \frac{2D}{D} = 2$$

Thus, the optimal sampling probability for sample k is $p_k = \frac{2}{\sum_{k'=1}^{N} 2} = \frac{1}{N}$, which is simply the uniform distribution.

7.6 Experiments

In the first set of experiments, we examine the quality of the kernel function approximation using the random Fourier features approach. To do this, we generated two synthetic data sets of size 1,000 samples. The first data set is drawn from a uniform distribution, and the second from a standard Gaussian distribution. Figure 7.3 shows the relative error of the kernel function approximation, using a Gaussian kernel ($k(x, y) = \exp\left[-\frac{||x-y||_2^2}{2\sigma}\right]$) with $\sigma = 0.1$.

As expected, the error decreases as the number of random features increases. Additionally, the relative errors for the uniform distribution decrease with the dimensionality of the original data set. Conversely, with very small numbers of features (i.e., D) the relative error decreases as the dimensionality of the data increases.

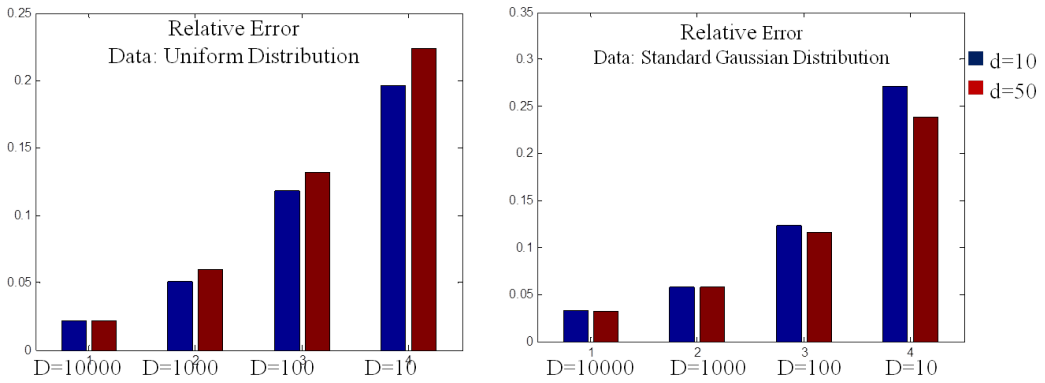

Figure 7.3: Fourier Kernel approximation errors on two datasets. d indicates the dimension of the data in the original space, and D is the size of feature vectors created via Fourier kernel approximation. The kernel function used is the Gaussian kernel: $k(x, y) = \exp^{-\frac{||x-y||_2^2}{2\sigma}}$.

We then examined the use of the Feature Belief Propagation, which uses random Fourier features methods in the context of inference on real data. Here, we used the data from molecular dynamics simulation of the Engrailed Homeodomain. We extracted the pairwise distances between the α-carbons of the protein and constructed a RKHS-embedded graphical model from the data. The dimensionality of the data was 178 variables, and we experimented with dataset of size 2500 samples. Using leave-one-out cross-validation, we learned a model from the training folds. We then randomly partitioned the variables into equal-sized sets and conditioned the models on one of the sets, and imputed the values of the remaining variables.

Figure 7.4 shows the root mean squared error (RMSE) of the imputed values, for different sizes of Fourier feature vector dimension(D). Figure 7.5 shows the average cpu time for each inference for these models. We used a Gaussian kernel with $\sigma = 0.1$ as the kernel bandwidth parameter.

As can be seen, the relative error of the modified KBP is substantially larger than the original algorithm. However, the run-times of the modified algorithm are substantially lower.

We next experimented with sub-sampling for KBP algorithm, using the Coreset selection method, and compared it with Uniform sub-sampling. Figures 7.6 and 7.7 show average RMSE,

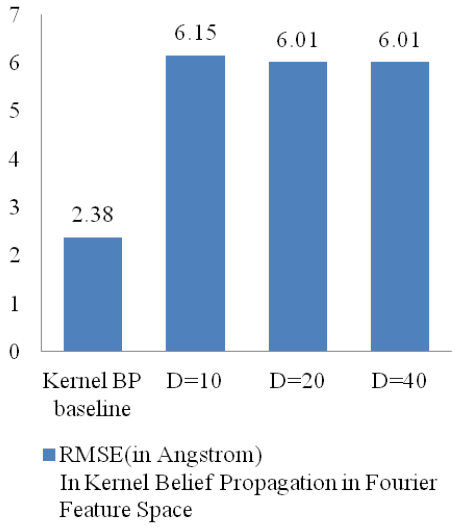

Figure 7.4: Root mean squared error for pairwise distance data of protein simulation data

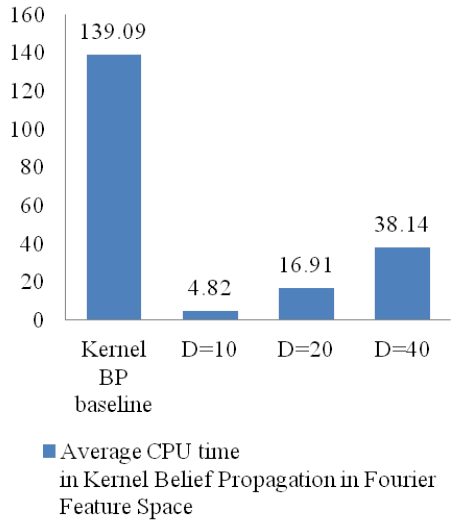

Figure 7.5: Average CPU time for pairwise distance data of protein simulation data

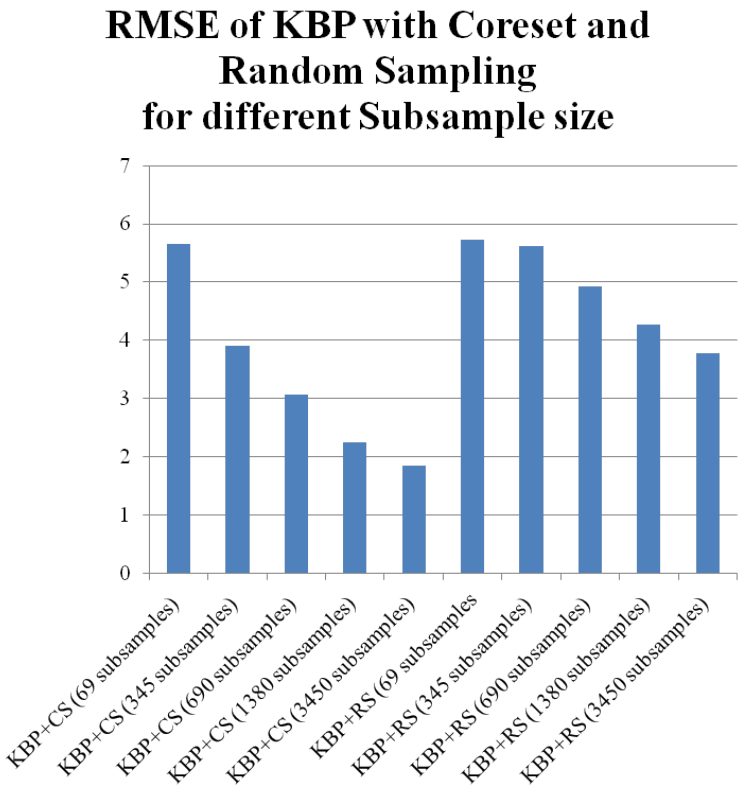

Figure 7.6: RMSE of Kernel Belief Propagation for different sub-sampling methods

and average CPU time of the sub-sampling and KBP inference combined, comparing Coreset sub-sampling and Uniform sub-sampling of the training data. Results are shown for different sizes of Coreset. and the corresponding uniformly sampled dataset with the same number of samples as the relevant Coreset, just where samples are selected from uniform distribution.

Also note that the leave-one-out cross validation experiments were done on the original dataset of size 10,000 samples, rather than the compressed dataset, so the experiments are comparable.

As we see, too much compression of the data leads to similar performance of the models. However as the size of the sub-sampled dataset is allowed to grow, Coreset sub-sampling adds more helpful samples to the training set, thus outperforming the random sub-sampling in terms of RMSE results.

