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

Narges Sharif Razavian

Thesis Proposal , November 2012

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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.

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(vMGM), and develop consistent and efficient algorithms for sparse structure learning and parameter estimation, and efficient inference. We compare our model to sparse Gaussian graphical model(GGM) and show it outperforms GGMs on synthetic and real protein dynamics datasets.

To solve the uni-modality problem of vMGMs, we propose improved structure learning and inference in non-parametric, kernel space embedded graphical models. We perform preliminary experiments for structure learning via Neighborhood selection. We then propose to improve the structure learning by using non-parametric regression and ensemble-of-tree models. Also, we propose methods to increase the scalability of the non-parametric graphical models by developing algorithms based on random features and kernel approximation methods, which specifically take advantage of shared components in the learning and inference process.

Making the assumption that protein dynamics data is independent and identically distributed is not correct when we are modeling dynamics in non-equilibrium and folding simulations, or when protein is undergoing various conformational changes. In this thesis, we also provide improved time-varying continuous graphical models to better model the protein dynamics and energy landscape. We show preliminary results on time-varying sparse Gaussian graphical models, applied to protein dynamics data, and propose to improve the model by using semi-parametric graphical models as the underlying model. If time permits, we also propose nonparametric clustering, based on Dirichlet process and hierarchical Dirichlet process, to improve the kernel based re-weighting of the likelihood, to learn better generative models of protein dynamics.

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. In this thesis we focus on MRFs, because the structures we deal with in our applications 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 tradeoff 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 [20],[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[19], 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. Semi-parametric[43],[36], and non-parametric [74],[76] graphical models can model complex distributions with multi-modal and arbitrary marginals. In

this thesis we will also focus on using these models for discovering the generative model of protein structure. Specifically, we will focus on improving the structure learning and scalability of those graphical models.

In this thesis, our main application is computational structural biology. We focus on modeling the protein structure from Molecular Dynamics(MD) simulation data. MD simulations are often used to characterize conformational dynamics of proteins[31]. These simulations are performed by numerically integrating Newton's laws of motion for a set of atoms. Conformational frames are written to disk into a trajectory for subsequent analysis. Current hardware allows MD simulations to continue up to milliseconds for many proteins[10],[58],[70],[79],[62], This time scale is often enough for identifying and studying the conformational sub-states relevant to biological function. We have access to several MD simulations of important proteins and enzymes, such as Engrailed Homeodomain (three 50-microsecond simulations of the protein at 300, 330 and 350 Kelvin), and CypA (Steered MD simulation with three different substrates), and will focus most of our experiments on these datasets.

A particular property of our application domain is the non-iid nature of our datasets. In the second part of our thesis, we 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. As we will see, we propose to learn semi parametric time varying graphical models, and also we propose to perform nonparametric clustering of the samples before learning the time-varying model which makes the protein energy landscape modeling more accurate.

Specifically, chapter 2 focuses on reviewing relevant and background work related to parametric, nonparametric and time varying undirected graphical models, which are capable of modeling continuous variables. In chapter 3, we will present our thesis work on graphical models over iid samples. In section 3.1, we will focus on structure and parameter learning and inference in von-Mises graphical models, and in section 3.2, we will focus on structure learning and inference in non-parametric graphical models, specifically for reproducing kernel Hilbert space embedded graphical models. We present our results, using Engrailed MD simulation dataset in both models. In chapter 4 we focus on time-varying models for non-iid samples. In section 4.1, we present preliminary results of learning sparse time-varying Gaussian graphical models and its application to CypA enzyme dynamics modeling. We propose semi-parametric time-varying graphical models in section 4.2, and nonparametric clustering for time-varying graphical models in section 4.3. Finally in chapter 5 we will present our proposed time line, for the completion of proposed tasks.

We label section headers with $Proposed\,Work$ to distinguish them from completed work, or background material.

1.1 Thesis Statement

We focus on development of graphical models to learn improved generative models of molecular dynamics simulation data, in two main categories:

First, when dealing with static domain, we propose

(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 novel structure learning in non-parametric (reproducing kernel Hilbert space embedded) models to improve the generative models of protein structure.

(3) Novel algorithms to scale RKHS learning and inference methods, to be able to handle large MD datasets, with specific focus to take advantage of shared components in the learning and inference process.

Second, when dealing with dynamic data from MD simulations, we propose:

(1) Time-Varying sparse Gaussian graphical model, to estimate better generative models of MD simulation data, which help in predicting the energy landscape of the protein

(2) Development of Novel time-varying semi-parametric graphical models, to improve generative models of MD simulation data,

(3) [Time permitting,] Nonparametric clustering based on Dirichlet process and hierarchical Dirichlet process, for time-varying graphical models, to learn sub-states and transitional states in the energy landscape of the protein.

Chapter 2

Background and Related Work

In this chapter, we will review existing techniques that can be used for dealing with inference, and structure and parameter learning of undirected graphical models. We will start by providing basic background about undirected graphical models. After that, we will review existing methods that deal with large multivariate continuous distributions, and we use computational structural biology as an example application, and look at the models that have been applied to this particular application.

We categorize the current methods for learning and inference over continuous undirected graphical models into two main categories. First category, covered in section 2.3, is the discrete and parametric approach, which models the relationship between variables as either discrete probability tables, or well formed continuous functions of certain function families, in which the functions can be represented compactly by a small set of parameters. Second group of methods are semi-parametric and non-parametric methods, which we will cover in section 2.4. Semi-parametric methods assume that the the graphical model has a parametric form, but only after the data is transformed into a feature space. Nonparametric methods, are based on embedding the graphical models and inference in a feature space specified by a kernel function, without imposing any parametric form in the feature space.

Finally, in section 2.5, we will review time varying models, which are important models when the data is not iid. Since the applications we have selected such as computational structural biology, have $non - iid$ datasets, we will focus on efficient structure learning and inferences for time-varying graphical models as well.

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 2.1 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

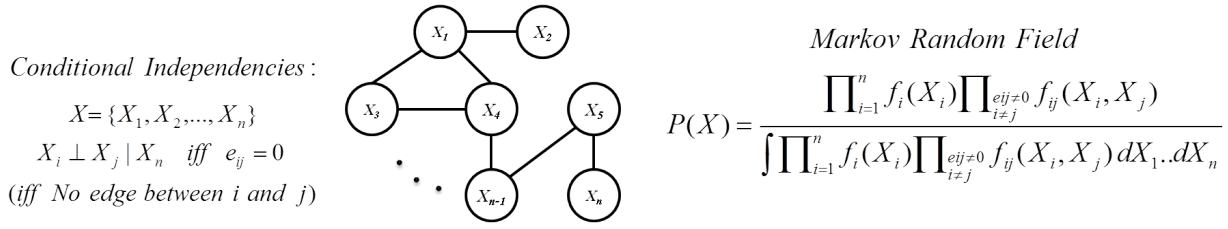


Figure 2.1: Markov Random Field Example

into a normalized set of node and edge factors, f_i and f_{ij}. The normalization term, sometimes 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) [60] is currently the most common approach to calculate 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 [55], is used on loopy UGMs. While loopy belief propagation is missing the convergence guaranties associated with BP, Yedida et. al. [87] 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 [53], and Gaussian mixture belief propagation[80], and particle belief propagation[28].

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 Problem Definition for Protein Structure Modeling

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 2.2 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

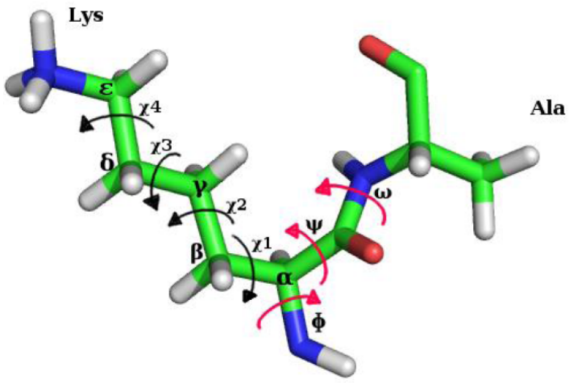


Figure 2.2: Backbone and side-chain dihedral angles of a di-peptide Lys-Ala protein. Picture from [57]

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.