The runtime of the Coreset sub-sampling is also not adversely affecting the speed of calcu-

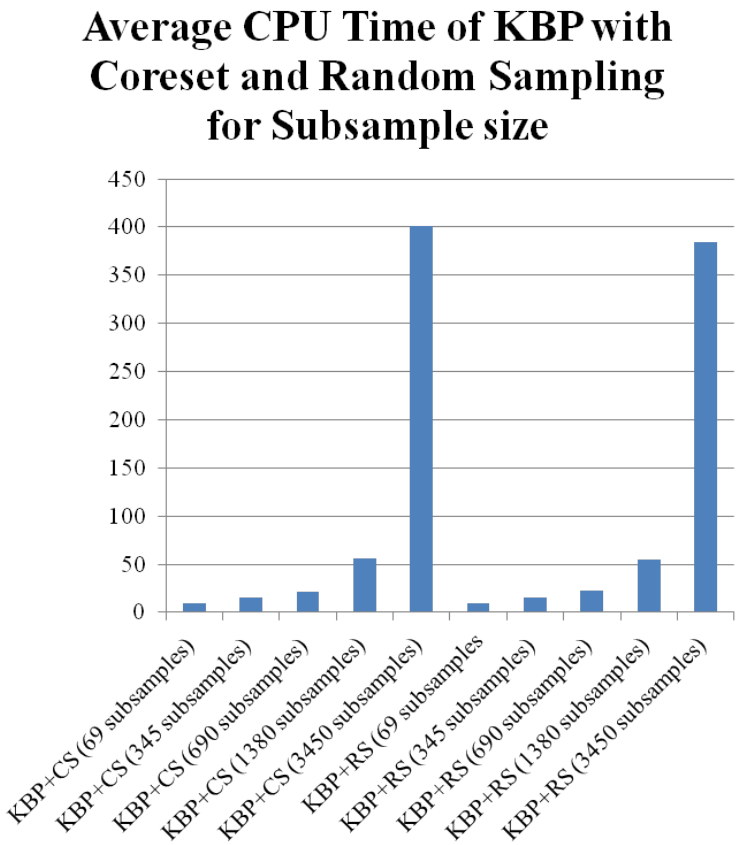

Figure 7.7: Average CPU time of Kernel Belief Propagation for different sub-sampling methods

lation compared to uniform sub-sampling, which indicates the Coreset selection to be a suitable algorithm for sub-sampling in this context.

We also ran comparison of Kernel belief propagation, $combined$ with sub-sampling both with and without the random Fourier features method. Figure 7.8 shows the RMSE of the imputed values on the protein data. For comparison, we also learned multivariate Gaussian model and Non-paranormal graphical model [45], Mixture of Gaussian and mixture of Nonparanormal as additional baselines. In case of the kernel inference, we used Gaussian kernel width of $\sigma = 0.1$.

The unmodified KBP algorithm outperforms both the Non-paranormal and the Gaussian Graphical Model, but with a substantially higher runtime. Two of the modified KBP algorithms (the one using Coreset sample selection, the other using random sample selection) perform nearly as well as the unmodified variant, with substantially reduced runtimes. The remaining modifications, which each employ random Fourier features perform worse than the others, although their runtimes are nearly as good as the Gaussian and the Non-paranormal. However, for the type of the data that we experimented with (closely connected distance data), we see that mixture of Gaussian is in fact the most accurate model, and is also among the fastest models. Still, the quality of the inference for nonparametric model is close to the best performing model, and the nonparametric models have the benefit of that the variables need not necessarily only be of the same type. This inference works for non-homogeneous variables such as mix of sequence and structure. We will discuss this as part of our future work in chapter 9.

Finally, we tested how far we can go for scaling to our big dataset of size 1,000,670 of Engrailed protein C-α pairwise distances. At this size, Feature space belief propagation and Kernel belief propagation both fail to scale, and data sub-sampling is the only solution that can scale and provide us with a solution.

We used the online version of Coreset sub-sampling, to compress the data into corset of size 3789 samples, which can easily be handled by Kernel Belief Propagation. We performed our Coreset selection algorithm on batches of size 3000, and performed compression hierarchically to generate the final Coreset. We also created a uniformly selected training data of the same size

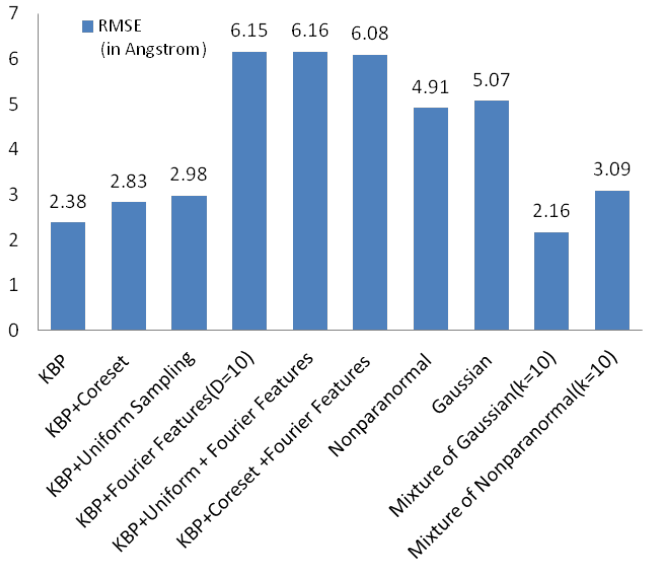

Figure 7.8: Root mean squared error for pairwise distance data of protein simulation data

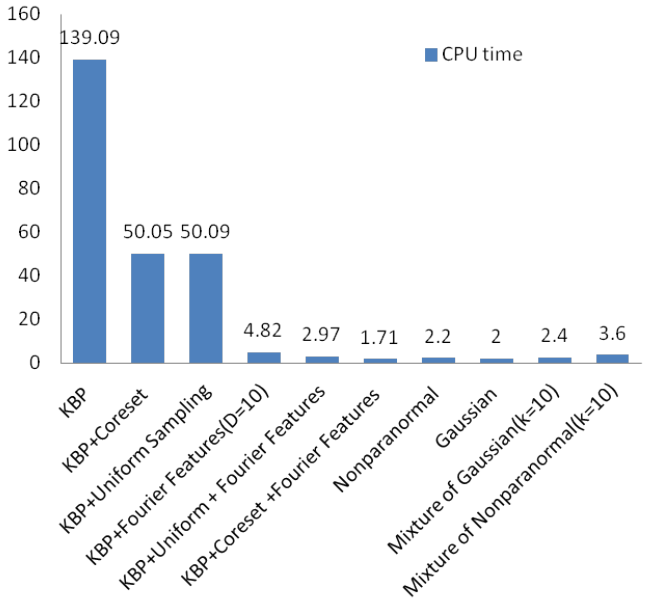

Figure 7.9: Average CPU time for pairwise distance data of protein simulation data

Coreset selection CPU time(in seconds)	Inference CPU time(in sec)	RMSE of Coreset Data(in angstrom)	RMSE of Uniform subsamples
4.5633e04	449.76	1.830	4.478

Table 7.1: CPU Time and RMSE for Experiments on Coreset Selection of the large dataset

data size	100	1000	2000	5000	10000	1,000,670
KBP (CPU time)	10.47	49.37	136.7	not scalable	not scalable	not scalable
KBP In Feature Space (CPU time)	2.05	12.48	35.24	126.08	272.38	not scalable
KBP+Coreset (CPU time)			10.08	156.00	400.91	45633

Table 7.2: CPU Time of several variations of Kernel Belief Propagation on different data sizes

(3789), and used that for the cross validation experiment. Table 7.1 shows the CPU time for this task and the root mean squared error of the leave-one-out cross validation inference tasks. Note that for the leave-one-out cross validation, we performed inference for every 100 sample, so in total the experiment tested 10,006 samples.

To wrap up the presented models, in Table 7.2 we compare the average runtime, and the scalability of the presented models for different sizes of training data(Using pairwise distances of C-α atoms). As it can be seen, Kernel belief propagation without modification takes the longest time and has the highest memory requirement. However as shown in different experiments throughout this chapter, the predictions are more accurate. If one can not afford the memory and runtime requirements of full KBP, they can either switch to Feature based belief propagation, which can scale better, however has some limitation on memory as well. The other option is to subsample the training data, and then perform KBP inference using the sub-sampled set.

7.7 Conclusions

In this chapter, we presented several variations on the Kernel Belief Propagation algorithm for RKHS-embedded graphical models. Our first variation was based on a method to approximate kernel function as a product of explicit Fourier feature vectors. We showed, for the first time, how to perform nonparametric kernel belief propagation in the feature space, and provided error analysis of the inference in this feature space. The remaining variations employed different sub-sampling schemes with, and without the Fourier feature approximation.

Our experimental results show that we can gain computational efficiency from sub-sampling, with only small increase in inference error. We also showed that using inference in Fourier feature space significantly reduces the run time of the inference. We did not observe the full effect of inference in Fourier feature space, however. Further investigation of the method and analysis of where it can be improved is part of our future work. Also, using the algorithm in the context of non-homogeneous variable sets, such as combination of sequence and structure models, is another important and exciting direction for future work, which we will discuss in the final chapter of this thesis.

Part III

Time Varying Gaussian Graphical Model

Computational structural molecular biology provides a fascinating and challenging application domain for development of time-varying graphical models. The energy landscape associated with each protein is a complicated surface in a high-dimensional space (one dimension for each conformational degree of freedom in the protein), which may contain many local minima (called *sub-states*) separated by energy barriers. Molecular Dynamics simulations are important tools that help sample this energy landscape with very high resolution, resulting in terabytes of data that span a few milliseconds of the protein dynamics.

Given the non-$i.i.d.$ nature of the data, analysis of these huge models requires time-varying graphical models, which are designed specifically for the non-independent data samples. In this part, in chapter 8 we will describe a sparse time-varying Gaussian graphical model, that we apply to analysis of protein dynamics of CypA enzyme.