2.3 Learning and Inference in Discrete and Parametric Graphical Models

2.3.1 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.[29][24].

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 over-fitting 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 these solutions.

2.3.2 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.[20] 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.

2.3.3 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 [19]. It closely approximates the wrapped normal distribution[12], but has the advantage that it is more tractable, mathematically [49]. 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 [46] over θ_1 and θ_2, for example, can be defined as:

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

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[47] and also multivariate[48] von Mises models applied to protein angles, and provide an algorithm based on full pseudo-likelihood to perform parameter estimation.[25] 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 next chapter, we develop a general sparse structure learning method, in addition to parameter estimation, and fast inference based on Expectation Propagation for von Mises graphical models.

2.3.4 Gaussian Mixtures Graphical Model

Both of GGM and von-Mises graphical methods have a major shortcoming, which is the unimodality 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

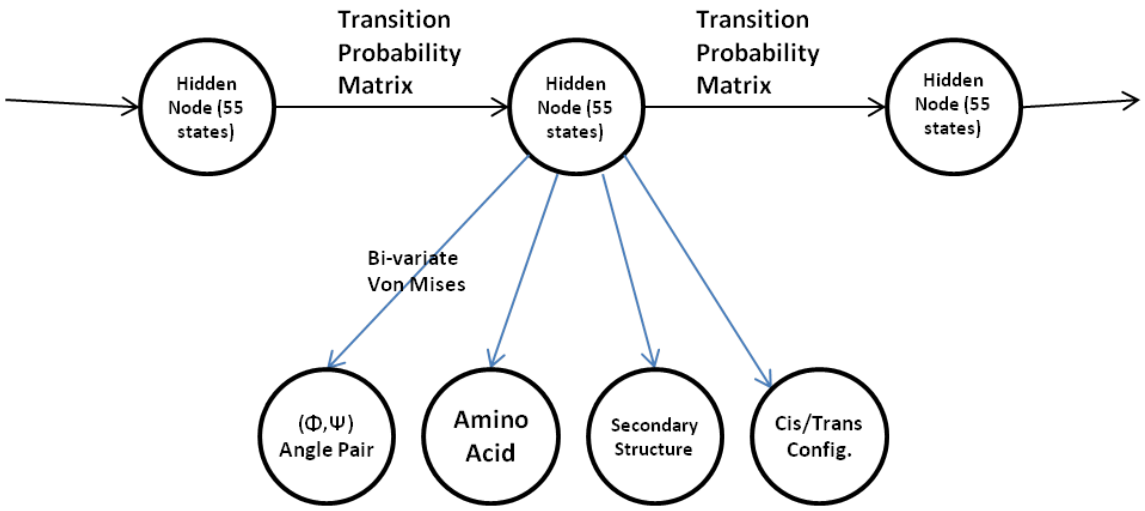


Figure 2.3: Generative model for Dynamic Bayesian Network with von Mises emissions

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 any reasonable estimation of the data.

Some groups have tackled this multi-modality by replacing unimodal factor functions with mixtures of Gaussian. In [80], 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.

Mixture of Gaussian graphical models are strong models, but optimizing the maximum number of components per message poses an NP-hard problem, unless all messages are assumed to share the number of mixture components, which is unrealistic. As we will see in the coming section, semi-parametric and non-parametric graphical models provide better solutions to this problem.

2.4 Structure Learning and Parameter Estimation and Inference in Semi-parametric and Non-parametric Graphical Models

Semi-parameric and non-parametric models try to reduce the number of parameters required in model specifications, by scaling them according to the number of samples themselves. In other words, these models use the data itself as the source to calculate the required densities and expectations, without imposing a specific parametric form on the model. In this section we will review state of the art semi-parametric and non-parametric methods.

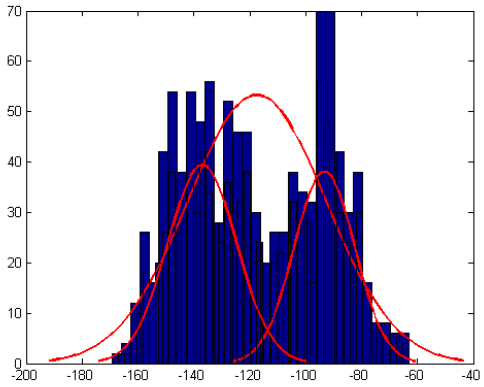


Figure 2.4: Histogram and Gaussian distribution fitting for Pro4 amino acid in Engrailed Protein, after fitting single and mixture of two Gaussians.

2.4.1 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.[43].

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.

2.4.2 Nonparametric Forest Graphical Models

Lafferty et. al. recently proposed a non-parametric forest structure learning method[36], 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[34][63].

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 [35], which they have not yet tested in this context.

2.4.3 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 [59].

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, $\phi(x) = e^{-\lambda x^2}[1, \sqrt{\frac{2\lambda}{1!}}x, \sqrt{\frac{(2\lambda)^2}{2!}}x^2, \sqrt{\frac{(2\lambda)^3}{3!}}x^3, ...]$ [40], 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.

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 2.5 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[56], [86], as well. Roth [66] proposed sparse kernel regression, which uses support vector method to solve the regression problem.

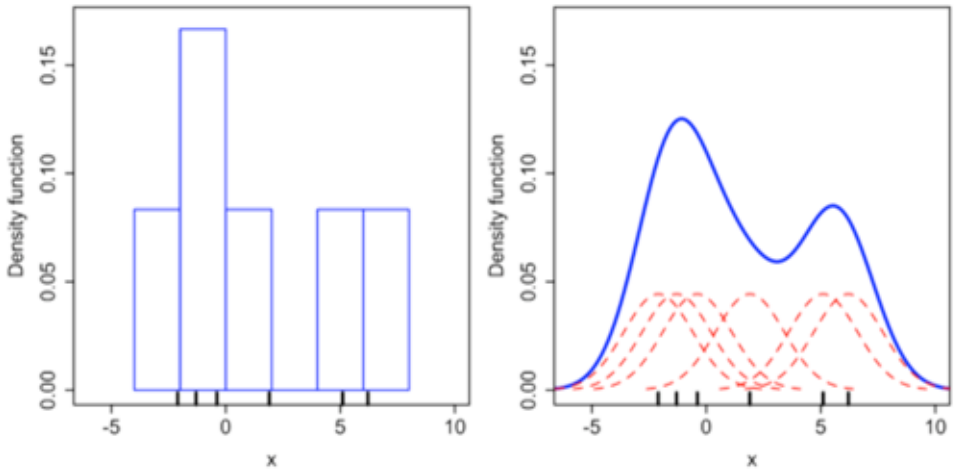


Figure 2.5: Kernel Density Estimation Example

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

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 $\Re$. [15].

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

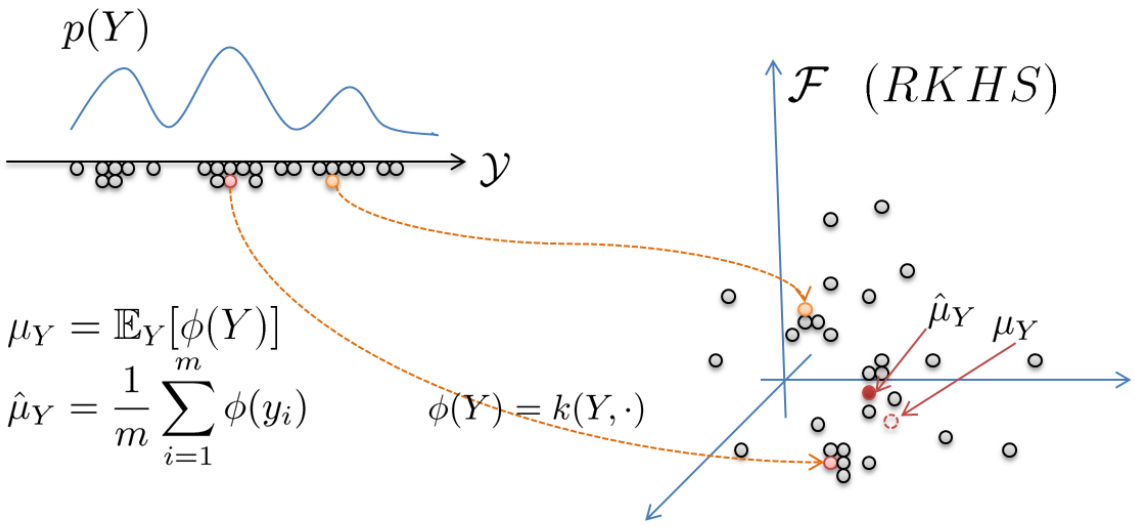


Figure 2.6: Reproducing Kernel Hilbert Space embedding of a probability distribution

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.[72] 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[78] 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 2.6 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. [73] define covariance operator, on pairs of variables (i.e. $D_{XY} = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$), as:

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

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:

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

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 2.7 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 [74], 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 [76], 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 proposal to use sparse structure learning methods, such as neighborhood selection[52], that have been successful in other contexts, to learn the structure in the RKH space, and perform nonparametric inference.

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 [74] and [76], the following formulations are presented:

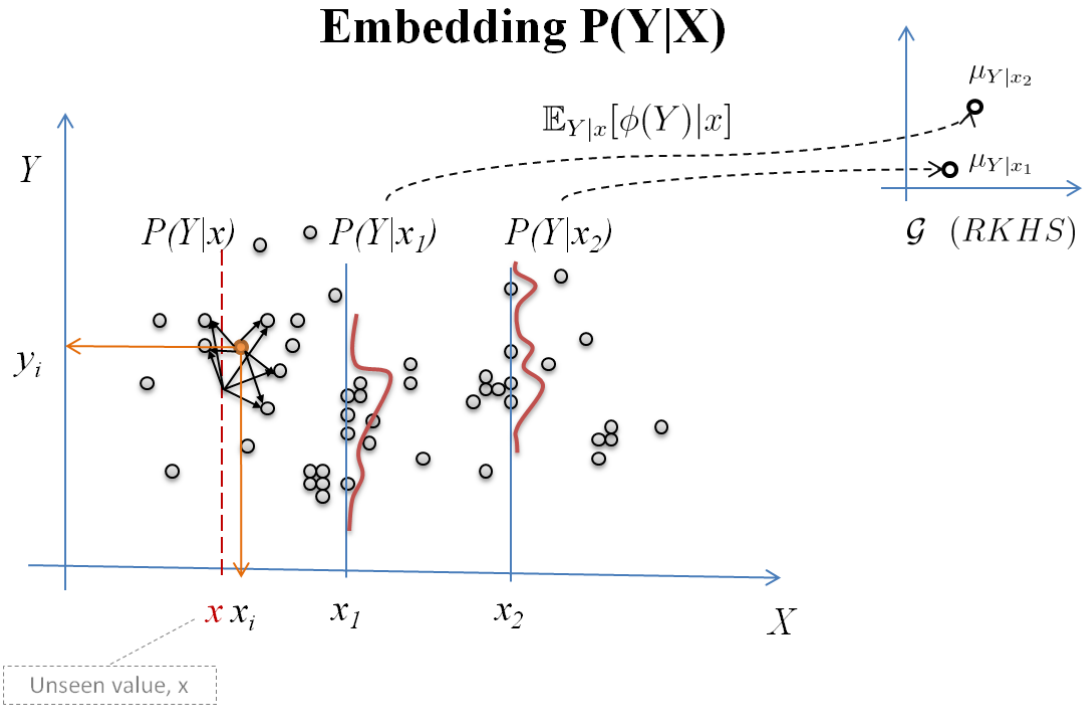


Figure 2.7: Reproducing Kernel Hilbert Space embedding of conditional distributions

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

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

Where $\bigodot$ is the element-wise vector product. Again, if we define $\beta_{ts} := (L + \lambda m I)^{-1} (\bigodot_{u \backslash s} K \beta_{ut})$ 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 propose to solve.

Tree Structure Learning for RKHS Tree Graphical Models

Recently in [77], 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 . [13], 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 [34][63].

According to [13], 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[34][63] with this distance measure, and learn an optimum tree structure non-parametrically in RKHS. We use this model as our baseline when comparing other structure learning algorithms in the RKH space, in chapter 3.2.1.

2.5 Time-varying continuous graphical models

Some data such as Molecular Dynamics simulations are inherently time-varying, and they do not fit the independent and identically distributed (iid) assumption. Time varying graphical models are models which are built to handle non-iid datasets, and model the trajectory of the data, the

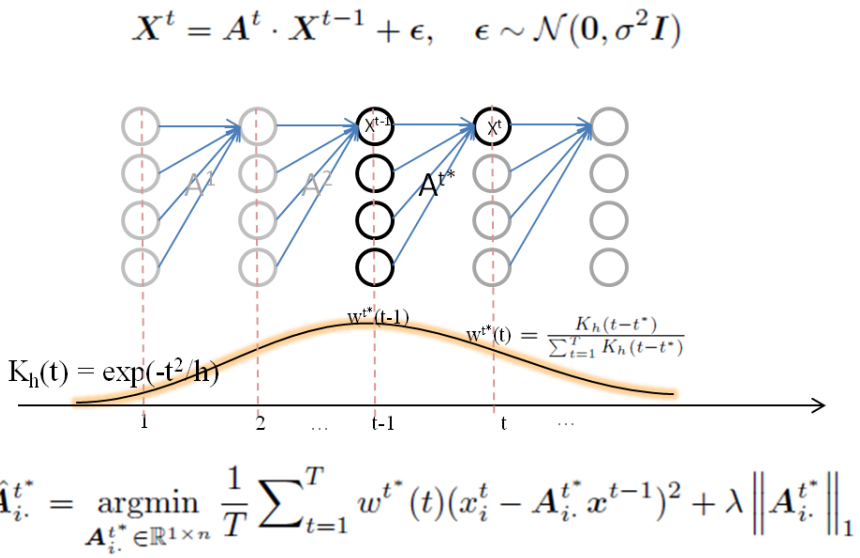


Figure 2.8: Kernel re-weighted dynamic networks

static sub-states, and also transition between them. Particularly in the case of protein folding molecular dynamics simulation datasets, it is usually the case where the data contains many locally optimal sub-states and several energy barriers and transitional sub-states between them.

Kolar et. al [33] presented two methods to estimate hidden structure and parameters of a time-varying network, both based on temporally smoothed L-1 regularized logistic regression. In one model the assumption is that the parameters and structure change smoothly and a weighted log likelihood is defined and optimized over the data. In the second model they assume a blocked model where the structure and parameters of the model is constant over a block of time and the model applies two regularization penalties, Lasso and group Lasso, to enforce sparse structure in each block and as small changes between blocks as possible. The group regularization technique is also presented in [32].

Another model to describe time-varying data is Dynamic Bayesian networks (DBNs) [23], [54]. DBNs are dynamic directed graphical models which describe sequential observations. The dynamics of the data is described through transition probabilities and states in the hidden variables of a DBN. Usually the transition links between the hidden variables is fixed and it is their states that control the dynamic model. For large time-varying networks such as protein structural data, these systems become infeasible.

Song et. al. [75] presented time-varying dynamic Bayesian networks, which is similar to the discrete time varying graphical model, based on smoothly varying structure which is designed using a kernel re-weighted loss function. They define their loss function as regression error. Figure 2.8 shows the example of the kernel re-weighted dynamic model, used in this model.

The dynamic models discussed so far are all on discrete data, and discretization has to be applied on continuous variables before modeling. As we saw in previous sections, discretization creates several problems, including the large number of parameters to estimate, which leads to over-fitting. Finale et. al.[18] presents infinite dynamic Bayesian network model as one solution,

where hidden variables, their hidden states, and the transitions and emissions are generated non-parametrically from the data, via Dirichlet processes and Indian Buffet Processes. Nonparametric models select the model complexity that is appropriate to the available data size. They use Gibbs sampling to estimate the model, states and transitions, and also perform inference. As with other Dirichlet process based models, their model is very slow to converge, and can only handle small discrete tables, and not continuous variables.