Chapter 8

Time varying Gaussian Graphical Models

In this chapter, we build a time-varying, undirected Gaussian graphical model of the system's internal degrees of freedom including the statistical couplings between them. The resulting model automatically reveals the conformational sub-states visited by the simulation, as well as the transition between them.

8.1 Introduction

A system's ability to visit different sub-states is closely linked to important phenomena, including enzyme catalysis[7] and energy transduction[40]. For example, the primary sub-states associated with an enzyme might correspond to the unbound form, the enzyme-substrate complex, and the enzyme-product complex. The enzyme moves between these sub-states through *transition states*, which lie along the path(s) of least resistance over the energy barriers. Molecular Dynamics provide critical insights into these transitions.

Our method is motivated by recent advances in Molecular Dynamics simulation technologies. Until recently, MD simulations were limited to timescales on the order of several tens of nanoseconds. Today, however, the field is in the midst of a revolution, due to a number of technological advances in software (e.g., NAMD[64] and Desmond[10]), distributed computing

(e.g., Folding@Home[61]), and specialized hardware (e.g., the use of GPUs[82] and Anton[74]). Collectively, these advances are enabling MD simulations into the millisecond range. This is significant because many biological phenomena, like protein folding and catalysis, occur on μs to msec timescales.

At the same time, long timescale simulations create significant computational challenges in terms of data storage, transmission, and analysis. Long-timescale simulations can easily exceed a terabyte in size. Our method builds a compact, generative model of the data, resulting in substantial space savings. More importantly, our method makes it easier to understand the data by revealing dynamic correlations that are relevant to biological function. Algorithmically, our approach employs L1-regularization to ensure sparsity, and a kernel to ensure that the parameters change smoothly over time. Sparse models often have better generalization capabilities, while smoothly varying parameters increase the interpretability of the model.

8.2 Analysis of Molecular Dynamics Simulation Data

Molecular Dynamics simulations involve integrating Newton's laws of motion for a set of atoms. Briefly, given a set of n atomic coordinates $\mathbf{X} = \{\vec{X}_1, ..., \vec{X}_n : \vec{X}_i \in \Re^3\}$ and their corresponding velocity vectors $\mathbf{V} = \{\vec{V}_1, ..., \vec{V}_n : \vec{V}_i \in \Re^3\}$, MD updates the positions and velocities of each atom according to an energy potential. The updates are performed via numerical integration, resulting in a conformational *trajectory*. When simulating reaction pathways, as is the case in our experiments, it is customary to analyze the trajectory along the *reaction coordinate* which simply describes the progress of the simulation through the pathway.

The size of the time step for the numerical integration is normally on the order of a femtosecond (10^{-15} sec), meaning that a 1 microsecond (10^{-6} sec) simulation requires one billion integration steps. In most circumstances, every 100th to 1000th conformation is written to disc as an ordered series of *frames*. Various techniques for analyzing MD data are then applied to these frames.

Traditional methods for analyzing MD data involve monitoring changes in global statistics (e.g., the radius of gyration, root-mean squared difference from the initial conformation, total energy, etc), and identifying sub-states using techniques such as quasi-harmonic analysis[33] [41], and other Principal Components Analysis (PCA) based techniques[6]. Quasi-harmonic analysis, like all PCA-based methods, implicitly assumes that the frames are drawn from a multivariate Gaussian distribution. Our method makes the same assumption but differs from quasi-harmonic analysis in three important ways. First, PCA usually averages over time by computing a single covariance matrix over the data. Our method, in contrast, performs a time-varying analysis, giving insights into how the dynamics of the protein change in different sub-states and the transition states between them. Second, PCA projects the data onto an orthogonal basis. Our method involves no change of basis, making the resulting model easier to interpret. Third, we employ regularization when learning the parameters of our model. Regularization is a common strategy for reducing the tendency to over-fit data by, informally, penalizing overly complicated models. In this sense, regularization achieves some of the same benefits as PCA-based dimensionality reductions, which is also used to produce low-complexity models.

The use of regularization is common in Statistics and in Machine Learning, but it has only recently been applied to Molecular Dynamics data[46] [48]. Previous applications focus on the problem of learning the parameters of force-fields for coarse-grained models, and rely on a Bayesian prior, in the form of inverse-Wishart distribution[46], or a Gaussian distribution[48] for regularization. Our method solves a completely different problem (modeling angular deviations of the all-atom model) and uses a different regularization scheme. In particular, we use L1 regularization, which is equivalent to using a Laplace prior. The use of L1 regularization is particularly appealing due to its theoretical properties of consistency — given enough data, the learning procedure learns the true model, and high statistical efficiency — the number of samples needed to achieve this guarantee is small.

8.3 Regularized Time-Varying Gaussian Graphical Models

Zhou et. al. [95] define a weighted log likelihood for time-varying Gaussian graphical models, as follows: Let $D^{1..T}_{(1),..(m)}$ be the set of training data, where each $D^t_{(i)}$ is a sample represented by n variables. For instance, in our modeling of MD data, each $D^t_{(i)}$ is a protein conformation. The time varying GGM parameter estimation algorithm extends the stationary GGM parameter learning as follows:'

$$\Sigma^{-1}(t) = \arg\max_{X \succ 0} \log|X| - trace(S(t)X) - \lambda\|X\|_1$$

Where, $S(t)$ is the $weighted\ covariance\ matrix$, and is calculated as follows:

$$S(t) = \frac{\sum_{s=1}^{T}\sum_{i=1}^{m} w_{st}(D_i^{(s)} - \mu)(D_i^{(s)} - \mu)^T}{\sum_{s=1}^{T} w_{st}}$$

The weights w_{st} are defined by a symmetric nonnegative kernel function.

Choice of the Kernel Function

The choice of the kernel function will let the model select for its specificity. A kernel with a larger span will push the time varying model to be less sensitive to abrupt changes in the network and capture the slower and more robust behaviors. On the other hand, as the kernel function span decreases, the time varying will be able to capture more short term patterns of interaction.

In our experiments we used a kernel from a triangular family which spans over 5 simulations before and after the experiment (Figure 8.1).

Experimenting with other Kernel families, and different kernel spans in an important part of our future work, which we will mention in the final section of this chapter.

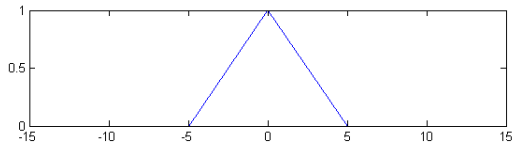

Figure 8.1: The Kernel functions of triangular family used in our experiment. $K = 1 - \frac{|x|}{5} * 1_{\{|x|<5\}}$

8.4 Convex Optimization for Parameter Estimation of Regularized Time Varying GGM

We use Block Coordinate Descent Algorithm to solve the stationary and time varying problems. This method has been proposed by Banerjee et. al.[4], and proceeds by forming the dual for the optimization case, and applying block coordinate descent to the dual form.

Recall that the primal form of both the stationary and time varying case is as follows:

$$\Sigma^{-1} = \arg\max_{X \succ 0} \log|X| - trace(SX) - \lambda\|X\|_1$$

To take the dual, we first rewrite the L1-norm as:

$$\|X\|_1 = \max_{\|U\|_\infty \leq 1} trace(XU)$$

where $\|U\|_\infty$ denotes the maximum absolute value element of the matrix U. Given this change of formulation, we can rewrite the primal form of the problem as:

$$\Sigma^{-1} = \max_{X \succ 0} \min_{\|U\|_\infty \leq \lambda} \log|X| - trace(X, S + U)$$

Thus, the optimal Σ^{-1} is the one that maximizes the worst case log likelihood, over all additive perturbations of the covariance matrix, S. Next, to obtain the dual form, we exchange the min and max, and the inner max objective function can now be solved analytically taking the

gradient and setting it to zero. This results in the new form of the objective function:

$$U^* = \min_{\|U\|_\infty \leq \lambda} -\log|S+U| - n$$

where n is the number of features in each sample. Once we solve this problem, the optimal Σ^{-1} can be computed as $\Sigma^{-1} = (S+U^*)^{-1}$.

Performing one last change of variables $W = S + U$, and forming the dual of the problem will bring us to the final form of our objective:

$$\Sigma^* = \max\{\log|W| : \|W - S\|_\infty \leq \lambda\}$$

This problem is smooth and convex, and for small values of n it can be solved by standard optimization techniques like interior point method. For larger values of n the interior point method becomes too inefficient, and another method, called Block Coordinate Descent can be used instead[4].

Block Coordinate Descent

The Block Coordinate Descent algorithm works as follows. For any matrix A, let $A_{\backslash k \backslash j}$ denote the matrix produced by removing column k and row j of the matrix. Let A_j also denote the column j, with diagonal element A_{jj} removed.

Block Coordinate Descent algorithm proceeds by optimizing one row and one column of the variable matrix W at a time. The algorithm iteratively optimizes all columns until a convergence criteria is met. The algorithm is summarized below:

Initialize $W^{(0)} := S + \lambda I$

Repeat until convergence

for $j = 1, \ldots n$ **do**

$y^* = \arg\min_y \{y^T W^{(j-1)}_{\setminus j \setminus j} y : \|y - S_j\|_\infty \leq \lambda\}$, Where $W^{(j-1)}$ denotes the current iterate.