A continuous and parametric time-varying model was presented in 2008 by Zhou et. al [89]. They introduced a time-varying Gaussian graphical model. They provided a smoothly varying structure and parameter estimation model, where a smoothing kernel was used to calculate weights in the weighted log likelihood, and they used a Gaussian graphical model to estimate the structure and parameters of the graph as it varied in time. This model scales easily to large protein folding datasets, but has several shortcomings: First, the model is not sparse, so to increase data efficiency we must develop an L1-regularized time-varying Gaussian graphical model, which we have already developed and will cover in the next chapter.

Second: Gaussian graphical models, while analytically efficient, don't fit the data in practical applications, so in the future chapter we will propose to replace the Gaussian graphical model with semi-parametric and nonparametric graphical model to keep the models efficient and powerful.

Third: The current work does not gives us any insight into how to chose our smoothing kernel to re-weight the log likelihood. In all the previous systems [33],[75],[89], the kernels have been symmetric, fixed-length functions, which were too simplified. In applications such as modeling protein folding trajectory, the sub-states have different lengths and the kernel with different effective length is a more reasonable. Thus we propose that the length of the kernel be designed based on a nonparametric clustering of the data into sub-states, so that we adjust the kernel for each sub-state separately. More detailed discussion of this topic will follow in chapter 4.

Chapter 3

Sparse Structure and Parameter learning and Inference for Continuous Graphical Models

In this chapter we propose our solutions for practical graphical models in parametric and non-parametric families of graphical models, and provide preliminary analysis, using computational structural biology as our application area.

In some parametric graphical models, specially von-Mises graphical models, no structure learning has been performed yet and also inference has been limited to Gibbs sampling which is slow to converge. In section 3.1 We propose to provide a model to simultaneously estimate sparse structure and the parameters of the von-Mises graphical models. Also since we deal with large proteins we propose to develop an efficient and consistent algorithm based on Expectation Propagation, to perform inference on the von-Mises graphical models. We will also use L1-regularized Gaussian Graphical Models[5] as our baseline.

Given the strengths of non-parametric graphical models, specifically the ability to deal with multi-modal and arbitrary distributions, we also propose to perform sparse structure learning in the RKHS space in section 3.2. We will use existing methods such as neighborhood selection[52], tree-ensemble learning[42], as well as non-parametric regression model[35], to perform sparse structure learning. We present our preliminary results using neighborhood selection method over Engrailed protein dataset. Current RKHS inference method is not scalable to the size of our dataset, so we also propose our solution to use hashing methods and parallelization to overcome the scalability issue.

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

Von Mises distribution is a continuous probability distribution defined on a circle, and is used in directional statistics. In this section, we introduce the undirected von-Mises Graphical model and present an algorithm for structure learning using L_1 regularization. We show that the learning algorithm is both consistent and efficient. We also introduce an efficient inference algorithm

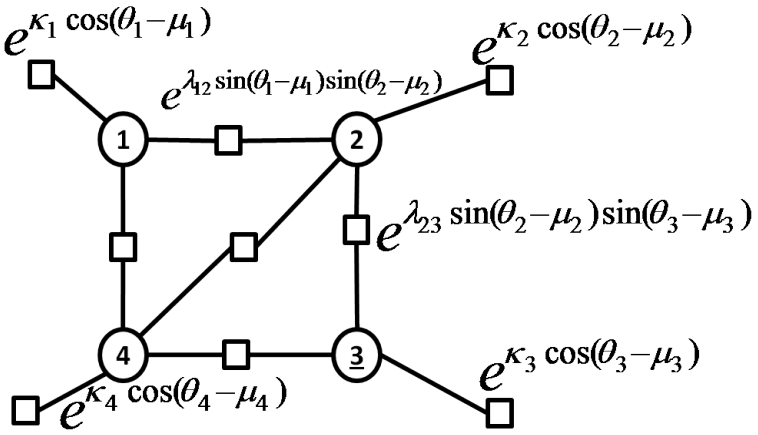


Figure 3.1: Factor Graph Representation for multivariate von Mises distribution. Each circular node is a variable, and the square nodes are factors.

based on 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.

3.1.1 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 [46] 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) [25]. This is significant because learning and inference can be performed analytically for multivariate Gaussian distributions. However, we will show in Section 3.1.6 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.

3.1.2 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 [25].

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.

3.1.3 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 3.1.1. 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.

Full pseudo-likelihood for von Mises Graphical Model

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

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

As discussed in section 3.1.2, each univariate conditional term for the vGM is itself a univariate von Mises distribution. Thus, the full pseudo likelihood can be re-written as:

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

where

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

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 [17]. 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^*, for all i and j values.

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 [17] proves that the Full Pseudo Likelihood is a $consistent$ estimator for the vGM.

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.[37],[26], [68],[85]). Structure learning algorithms based on L_1 regularization are particularly interesting because they exhibit consistency and high statistical efficiency (see [83] for a review). We use an algorithm introduced by Schmidt et.al. [68] 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. [68] 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 3.1.3:

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

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

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 [46].

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.

3.1.4 Inference in von Mises Graphical Models

One of the most widely used Inference techniques on Graphical Models is Belief Propagation [60]. 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.

Tom Minka [53] proposed a Expectation propagation as an approximate inference for these graphical models. Expectation Propagation is similar to Belief Propagation, except that the messages that are exchanged between the nodes only contain the $expectation$ of the marginalization, rather than the exact values.

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.

3.1.5 Expectation propagation for von Mises Graphical Models

Recall from section 3.1.1, 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.

As mentioned in previous section, in Expectation Propagation, messages contain the expectations rather than the actual parameters. In EP 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.

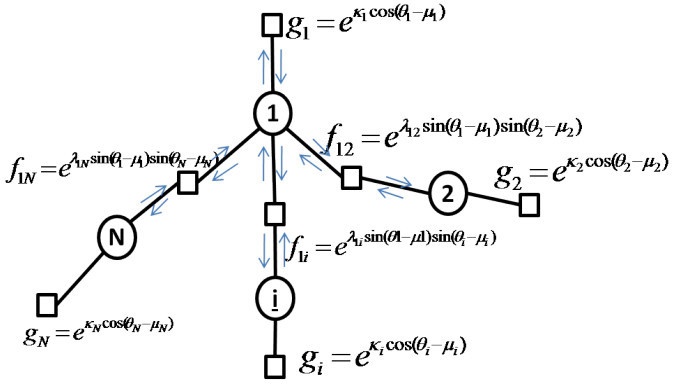


Figure 3.2: Von Mises Factor graph representation, and the messages exchanged between the factors and variable x_1.

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. [49] 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 messsage 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 $Z = \frac{1}{2\pi} I_0(\kappa_z^*)$.

3.1.6 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.

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. 3.1.2) 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 (Sec3.1.3) to learn a vGM from each data set. For comparison, we also used the structure learning algorithm presented in [67] 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 Expectation propagation, 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 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

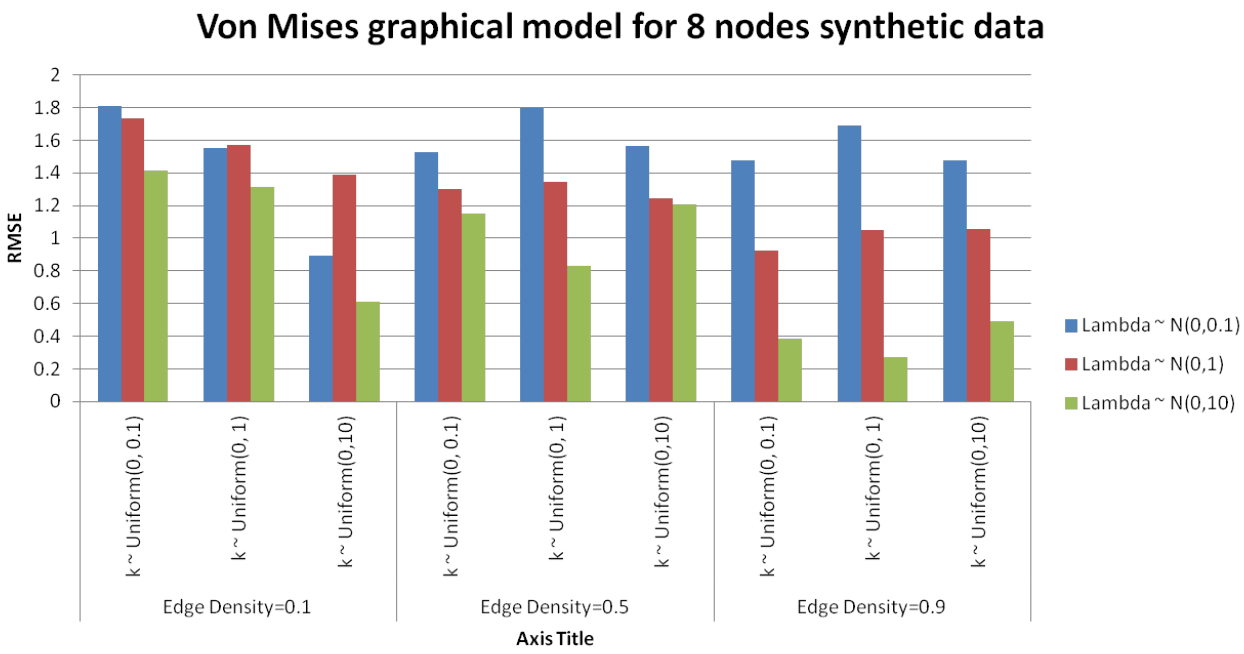


Figure 3.3: RMSE of estimated von-Mises graphical models on synthetic data with 8 nodes

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.

As previously mentioned (Sec. 3.1.1), a vGM can be well-approximated with a GGM when the variables have low variance (i.e., high κ).

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:

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

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

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

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

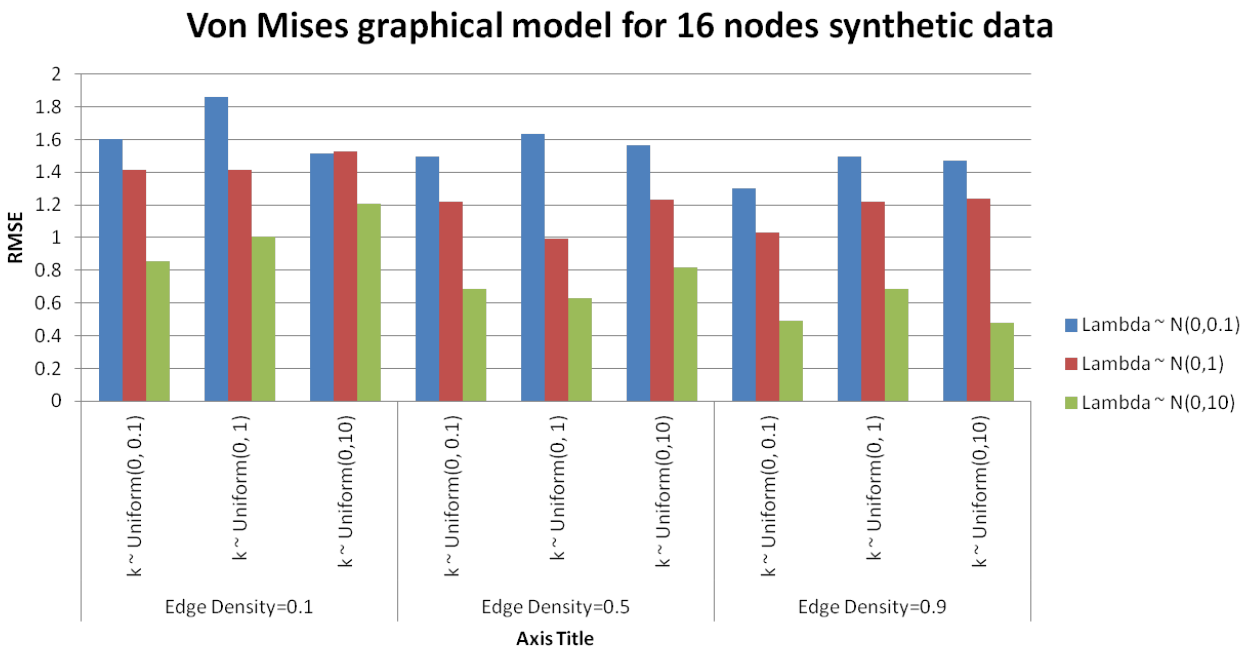


Figure 3.4: RMSE of estimated von-Mises graphical models on synthetic data with 16 nodes

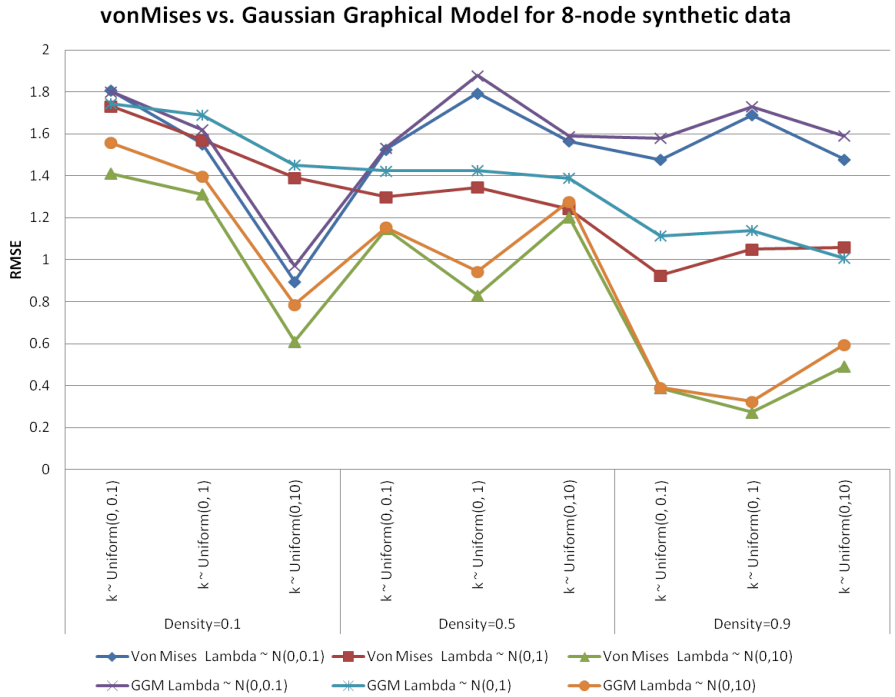


Figure 3.5: Comparing von-Mises graphical model with Gaussian graphical model on synthetic data with 8 nodes

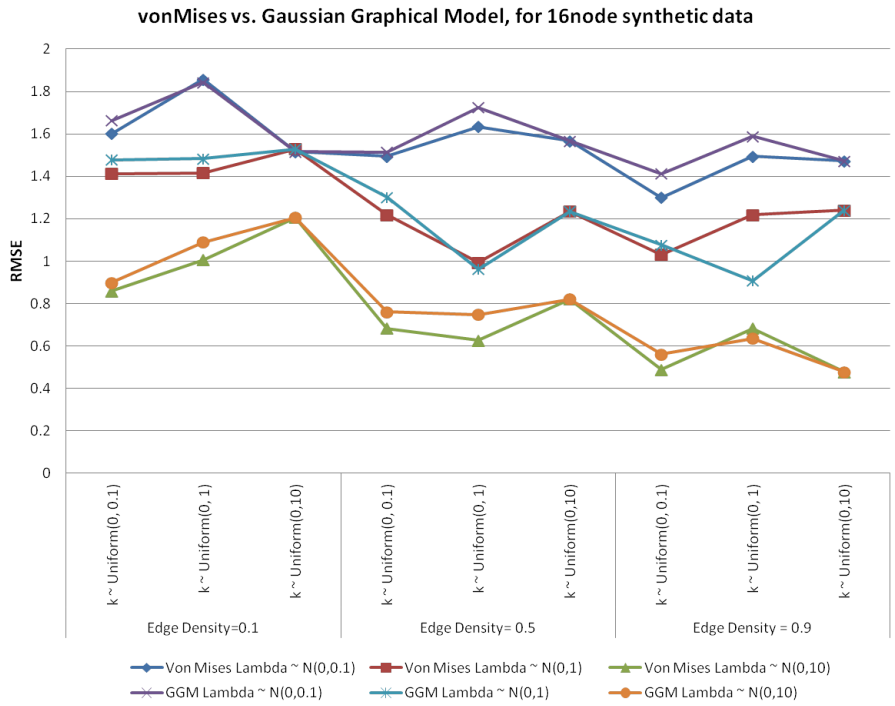


Figure 3.6: Comparing von-Mises graphical model with Gaussian graphical model on synthetic data with 16 nodes

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

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 $(\Sigma^{-1})_{ij}=-\Lambda_{ij}$.