Update $W^{(j)}$ as $W^{(j-1)}$ with column/row W_j replaced by y^*.

Let $W^{(0)} = W^{(n)}$

Test for convergence when the $W^{(0)}$ satisfies: $trace((W^{(0)})^{-1}S) - n + \lambda\|(W^{(0)})^{-1}\|_1 \leq \epsilon$.

end for

The $W^{(j)}$s produced in each step are strictly positive definite. This property is important because the dual problem estimates the covariance matrix Σ, rather than the inverse covariance matrix. The network conditional dependencies which we are interested in are encoded in the inverse covariance matrix, Σ^{-1}, so the strictly positivity of $W^{(j)}$ will guarantee that the optimum Σ will be reversible, and that we can compute the final answer Σ^{-1} from the $W^{(j)}$.

The time complexity of this algorithm has also been estimated to be $O(n^{4.5}/\epsilon)$[4], when converging to ϵ suboptimal solution. This complexity is better than $O(n^6/\log(\frac{1}{\epsilon}))$, which would have been achieved using the interior point method on the dual form[86].

We used this algorithm in our experiments, to estimate a L1-Regularized Time-Varying Gaussian Graphical Model on the MD simulation data. The experimental conditions, model selection, and the result of the experiments will be presented in the next section.

8.5 Results

We applied our method to three simulations of the human form of the enzyme *cyclophilin A (CypA)*. CypA isomerizes the ω bond of its substrate and it is an important receptor for several immuno-suppresive drugs and HIV infection. Our three simulations correspond to three different substrates: (i) The hexa-peptide His-Ala-Gly-Pro-Ile-Ala from the HIV-1 capsid protein (PDB ID: 1AWQ); (ii) the dipeptide Ala-Pro (PDB ID: 2CYH); and (iii) the tetra-peptide Ala-Ala-Pro-

Phe (PDB ID: 1RMH).

Previous studies have identified a set of 25 highly conserved residues in the cyclophilin family[3]. In particular, residues P30, T32, N35, F36, Y48, F53, H54, R55, I57, F60, M61, Q63, G65, F83, E86, L98, M100, T107, Q111, F112, F113, I114, L122, H126, F129 are all highly conserved. Experimental work[9] and MD simulations[2, 3] have also implicated these residues as forming a network that influences the substrate isomerization process. Significantly, this network extends from the flexible surface regions of the protein to the active site residues of the enzyme (residues R55, F60, M61, N102, A103, F113, L122, and H126). The previous studies identified this network by examining atomic positional fluctuations and the correlations between them. In contrast, our study focuses on the angular correlations, as revealed by our algorithm. Positional fluctuations are ultimately caused by the angular fluctuations, so our study is complementary to the previous work.

8.5.1 Simulations

The details of the three MD data sets have been reported previously[3]. Briefly, each data set is generated by performing 39 independent simulations in explicit solvent along the reaction coordinate. The first simulation starts with the substrate's ω angle at 180° (i.e., *trans*) from which 400 frames are extracted, corresponding to 400 ps of simulated time. The second simulation starts with the substrate's ω angle at 175°, from which another 400 frames are obtained. Subsequent simulations increment the ω by 5° until the 0° (i.e., *cis*) configuration is reached. Each frame corresponds to one protein conformation, and is represented as a vector of dihedral angles – one for each variable. For each residue there is a variable for each of ϕ, ψ, ω, and the side chain angles χ(between 0 and 4 variables, depending on residue type). The time-varying graphical models are learned from the resulting 15,600 frames.

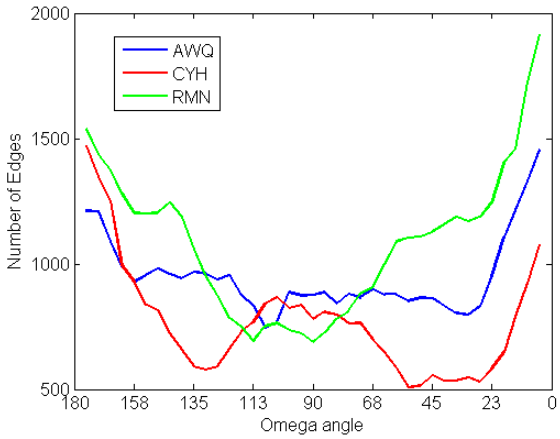

Figure 8.2: Edge Density Along Reaction Coordinate. The number of edges learned from the three MD simulations of CypA in complex with three substrates (AWQ, CYH, and RMH) are plotted as a function of the ω angle. AWQ is the largest substrate, CYH is the smallest substrate.

8.5.2 Model Selection

We used the imputation method, as previously mentioned in 3.7.1 to select the regularization penalty, λ. The value $\lambda = 1,000$ was found to be the smallest value consistently giving zero edges across all permuted data sets. In our experiments we used a more stringent value ($\lambda = 5,000$) in order to ensure that our edges don't reflect spurious correlations. This conservative choice reflects the importance of not including any spurious correlations in our final results.

8.5.3 Edge Density Along Reaction Coordinate

We define *edge density* to be the number of recovered edges, divided by total number of possible edges. As previously mentioned, each data sets comprises 39 individual simulations. The learning algorithm identifies a set of edges in each simulation, employing a kernel to ensure smoothly varying sets of edges. Figure 8.2 plots the number of edges for data set along the reaction coordinate. Qualitatively, the number of edges decreases until the transition state, and then rises for each substrate. The three substrates, however, also show significant differences in the number of local minima, the location and width of the minima, and the minimum number of edges.

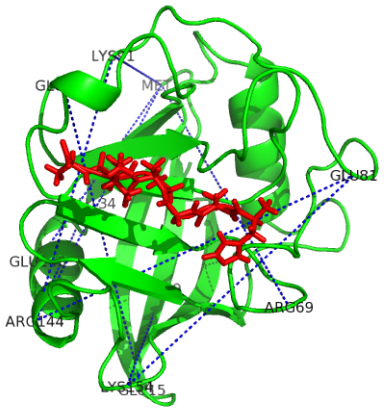

Figure 8.3: Top 10 Persistent Edges. For simplicity, only the top 10 couplings are shown.

Differences in the number and width of minima might be suggestive of differences in the kinetics of the reactions, although we have not been able to identify any published data on the isomerization rates for these specific substrates. We note, however, that the magnitude of the minima is correlated with the size of the substrate. In particular, the minimum value of the curve labeled AWQ (the largest substrate) is larger than the minimum value of the curve labeled RMH (the second largest substrate) which, in turn, is larger than the minimum value of the curve labeled CYH (the smallest substrate). Edge density corresponds to the total amount of coupling in the system. Thus, these results suggest that when approaching the transition state the angles tend to decouple. At the same time, the dependency on size suggest that larger substrates may require more coupling than smaller ones in order to pass through the transition state of the reaction coordinate.

8.5.4 Persistent, Conserved Couplings

We next examined the set of edges to get a sense for which couplings are persistent. That is, edges that are observed across the entire reaction coordinate and in all three simulations. We computed $P^a_{i,j}$, the probability that edge (i, j) exists in substrate a. Then, we computed the product $P_{i,j} = P^a_{i,j} * P^b_{i,j} * P^c_{i,j}$ as a measure of persistence. We then identified the edges where

$P_{i,j} > 0.5$, yielding a total of 73 edges (out of $\binom{165}{2} = 13,530$ possible edges). The top 10 of these edges are shown in Figure 8.3. Notice that the edges span large distances. Each of the top 10 edges relates how distal control could occur within CypA; these edges typically connect one network region with the other. For example, region 13-15 is connected to 146-152 which connect to farther off regions including 68-76 and 78-86.

8.5.5 Couplings to the Active Site and Substrate

According to our analysis of the dihedral angular fluctuations, the set of residues most strongly coupled to the substrate are residues 1, 13, 14, 125, 147, and 157. None of these residues is in the active site (residues 55, 60, 61, 102, 103, 113, 122, 126), although residue 125 is sequentially adjacent to an active site residue. The set of resides most strongly coupled to the active site include residues 1, 9, 13, 14, 81, 86, 91, 120, 125, 142, 151, 154, and 165. Of these, only residue 86 is among the previously cited list of highly conserved residues. Thus, the conservation of angular deviations observed across substrates is distinct from the residue conservation within the family. We can conclude that the conservation of angular deviation is an inherent feature of the structure of the protein, as opposed to its sequence.

8.5.6 Transient, Conserved Couplings

Next, we identified the edges that are found across all three substrates, but are only found in one segment of the reaction coordinate. To do this we first partitioned the reaction coordinate into three parts: (i) $\omega \in [180, 120)$; (ii) $\omega \in [120, 60)$; and (iii) $\omega \in [60, 0]$, which we will refer to as the *trans*, *transition*, and *cis* states, respectively. We then identified the edges that occur exclusively in the *trans* state, those occurring exclusively in the transition state, and those occurring exclusively in the *cis* state. Four such edges were found for the *trans* state: (49,81), (1,143), (143, 144), and (1 154); five edges were found for the transition state: (9,157),(82,140), (9,157), (91, 157), and (144, 157); and sixty one edges were found for the *cis* state. A subset of

these edges are shown in Figure 8.4. The coupling of the edges reveal clues about how couplings between network regions varies with the reaction coordinate. In the trans state one can see couplings between network regions 142-156 and 78-86, while in the cis state there are couplings between network regions 13-15 and 89-93.

8.5.7 Substrate-Specific Couplings

Finally, we identified couplings that are specific to each substrate. As in the previous section, we partitioned the reaction coordinate into the *trans*, transition, and *cis* states. We then identified the edges that occur exclusively in the AWQ substrate, those occurring exclusively in the CYH substrate, and those occurring exclusively in the RMH substrate.

We found 62, 8, and 24 such edges, respectively. A subset of those edges are shown in Figure 8.5. Looking at the couplings one can notice that the edges lie on the network regions (13-15, 68-74, 78-86 and 146-152). However, the coupled residues change from substrate to substrate which implies a certain specificity in the dynamics.

8.6 Discussion and Summary

Molecular Dynamics simulations provide important insights into the role that conformational fluctuations play in biological function. Unfortunately, the resulting data sets are both massive and complex. Previous methods for analyzing these data are primarily based on dimensionality reduction techniques, like Principal Components Analysis, which involves averaging over the entire data set and projects the data into a new basis. Our method, in contrast, builds a time-varying graphical model of the data, thus preserving the temporal nature of the data, and presenting data in its original space. Moreover, our methods uses L1 regularization when learning leading to easily interpretable models. The use of L1 regularization also confers desirable theoretical properties in terms of consistency and statistical efficiency. In particular, given enough data, our

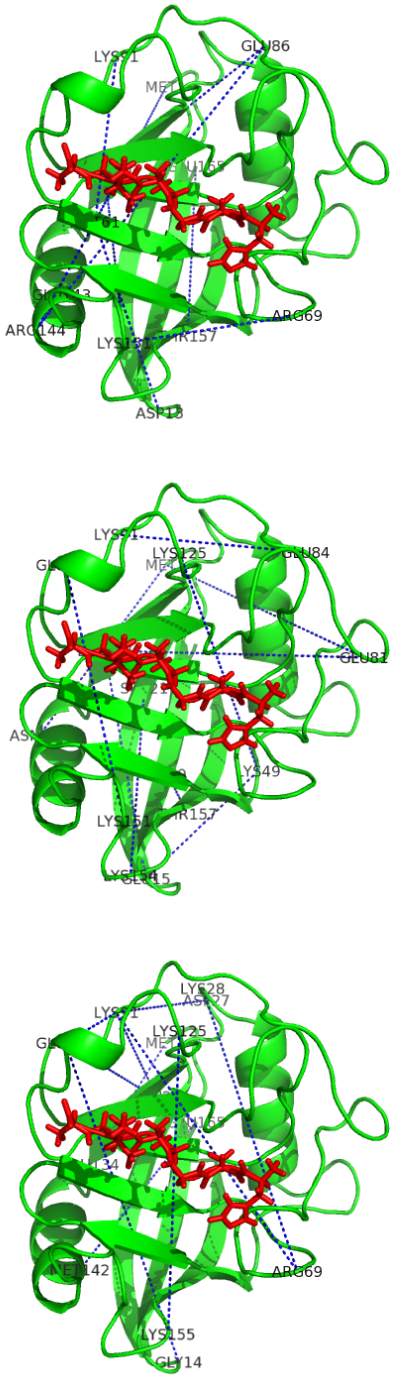

Figure 8.4: Transient Edges. The set of edges seen exclusively in the *trans* (top), transition (middle), and *cis* (bottom) states, respectively. For simplicity, only the top 10 couplings are shown.

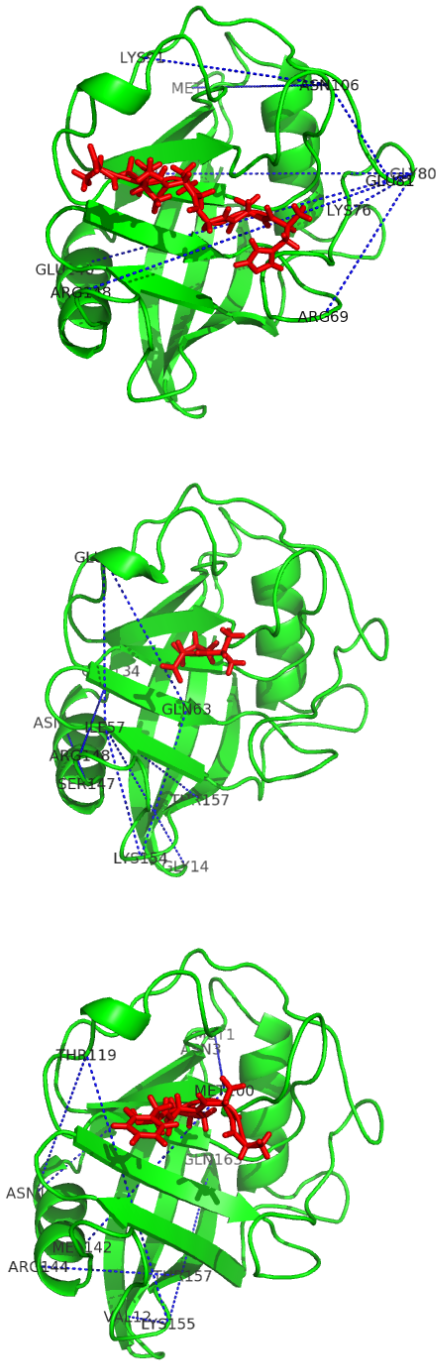

Figure 8.5: Substrate-specific Edges. The set of edges seen exclusively in the AWQ (top) CHY (middle), and RMH (bottom) substrates. For simplicity, only the top 10 couplings are shown.

method will learn the 'true' model, and the number of samples needed to achieve this guarantee is small.

We demonstrated our method on three simulations of Cyclophilin A, revealing both similarities and differences across the substrates. Coupling tends to first decrease and then increase along the reaction coordinate. As observed from Fig. 8.2, the variation in simulations with longer peptides (1AWQ and 1RMH) show similar behavior in and around the transition state, while 1CYH, with the dipeptide shows an increase in the number of edges. This difference is perhaps a result of the fact that dipeptides such as Ala-Pro can potentially act as inhibitors for CypA[94]. Although, the significance of these differences cannot be discussed in the light of mechanistic behavior in CypA, the ability of our method to detect subtle, yet important changes during the course of such simulations is in itself a valuable tool for biologists.

There is also evidence that there are both state-specific and substrate-specific couplings, all of which are automatically discovered by the method. We have discovered that over the course of the reaction, the network regions as identified by previous work[1] couple directly to the active site residues (see Fig. 8.4). The method is also able to pick out subtle changes in the dynamics as seen by the edges that appear in substrate-specific couplings (see Fig. 8.5). These differences are present exactly on the network regions, implying that the alteration in the dynamics of these regions may be responsible for catalysis with respect to specific substrates. An interesting direction of further research is to study how presence of CypA inhibitors such as cyclosporin can alter the dynamics in these network regions to understand the mechanistic underpinnings of CypA function.

Currently, our model assumes that the underlying distribution is multivariate Gaussian. As we saw in the previous chapter, there are other parametric, semi-parametric, and nonparametric graphical models that provide a better generative model of the data. In the future direction we will describe possible extensions to the model, to take advantage of more accurate graphical models in the time varying framework.

Also, our experiments were limited in that they only examined a symmetric fixed length

kernel. In applications such as protein folding trajectory modeling, the probability of sampling the protein in each sub-state is inversely correlated with the free energy of the sub-state, and any MD simulation of the data contains different spans of samples, each from a different local minima of the folding trajectory. Another interesting future work direction is to adjust the kernel length accordingly, using nonparametric clustering via Dirichlet processes, befor optimizing the weighted log likelihood.

Chapter 9

Conclusions and Future Works

In this thesis, we presented a hierarchy of graphical models, capable of handling challenges specifically present in modeling of Molecular Dynamics(MD) simulation data of Protein structures.

First, we presented a unified framework to estimate and use von Mises graphical models, where variables are distributed according to von Mises distribution, designed for angular variables. To estimate the structure and parameters of these models we optimized regularized pseudo likelihood of the training data, and proved the consistency of our estimation method. We also developed an inference based on nonparametric belief propagation for von Mises graphical models. Our results showed that compared to Gaussian graphical models, von Mises graphical models perform better on synthetic and real MD simulation data.

Second, we extended our von Mises graphical models to mixture of von Mises graphical models, and developed weighted Expectation Maximization algorithm, using parameter estimation and inference techniques that we had developed for single von Mises graphical models. Our experiments on backbone and side-chain angles of arginine amino acid over 7K non-redundant protein structures confirmed that mixture of von Mises model has better imputation and likelihood scores for the data, out performing Gaussian, mixture of Gaussian and von Mises models, significantly.