In accordance with this, we observe 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.

Tables 3.1 and 3.2 show the $time$ and $RMSE$ of EP inference, 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
EP	0.34	0.37	0.79	1.34

Table 3.1: Comparing computational time(seconds) for Gibbs vs. EP inference for different nodes

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

RMSE	8 nodes	16 nodes	32 nodes	64 nodes
Gibbs	1.95	2.12	1.89	1.83
EP	1.52	1.51	1.40	1.31

Table 3.2: Comparing RMSE(degrees) for Gibbs vs. EP inference for different nodes

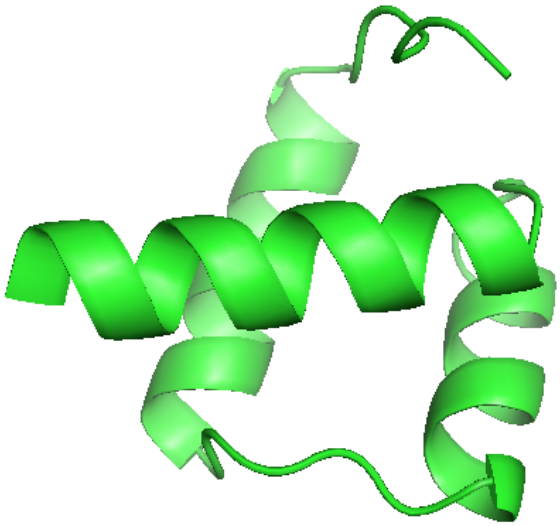


Figure 3.7: Engrailed Homeodomain

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 3.7. The DNA-binding domains of the homeotic proteins, called homeodomains (HD), play an important role in the development of all metazoans [21] and certain mutations to HDs are known to cause disease in humans [16]. Homeodomains fold into a highly conserved structure consisting of three alpha-helices wherein the C-terminal helix makes sequence-specific contacts in the major groove of DNA [22]. The Engrailed Homeodomain is an ultra-fast folding protein that is predicted to exhibit significant amounts of helical structure in the denatured state ensemble [51]. Moreover, the experimentally determined unfolding rate is of 1.1E + 03/sec [50], 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[70], a special-purpose supercomputer designed to perform long-timescale simulations. Each simulation had more than 500,000 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.

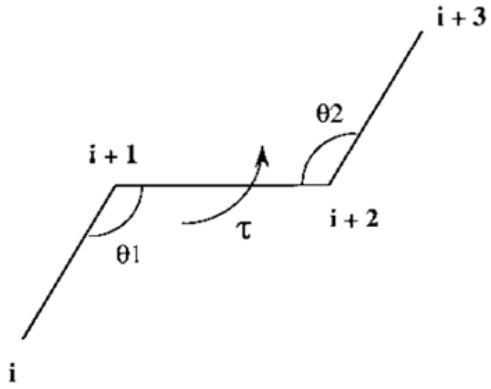


Figure 3.8: Theta and Tau angles representing a Protein Structure

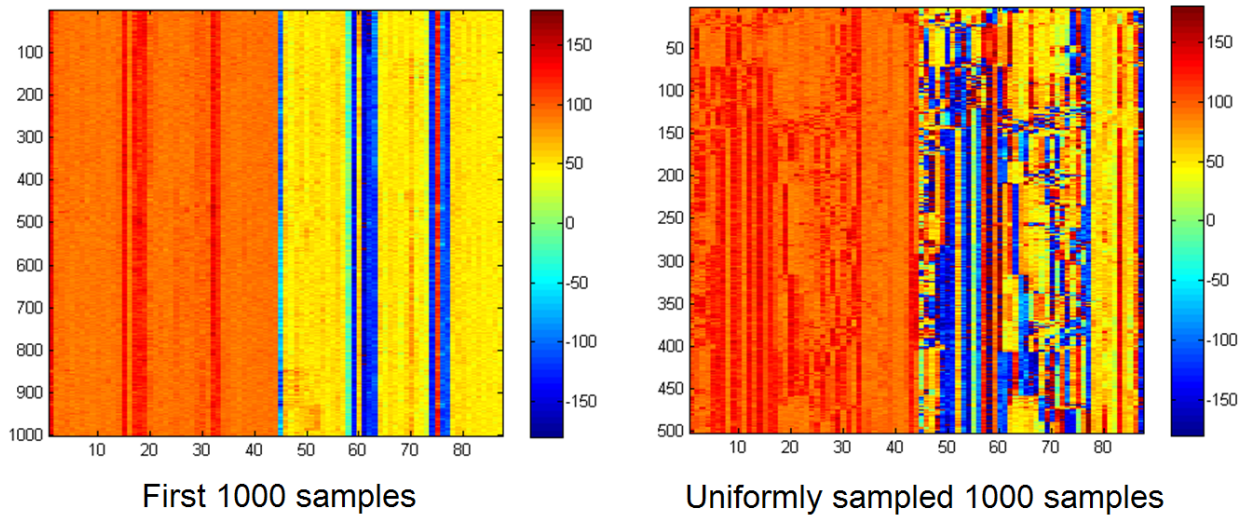


Figure 3.9: Engrailed protein structure frames

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 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. [50] 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. [51] identified Leucine16, as one of the core hydrophobic residues which plays important part in the stabilization of the protein.

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

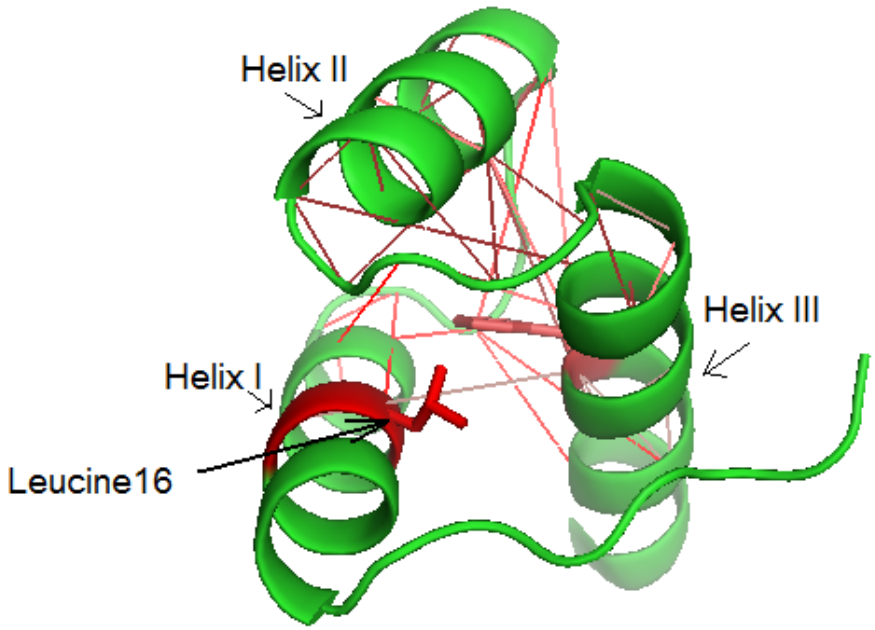


Figure 3.10: Structure Learned for Engrailed Homeodomain, via von Mises graphical models

As indicated in Figure 3.10, we also see that this residue couples strongly with Helix III region.