Third, we used nonparametric graphical models, which have higher representational power, to model our data. To be able to perform inference in practical domains, we need sparse structure learning prior to the inference. So we provided comparison of several state of the art structure learning methods, which included Neighborhood Selection, Nonparametric Tree structure learning, and Kernel Based Normalized Cross Covariance Operator (NOCCO). We compared them over synthetic data and in the context of inference for real protein MD simulation data. Our experiments showed that Neighborhood selection method has significant advantages in terms of accuracy of the structure learning, scalability.

Fourth, we proposed inference in explicit feature space, instead of kernel space, to deal with scalability issue of the reproducing kernel Hilbert space inference, when size of the data is large. We also used Fourier random feature approximation of Gaussian kernel, to perform the inference for Gaussian kernels as well. Our experiments showed significant improvement in terms of speed of inference and memory requirement, with a cost of decrease in accuracy of the results as measured in root mean squared error of imputation.

We also described the combination of Coreset sub-sampling with nonparametric belief propagation. Coresets provide a sub-sampling of the data, such that the samples are from both high density and low density regions of the input space, and therefore models trained on the Coreset exhibit enough robustness, compared to uniform sub-sampling, which tends to miss samples from lower density regions. Our experiments showed that using the sub-sampled data improves the scalability of the inference and also has smaller adverse effect on quality of the predictions of the model.

Finally, we focused on developing time varying graphical models, for situations whether the molecular dynamics simulation data is not identically distributed. In particular we presented our $sparse$ time varying Gaussian graphical model, which uses a smoothing kernel to interpolate samples from different time windows. Our result over CypA molecular simulation data showed such models are capable of discovering information from MD data which has been validated experimentally.

The hierarchy of graphical models developed and used in this thesis provide a trade-off between representative power as well as generalization power and scalability. Our best results for modeling angular distribution was achieved when we used Mixture of von Mises graphical models, which are specific for angular variables. This means that a model specifically developed for the problem domain of interest could achieve best result. However, we saw that nonparametric graphical models were more general, and simply by defining appropriate kernels, we can handle different variable types simultaneously and easily. When our data contains different types of variable, such models are our best solution. The price of this representational power, as we saw, was scalability of the inference. However as we discussed several techniques could improve the scalability of such models as well. Also, when the problem is inherently time varying, developing sequence of local graphical models will let us take advantage of scalability advantage of learning smaller models along with the representational power that can be gained from the chain of time-varying graphical models.

Table 9.1 summarizes these findings. N is the number of samples, d is the dimensionality of each sample point in the original space, K is the number of mixing components in each mixture model. δ is the time complexity of calculating trigonometric moments in case of von Mises models, and D is the dimension of approximated Fourier features in the Fourier based feature belief propagation. Finally M is the number of data-points sub-sampled by core-set selection method prior to Kernel Belief propagation. As described in throughout this thesis, according to the availability of memory and runtime resources, and based on the dimensionality of the data, size of the training data and desired complexity of the model (i.e. in terms of number of mixing components), one should select the appropriate model for their application.

This thesis explored many ideas, however there are many more directions which are left to explore in the future.

Model	Error on angular data (Section 6.5.2)	Error on pairwise distance data (section 7.6)	Training runtime	Inference runtime	Training memory requirement	Inference memory requirement	Can handle inhomogeneous variables?
Gaussian	8.46	5.07	$O(Nd^2 + d^3)$	$O(d^3)$	$O(d^2)$	$O(d^2)$	no
Mixture of Gaussian(K mixing components)	8.21	2.16(best)	$O(kNd^2 + Kd^3)$	$O(kd^3)$	$O(kd^2)$	$O(kd^2)$	no
NonParanormal	8.43	4.91	$O(Nlog(N) + Nd^2 + d^3)$	$O(log(N) + d^3)$	$O(N + d^2)$	$O(N + d^2)$	no
Mixture of Nonparanormal(K mixing components)	7.63	3.09	$O(Nlog(N) + KNd^2 + Kd^3)$	$O(log(N) + Kd^3)$	$O(N + Kd^2)$	$O(N + Kd^2)$	no
von Mises	6.93 (second best)	n/	$O(Nd^3)$	$O(N^2(\delta + d))$	$O(d^2)$	$O(d^2)$	no
Mixture of von Mises(K mixing components)	5.92 (best)	n/	$O(KNd^3)$	$O(KN^2(\delta + d))$	$O(Kd^2)$	$O(Kd^2)$	no
Kernel Belief Propagation	7.3(third best)	2.38(Second best)	$O(Nd^2 + d^3 + N^2d)$	$O(d^3N^3)$	$O(N^2d)$	$O(N^2d+Nd^2)$	yes
Kernel Belief Propagation + Fourier Feature	n/	6.15	$O(Nd^2 + d^3 + N^2d)$	$O(d^3DN^2)$	$O(DNd)$	$O(NdD + Dd^2)$	no
Kernel Belief Propagation + training data subsampling (M subsamples)	n/	2.83(third best)	$O(Md^2 + d^3 + M^2d)$	$O(d^3M^3)$	$O(M^2d)$	$O(M^2d + Md^2)$	yes

Table 9.1: Comparison of graphical models developed in this thesis

9.1 Future Directions

There are several areas for exploration in the future.

9.1.1 Feature Space Belief Propagation

In Chapter 7, we presented how to use the feature approximation of Gaussian Kernel in Feature space embedded belief propagation, to improve the runtime of the method. In particular we used random Fourier feature approximation for Gaussian kernel. In applications where we deal with angular variables, it makes sense to perform this feature approximation for other kernels including trigonometric kernel as discussed in section 6.5.2. One can perform feature approximation, using Fourier transformation of the trigonometric kernel, and calculation of this transformation remains an interesting part of the future work.

9.1.2 Inhomogeneous Variables in Nonparametric Models

As we mentioned in section 7.7, the main benefit of the nonparametric inference methods is the ability of these models to seamlessly handle multiple variable types, (including Angular and non-angular, discrete and continuous, and even structured variables) in the same frame work, provided that one can define appropriate kernel matrices for the variable type. This issue has been a large challenge in hybrid graphical models so far, and as part of the future work we recommend that the joint sequence and structure models of protein be modeled, through this inference model. In particular the joint sequence/structure models can be directly useful for drug design, which is a very important and challenging application.

9.1.3 Nystrom method and Comparison to Incomplete Cholesky Decomposition for kernel approximation

In chapter 7 we proposed using nonparametric belief propagation in feature space, to reduce runtime and memory requirement of the kernel inference. Another approach that has been tried before by Song et. al. [79], is Incomplete Cholesky Decomposition(ICD). Similar to ICD, one can also perform Nystrom kernel approximation[59], which is another low rank approximation method for decomposing the kernel into smaller units, which reduce the total runtime and memory requirements of the message update and belief propagation. In our initial investigation, we derived the message passing update relations when one uses Nystrom kernel approximation, however implementation and comparison of the full kernel belief propagation using the Nystrom method will be part of future work.

9.1.4 Integration into Graph lab

Currently our experiments and implementations are done under Matlab environment. Recently, extensive effort has been spent on implementing inference methods in graphical models in parallel, where specific graph-cut algorithms has been designed to make the inference methods as fast and scalable as possible. One example of such systems is GraphLab[47], which implements several methods including basic Kernel Belief Propagation, and takes advantage of Map-reduce paradigm as well. As part of future work, integration of Feature space kernel belief propagation in such models is recommended, where we can see the true effect of the scalability methods.

9.1.5 Time Varying Graphical Models

Currently, we estimated time varying graphical models based on fixed-length symmetric kernel, to optimize the weighted log likelihood. After the models are learned we can use linkage clustering algorithm to cluster different graphical models learned for different time frames. Then we

can build a state-transition matrix on top of the clusters, to be able to understand the energy landscape. The number of our clusters are selected heuristically, however, based on the assumption that the height of energy barriers in protein folding trajectory is proportional to the number of samples drawn from the model[69].

As part of the future work, one can perform the clustering of the data non-parametrically, based on Dirichlet process (DP) models [15], so as to solve the issue of model selection.

Also, the local graphical model that is estimated for each span of the data can become more powerful, by using non-paranormal graphical models as reviewed in chapter 5.1, instead of Gaussian graphical models. The non-paranormal graphical model have the benefit that they can be optimized via log likelihood analytically, while they have the advantage that the data is in feature space, and thus, we are able to model more sophisticated graphical models.

Bibliography

[1] P. K. Agarwal. Computational studies of the mechanism of cis/trans isomerization in hiv-1 catalyzed by cyclophilin a. *Proteins: Struct. Funct. Bioinform.*, 56:449–463, 2004. 8.6

[2] P. K. Agarwal. Cis/trans isomerization in hiv-1 capsid protein catalyzed by cyclophilin a: Insights from computational and theoretical studies. *Proteins: Struct., Funct., Bioinformatics*, 56:449–463, 2004. 8.5

[3] P. K. Agarwal, A. Geist, and A. Gorin. Protein dynamics and enzymatic catalysis: Investigating the peptidyl-prolyl cis/trans isomerization activity of cyclophilin a. *Biochemistry*, 43:10605–10618, 2004. 8.5, 8.5.1

[4] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008. ISSN 1532-4435. 8.4, 8.4

[5] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *J. Mach. Learn. Res.*, 9:485–516, June 2008. ISSN 1532-4435. 1, 2.2.2, 5.1

[6] H. J. C Berendsen and S. Hayward. Collective protein dynamics in relation to function. *Current Opinion in Structural Biology*, 10(2):165–169, 2000. 8.2

[7] D.D. Boehr, D. McElheny, H.J. Dyson, and P.E. Wright. The dynamic energy landscape of dihydrofolate reductase catalysis. *Science*, 313(5793):1638–1642, 2006. 8.1

[8] Wouter Boomsma, Kanti V. Mardia, Charles C. Taylor, Jesper Ferkinghoff-Borg, Anders Krogh, and Thomas Hamelryck. A generative, probabilistic model of local protein structure. *Proceedings of the National Academy of Sciences*, 105(26):8932–8937, 2008. doi: 10.1073/pnas.0801715105. 2.2.3, 4.2

[9] Daryl A. Bosco, Elan Z. Eisenmesser, Susan Pochapsky, Wesley I. Sundquist, and Dorothee Kern. Catalysis of cis/trans isomerization in native hiv-1 capsid by human cyclophilin a. *Proc. Natl. Acad. Sci. USA*, 99(8):5247–5252, 2002. 8.5

[10] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, and D. E. Shaw. Scalable algorithms for molecular dynamics simulations on commodity clusters. *SC Conference*, 0:43, 2006. doi: http://doi.ieeecomputersociety.org/10.1109/SC.2006.54. 8.1

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. 3.4.3

[12] Ernst Breitenberger. Analogues of the normal distribution on the circle and the sphere. *Biometrika*, 50(1/2):pp. 81–88, 1963. ISSN 00063444. URL http://www.jstor.org/stable/2333749. 2.2.3

[13] John-Marc Chandonia, Gary Hon, Nigel S Walker, Loredana Lo Conte, Patrice Koehl, Michael Levitt, and Steven E Brenner. The astral compendium in 2004. *Nucleic acids research*, 32(suppl 1):D189–D192, 2004. 4.3.1

[14] Myung Jin Choi, Vincent Y. F. Tan, Animashree Anandkumar, and Alan S. Willsky. Learning latent tree graphical models. *J. Mach. Learn. Res.*, 12:1771–1812, July 2011. ISSN 1532-4435. 5.3.4, 6, 6.2

[15] J B Macqueen D Blackwell. Ferguson distributions via polya urn schemes, 1973. 9.1.5

[16] Hal Daum'e III. From zero to reproducing kernel hilbert spaces in twelve pages or less. February 2004. 5.3.2

[17] Angela V. DElia, Gianluca Tell, Igor Paron, Lucia Pellizzari, Renata Lonigro, and Giuseppe Damante. Missense mutations of human homeoboxes: A review. *Human Mutation*, 18(5): 361–374, 2001. ISSN 1098-1004. doi: 10.1002/humu.1207. URL http://dx.doi.org/10.1002/humu.1207. 3.7.2

[18] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39 (1), 1977. 4.1

[19] Joshua Dillon and Guy Lebanon. Statistical and computational tradeoffs in stochastic composite likelihood. 2009. 3.4.2

[20] Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast monte carlo algorithms for matrices i: Approximating matrix multiplication. Technical report, SIAM Journal on Computing, 2004. 7.5

[21] Michael Feig, John Karanicolas, and Charles L Brooks III. Mmtsb tool set: enhanced sampling and multiscale modeling methods for applications in structural biology. *Journal of Molecular Graphics and Modelling*, 22(5):377–395, 2004. 4.3.1

[22] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. pages 2142–2150, 2011. URL http://books.nips.cc/papers/files/nips24/NIPS2011_1186.pdf. 4.2, 7, 7.4, 7.4

[23] N.I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1993. 1, 2.2.3

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008. doi: 10.1093/biostatistics/kxm045. URL http://biostatistics.oxfordjournals.org/content/9/3/432.abstract. 1, 2.2.2

[25] Kenji Fukumizu, Arthur Gretton, Bernhard Scholkopf, et al. Kernel measures of conditional dependence. 2007. 6, 6.3, 6.4

[26] W J Gehring, M Affolter, and T Burglin. Homeodomain proteins. *Annual Review of Biochemistry*, 63(1):487–526, 1994. doi: 10.1146/annurev.bi.63.070194.002415.

URL http://www.annualreviews.org/doi/abs/10.1146/annurev.bi.63.070194.002415. 3.7.2

[27] Walter J. Gehring, Yan Qiu Qian, Martin Billeter, Katsuo Furukubo-Tokunaga, Alexander F. Schier, Diana Resendez-Perez, Markus Affolter, Gottfried Otting, and Kurt Wuthrich. Homeodomain-dna recognition. *Cell*, 78(2):211 – 223, 1994. ISSN 0092-8674. doi: 10.1016/0092-8674(94)90292-5. URL http://www.sciencedirect.com/science/article/pii/0092867494902925. 3.7.2

[28] Tim Harder, Wouter Boomsma, Martin Paluszewski, Jes Frellsen, Kristoffer E. Johansson, and Thomas Hamelryck. Beyond rotamers: a generative, probabilistic model of side chains in proteins. *BMC Bioinformatics*, 11:306, 2010. 2.2.1

[29] Gareth Heughes. *Multivariate and time series models for circular data with applications to protein conformational angles*. PhD Thesis, Department of Statistics, University of Leeds. 2.2.3, 3.2, 3.3

[30] Holger Hofling and Robert Tibshirani. Estimation of sparse binary pairwise markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10:883–906, April 2009. 3.4.3

[31] Alexander Ihler and David McAllester. Particle belief propagation. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS) 2009*, pages 256–263, Clearwater Beach, Florida, 2009. JMLR: WCP 5. 2.1

[32] Roland L. Dunbrack Jr and Martin Karplus. Backbone-dependent rotamer library for proteins application to side-chain prediction. *Journal of Molecular Biology*, 230(2):543 – 574, 1993. ISSN 0022-2836. doi: 10.1006/jmbi.1993.1170. 2.2.1

[33] M. Karplus and J. N. Kushick. Method for estimating the configurational entropy of macromolecules. *Macromolecules*, 14(2):325–332, 1981. 8.2

[34] M. Karplus and J. A. McCammon. Molecular dynamics simulations of biomolecules. *Nat. Struct. Biol.*, 9:646–652, 2002. 1

[35] Jr. Kruskal, Joseph B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):pp. 48–50, 1956. ISSN 00029939. URL http://www.jstor.org/stable/2033241. 5.2, 5.3.4

[36] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nystrom method. *J. Mach. Learn. Res.*, 98888:981–1006, June 2012. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2343676.2343678. 7.4

[37] J. Lafferty and L. Wasserman. Rodeo: Sparse, greedy nonparametric regression. *Annual of Statistics*, 36(1):28–63, 2008. 5.2, 6, 6.4

[38] J. Lafferty, H. Liu, and L. Wasserman. Sparse Nonparametric Graphical Models. *ArXiv e-prints*, January 2012. 1, 5.2

[39] Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of markov networks using l_1-regularization. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 817–824. MIT Press, Cambridge, MA,

2007. 3.4.3

[40] David M. Leitner. Energy flow in proteins. *Annu. Rev. Phys. Chem.*, 59:233–259, 2008. 8.1

[41] R. M. Levy, A. R. Srinivasan, W. K. Olson, and J. A. McCammon. Quasi-harmonic method for studying very low frequency modes in proteins. *Biopolymers*, 23:1099–1112, 1984. 8.2

[42] M. Li, James T. Kwok, and B. L. Lu. Making Large-Scale Nyström Approximation Possible. *ICML 2010: Proceedings of the 27th international conference on Machine learning*, pages 1–8, May 2010. 7

[43] Chih-Jen Lin. Support Vector Machines. *Talk at Machine Learning Summer School, Taipei*, 2006. 5.3

[44] Yi Lin and Hao Helen Zhang. Component selection and smoothing in multivariate nonparametric regression. *The Annals of Statistics*, 34(5):2272–2297, 2006. 6.4

[45] Han Liu, John Lafferty, and Larry Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *J. Mach. Learn. Res.*, 10:2295–2328, December 2009. ISSN 1532-4435. 1, 5.1, 7.6

[46] P. Liu, Q. Shi, H. Daumé III, and G.A. Voth. A bayesian statistics approach to multiscale coarse graining. *J Chem Phys.*, 129(21):214114–11, 2008. 8.2

[47] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. Graphlab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, Catalina Island, California, July 2010. 9.1.4

[48] L. Lu, S. Izvekov, A. Das, H.C. Andersen, and G.A. Voth. Efficient, regularized, and scalable algorithms for multiscale coarse-graining. *J. Chem. Theory Comput.*, 6:954ñ965, 2010. 8.2