3.1.7 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 section, 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 the Expectation Propagation inference, developed specifically for Von-Mises graphical models. We used special techniques, including trigonometric moment matching, to derive the estimated parameters of the Expectation Propagation 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 section of this chapter, we will focus on models for multi-modal rather than uni-modal graphical models, using kernel space embedding technique.

3.2 Proposed Work: Graphical Models in Reproducing Kernel Hilbert Space - Sparse Structure learning and Inference

Reproducing kernel Hilbert space embedding of graphical models allows us to have multi-modal and hybrid graphical models. As discussed in section 2.4.3, structure learning in reproducing kernel Hilbert space currently only exists for Tree structured graphical models. For general graph structures, while there are already techniques such as neighborhood selection [52], ensemble of trees[42], and nonparametric greedy sparse regression models[35], none has yet been used in RKH space. In this section, we will review existing techniques for sparse structure learning in reproducing kernel Hilbert space, and as preliminary work, apply neighborhood selection method to the problem and perform inference. We will compare neighborhood selection based method to the tree structure learning method in RKHS, and also to the non-paranormal and sparse Gaussian graphical models, on syntactic and real protein data. We will then provide our proposed next steps to improve structure learning and scalability.

3.2.1 Sparse Structure Learning in Kernel Space by Neighborhood Selection

The problem of structure learning in Markov random fields is an NP-Hard problem, 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. [52], 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 regression problem, and can be solved efficiently with the Lasso regression[82]. They prove that this method is consistent, and in the limit recovers the true structure.

We use this method to perform the structure in RKH space, and compared it with the non-parametric tree structure learning algorithm. In the following section, we will cover our current set of experiments and results.

3.2.2 Experiments on Neighborhood Selection for RKHS Graphical Model Structure Learning

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 3.1.6. We used the same two sub-sampled datasets, first1000, and uniformly sampled 1000, as described in figure 3.9.

Data	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 3.4: 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
7.30	7.41

Table 3.5: RMSE result for RKHS using Neighborhood selection, versus Tree structure learning on First1000 dataset. Both methods used triangular kernel.

Evaluation

Currently we have performed leave-one-out cross-validation, and have calculated the Root Mean Squared Error (RMSE) of the test frame, given that the model is learned from the training set.

For each test frame, we have also assumed randomly selected 50% of the variables of the frame are observed, and have predicted the rest of the variables, given these observations and the training data. For each frame we have repeated this 50% subset selection 20 times.

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

Results

Table 3.4 shows the results of running the full cross validation experiment, on RKHS method with 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: Gaussian kernel, $K_1 = e^{-\lambda||x-y||^2}$, and triangular kernel, $K_2 = e^{-\lambda(sin(||x-y||))^2}$.

Table 3.5 shows the RMSE results of tree structure learning, versus the neighborhood selection method, 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.

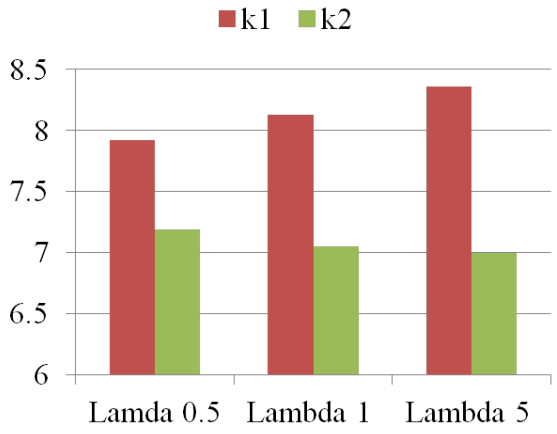


Figure 3.11: Effect of structure sparsity in neighborhood selecton on RMSE in RKHS inference

Also we observe that neighborhood selection outperforms the tree-based nonparametric structure learning, however we should note that neighborhood selection with Gaussian kernel over angular data does worse than tree structured 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 3.2.4.

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 3.11 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.

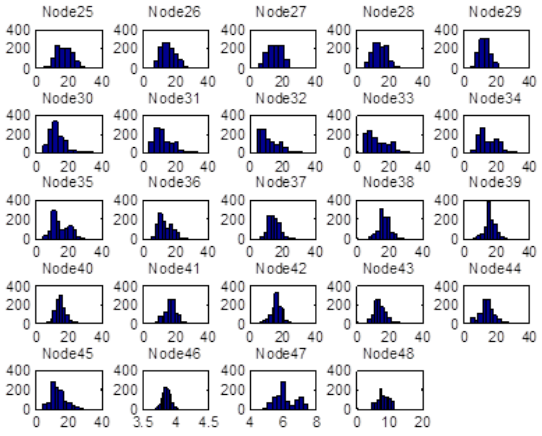
Experiments on Pairwise Distances Network

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 subsampled data.

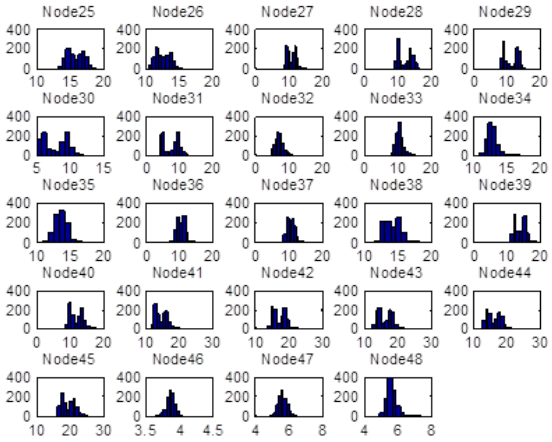
Figure 3.12 shows a collection of univariate marginals in both of the sub-sampled data set, and we see that multi-modal and asymmetric distributions are very common in both datasets.

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 3.6 shows the result of the RMSE calculations:

Based on these preliminary results, RKHS with Neighborhood selection outperforms all other existing methods in predicting the distances, as well as the angles.



Some marginals for Uniform1000 samples



Some marginals for First1000 samples

Figure 3.12: Marginal Distributions for a subset of pairwise distance variables in two sample sets

Data	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 3.6: RMSE result comparison for RKHS, non-Paranormal and Gaussian graphical models over Distance variables

3.2.3 Summary of Completed Work

In this section, we estimated sparse structure of the graph, via neighborhood selection, and performed inference in RKH space. We compared this method to the nonparametric tree based structure learning, and used two different data type for our experiments: Angular and Distance data. We showed that a relevant kernel is important for inference, and through our experiments, showed that Neighborhood selection with Triangular kernel outperforms nonparametric Tree structure learning method, 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. As we will see, in section 3.2.5 there are several possibilities to investigate in order to make the method more scalable.

Another problem with the current RKHS model is that, even though neighborhood selection provides initial method to estimate the hidden structure in the data, the linear regression imposes a very strict constraint on the structure, and this might hurt the strength of the fully nonparametric RKHS embedded inference. In section 3.2.4, we will also propose to improve the structure learning algorithm of the model, by taking advantage of a variety of techniques from sparse kernel learning to nonparametric regression method and Tree ensamble learnings.

3.2.4 Proposed Work: Structure Learning for RKHS-Embedded Graphical Models

As we covered earlier, Neighborhood selection[52] is a consistent and robust method for structure learning. In RKH Space, it outperforms the other existing method for structure learning. However, neighborhood selection is based on linear regression, which is unreasonable when the rest of the model is in kernel space. There are currently a variety of methods available for performing kernel based and nonparametric regression:

In [35], Lafferty and Wasserman proposed a sparse non-parametric regression, called $Rodeo$, which is based on a series of hypothesis test, via non parametric kernel regression, and selects a subset of variables according to a threshold. Alternatively, Huang et. al.[27] assumes the regression to be via additive components, and estimates sparse component selection based on group Lasso. Lin et. al. [41], on the other hand, uses smoothing splines to perform variable selection. In this thesis we propose to investigate and apply these kernel based sparse regression methods to improve the quality of our structure estimation model in RKH space. Also many sparse additive models have been developed [**?**], which can be used.