[49] K. V. Mardia. Statistics of directional data. *J. Royal Statistical Society. Series B*, 37(3): 349–393, 1975. 2.2.3, 3.2, 3.4.3

[50] Kanti V. Mardia, Charles C. Taylor, and Ganesh K. Subramaniam. Protein bioinformatics and mixtures of bivariate von mises distributions for angular data. *Biometrics*, 63(2):505–512, 2007. doi: doi:10.1111/j.1541-0420.2006.00682.x. URL `http://www.ingentaconnect.com/content/bpl/biom/2007/00000063/00000002/art00022`. 2.2.3

[51] Kanti V. Mardia, Gareth Hughes, Charles C. Taylor, and Harshinder Singh. A multivariate von mises distribution with applications to bioinformatics. *Canadian Journal of Statistics*, 36(1):99–109, 2008. ISSN 1708-945X. doi: 10.1002/cjs.5550360110. URL `http://dx.doi.org/10.1002/cjs.5550360110`. 2.2.3

[52] K.V. Mardia and P.E. Jupp. *Directional statistics*. Wiley Chichester, 2000. 2.2.3, 3.6

[53] Ugo Mayor, Christopher M. Johnson, Valerie Daggett, and Alan R. Fersht. Protein folding and unfolding in microseconds to nanoseconds by experiment and simulation. *Proceedings of the National Academy of Sciences*, 97(25):13518–13522, 2000. doi: 10.1073/pnas.250473497. URL `http://www.pnas.org/content/97/25/13518.abstract`. 3.7.2, 3.7.2

[54] Ugo Mayor, J. Gunter Grossmann, Nicholas W. Foster, Stefan M.V. Freund, and Alan R. Fersht. The denatured state of engrailed homeodomain under denaturing and native conditions. *Journal of Molecular Biology*, 333(5):977 – 991, 2003. ISSN 0022-2836. doi: 10.1016/j.jmb.2003.08.062. URL http://www.sciencedirect.com/science/article/pii/S0022283603011082. 3.7.2, 3.7.2

[55] Nicolai Meinshausen and Peter Bhlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):pp. 1436–1462, 2006. ISSN 00905364. URL http://www.jstor.org/stable/25463463. 5.3.2, 6, 6.1

[56] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, pages 362–369, 2001. 2.1, 3.5

[57] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI'99, pages 467–475, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-614-9. URL http://dl.acm.org/citation.cfm?id=2073796.2073849. 2.1

[58] E. Nadaraya. On estimating regression. *Theory of Prob. and Appl.*, 9:141–142, 1964. 5.3.1

[59] E.J. Nystrm. ber die praktische auflsung von integralgleichungen mit anwendungen auf randwertaufgaben. *Acta Mathematica*, 54(1):185–204, 1930. ISSN 0001-5962. URL http://dx.doi.org/10.1007/BF02547521. 7, 9.1.3

[60] Michael R Osborne, Brett Presnell, and Berwin A Turlach. A new approach to variable selection in least squares problems. *IMA journal of numerical analysis*, 20(3):389–403, 2000. 6.1

[61] V. S. Pande, I. Baker, J. Chapman, S. P. Elmer, S. Khaliq, S. M. Larson, Y. M. Rhee, M. R. Shirts, C.D. Snow, E. J. Sorin, and B. Zagrovic. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers*, 68(1): 91–109, 2003. 8.1

[62] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076, 1962. ISSN 00034851. 5.3

[63] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241 – 288, 1986. ISSN 0004-3702. doi: 10.1016/0004-3702(86)90072-X. URL http://www.sciencedirect.com/science/article/pii/000437028690072X. 2.1, 3.5

[64] J. C. Philips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. V. Kale, and K. Schulten. Scalable molecular dynamics with namd. *J. Comp. Chem.*, 26(16):1781–1801, 2005. 8.1

[65] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957. 5.2, 5.3.4

[66] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *In Advances in Neural Information Processing Systems (NIPS*, 2007. 7, 7.2

[67] Pradeep Ravikumar, Martin J Wainwright, Garvesh Raskutti, and Bin Yu. High-

dimensional covariance estimation by minimizing 1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011. 6.1

[68] Narges Razavian, Subhodeep Moitra, Hetu Kamisetty, Arvind Ramanathan, and Christopher J. Langmead. Time-varying gaussian graphical models of molecular dynamics data. *Proceedings of 3DSIG 2010 Structural Bioinformatics and Computational Biophysics*, 2010. (document), 2.1

[69] Narges Razavian, Hetunandan Kamisetty, and Christopher Langmead. Learning generative models of molecular dynamics. *BMC Genomics*, 13(Suppl 1):S5, 2012. ISSN 1471-2164. doi: 10.1186/1471-2164-13-S1-S5. URL `http://www.biomedcentral.com/1471-2164/13/S1/S5`. 9.1.5

[70] Volker Roth. Sparse kernel regressors. 2130:339–346, 2001. 5.3.1

[71] Mark Schmidt. Least squares optimization with l1-norm regularization. 2005. 6.1

[72] Mark Schmidt, Glenn Fung, and Romer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. pages 286–297, 2007. 3.7.1

[73] Mark Schmidt, Kevin Murphy, Glenn Fung, and Rmer Rosales. Structure learning in random fields for heart motion abnormality detection. In *CVPR*. IEEE Computer Society, 2008. 3.4.3

[74] D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, M. P. Eastwood, J. Gagliardo, J. P. Grossman, C. R. Ho, D. J. Ierardi, I. Kolossváry, J. L. Klepeis, T. Layman, C. McLeavey, M. A. Moraes, R. Mueller, E. C. Priest, Y. Shan, J. Spengler, M. Theobald, B. Towles, and S. C. Wang. Anton, a special-purpose machine for molecular dynamics simulation. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 1–12, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-706-3. doi: http://doi.acm.org/10.1145/1250662.1250664. 3.7.2, 8.1

[75] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *Algorithmic Learning Theory*. Springer, 2007. Invited paper. 5.3.2

[76] Alex J. Smola and Bernhard Schokopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 911–918, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2. 7

[77] L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions. In *International Conference on Machine Learning*, 2009. 5.3.2

[78] L. Song, A. Gretton, and C. Guestrin. Nonparametric tree graphical models. In *Artificial Intelligence and Statistics (AISTATS)*, 2010. 1, 5.3.2, 5.3.3

[79] L. Song, A. Gretton, D. Bickson, Y. Low, and C. Guestrin. Kernel belief propagation. In *International Conference on Artifical Intelligence and Statistics (AISTATS)*, 2011. 1, 5.3.2, 5.3.3, 7, 7.1, 7.3, 9.1.3

[80] L. Song, A. Parikh, and E. Xing. Kernel embeddings of latent tree graphical models. In *Neural Information Processing Systems (NIPS)*, 2011. 5.3.4, 6, 6.2

[81] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, March 2002. ISSN 1532-4435. doi: 10.1162/153244302760185252. URL http://dx.doi.org/10.1162/153244302760185252. 5.3.2

[82] J.E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten. Accelerating molecular modeling applications with graphics processors. *J. Comp. Chem.*, 28:2618–2640, 2007. 8.1

[83] Erik B. Sudderth, Alexander T. Ihler, Michael Isard, William T. Freeman, and Alan S. Willsky. Nonparametric belief propagation. *Commun. ACM*, 53(10):95–103, October 2010. ISSN 0001-0782. doi: 10.1145/1831407.1831431. URL http://doi.acm.org/10.1145/1831407.1831431. 2.1, 2.2.4, 3.5

[84] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):pp. 267–288, 1996. ISSN 00359246. URL http://www.jstor.org/stable/2346178. 6.1

[85] JA Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006. 3.4.3

[86] L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19:499–533, 1998. 8.4

[87] Martin J. Wainwright, Pradeep Ravikumar, and John D. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1465–1472. MIT Press, Cambridge, MA, 2007. 3.4.3

[88] Geoffrey S. Watson. Smooth regression analysis. *Sankhy: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):pp. 359–372, 1964. ISSN 0581572X. URL http://www.jstor.org/stable/25049340. 5.3.1

[89] Christopher Williams and Matthias Seeger. Using the nystrm method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001. 7

[90] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51 (7):2282 – 2312, july 2005. ISSN 0018-9448. doi: 10.1109/TIT.2005.850085. 2.1

[91] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved nystrom low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 1232–1239, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. doi: 10.1145/1390156.1390311. 7.4

[92] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Scholkopf. Kernel-based conditional independence test and application in causal discovery. *arXiv preprint arXiv:1202.3775*, 2012. 6, 6.4

[93] Tuo Zhao, Kathryn Roeder, and Han Liu. Kernel based conditional independence test ad

application in causal discovery. 2012. 6.1

[94] Yingdong Zhao and Hengming Ke. Mechanistic implication of crystal structures of the cyclophilindipeptide complexes,. *Biochemistry*, 35(23):7362–7368, 06 1996. URL http://dx.doi.org/10.1021/bi960278x. 8.6

[95] Shuheng Zhou, John D. Lafferty, and Larry A. Wasserman. Time varying undirected graphs. In *COLT*, pages 455–466, 2008. 8.3