Another family of methods for structure learning is the Ensemble of Tree graphical models. Lin et. al. [42] develop a model which is a mixture model over all possible tree structures, and under certain parametric graphical models, the final partition function has a closed form solution. As we saw in section 2.4.2, forest density estimation is another possible structure estimation

method, based on a collection of sub-trees[36]. In this thesis, we propose to investigate these structure learning methods as well, and use them to improve the structure learning for Kernel-space embedded graphical models.

3.2.5 Proposed Work: Scaling Reproducing Kernel Hilbert Space Models to Large Datasets

Several classic methods, such as Incomplete Cholesky Decomposition[**?**], or random features[64] can provide a solution for the scalability issue of nonparametric graphical models.

The Incomplete Cholesky Decomposition method uses Gram-Schmidt orthogonalization procedure to factor the Kernel matrix into a product of lower dimensional components, and then proceeds by approximating these components using fewer pivot data samples. Song et. al. [76] use this method and select the pivot samples via a greedy algorithm. They approximate the feature matrix Φ, as a $weighted$ combination of a subset of its column. (i.e. $\Phi \approx \Phi_I W_t$ where $I := \{i_1, ..i_l\} \subseteq \{1, .., m\}$). Φ_I is a sub-matrix of the Φ. This approximation allows the kernel matrix to have a low rank factorization(i.e. $K = W_t^T K_{II} W_t$) and be computed efficiently. Song et. al. use Gram-Schmidt orthogonalization procedure to select a basis set and define Φ_I and W_t. They also approximate the tensor product, to get a constant time message update.

When dealing with continuous variables and Gaussian kernels, Rahimi et. al. [64] develop random mappings, that map feature vectors of a shift-invariant kernel function, such as Gaussian, into a lower dimensional space, and approximates the function evaluations, $f(x)$, as a linear product in the lower dimension space, rather than the original sample space.

Their proposed mapping is based on the Fourier components of the kernel, and approximates the new feature map, $z(x)$, (i. e. $z(x)z(y) \approx k(x-y)$), after sampling D iid samples $\{w_1, .., w_D\}$, uniformly, from Fourier transformation of the kernel. (i.e. $w_i \sim (2\pi)^{-D/2} e^{-frac||w||_2^2 2}$, for Gaussian Kernel). Given the samples, the projected feature maps are defined as

$$z(x) = \sqrt{\frac{1}{D}[cos(w_1' x)...cos(w_D' x) sin(w_1' x)...sin(w_D' x)]'}$$

In another research, Shi et. al. [71], introduce $hash$ kernels. Hash kernel are built for high dimensional feature spaces, such as string kernel features, where the features are usually words in the whole corpus. In hash kernels, a hash function is used to approximate the large feature vector into a lower dimensional feature vector (i.e. $\overline{K}(x, x') = \langle \overline{\phi}(x), \overline{\phi}(x') \rangle$). The lower dimensional feature elements $\overline{\phi}_j(x)$ are calculated as the sum of a subset of the original feature elements. This subset is the set of all indexes which the hash function maps into j. (i.e. $\overline{\phi}_j(x) = \sum_{h(i)=j} \phi(x)$). This technique is more suitable for discrete and high dimensional feature spaces.

In this thesis, we propose to apply both methods of sampling, and kernel approximations, to improve the scalability of the RKHS graphical models. Specifically, the challenge associated with RKHS graphical models in high dimension and with large sample sets is that, the kernel matrices for each dimension should be calculated independently, and even by using current scalability solutions we will still recalculate and re-factorize each matrix independently. In this thesis we propose to perform a simultaneous factorization of the kernel matrices for all dimensions, via incomplete Cholesky decomposition, and we propose to use both random feature AND sample selection in our solution.

Chapter 4

Structure and parameter learning and inference for dynamic domain

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 chapter, in section 4.1 we will describe a sparse time-varying Gaussian graphical model, that we apply to analysis of protein dynamics of CypA enzyme. In section 4.2, we will propose to improve the time-varying graphical model by allowing the underlying model to be non-parametric. In section 4.3, we propose to develop a time-varying model that clusters the samples from the landscape before learning the time-varying model, using Dirichlet process for single simulations, and hierarchical Dirichlet process for multiple related simulations.

4.1 Time varying Gaussian Graphical Models

In this section, 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.

4.1.1 Introduction

A system's ability to visit different sub-states is closely linked to important phenomena, including enzyme catalysis[7] and energy transduction[38]. 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[61] and Desmond[10]), distributed computing (e.g., Folding@Home[58]), and specialized hardware (e.g., the use of GPUs[79] and Anton[70]). 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.

4.1.2 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[30] [39], 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[44] [45]. 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[44], or a Gaussian distribution[45] 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.

4.1.3 Regularized Time-Varying Gaussian Graphical Models

Zhou et. al. [90] 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 4.1).

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.

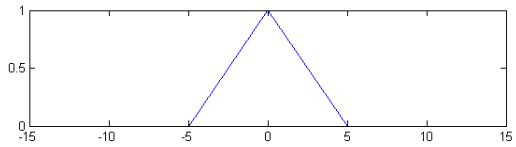


Figure 4.1: The Kernel functions of triangular family used in our experiment. $K = 1 - \frac{|x|}{5} * 1_{\{|x|<5\}}$

4.1.4 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. Here's the algorithm:

==========

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

Repeat until convergence

1. For $j = 1, \ldots n$

$1(a)$ $y^* = \arg\min_y \{y^T W^{(j-1)}_{\backslash j \backslash j} y : \|y - S_j\|_\infty \leq \lambda\}$

Where $W^{(j-1)}$ denotes the current iterate.

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

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

3. Test for convergence when the $W^{(0)}$ satisfies:

$trace((W^{(0)})^{-1} S) - n + \lambda \|(W^{(0)})^{-1}\|_1 \leq \epsilon.$

==========

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[84].

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.

4.1.5 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.

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.

Model Selection

We used the imputation method, as previously mentioned in 3.1.6 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.

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 4.2 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.

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

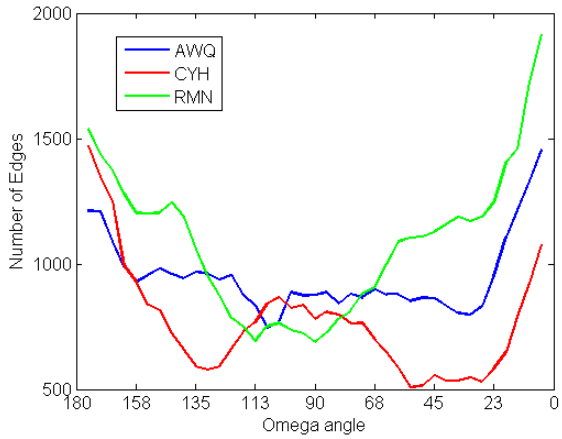


Figure 4.2: Edge Density Along Reaction Coordinate. The number of edges learned from the three MD simulations of CypA in complex with three substrates (AWQ, CYH, and RMH) are plotted as a function of the ω angle. AWQ is the largest substrate, CYH is the smallest substrate.

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.

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 4.3. 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.

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

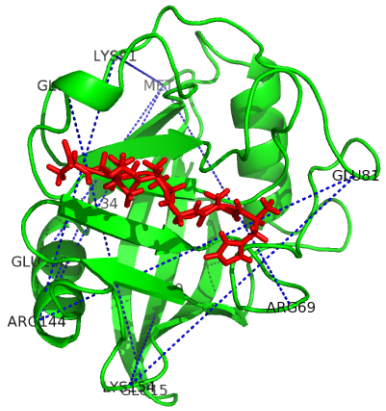


Figure 4.3: Top 10 Persistent Edges. For simplicity, only the top 10 couplings are shown.

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.

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 4.4. 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.

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 4.5. 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.

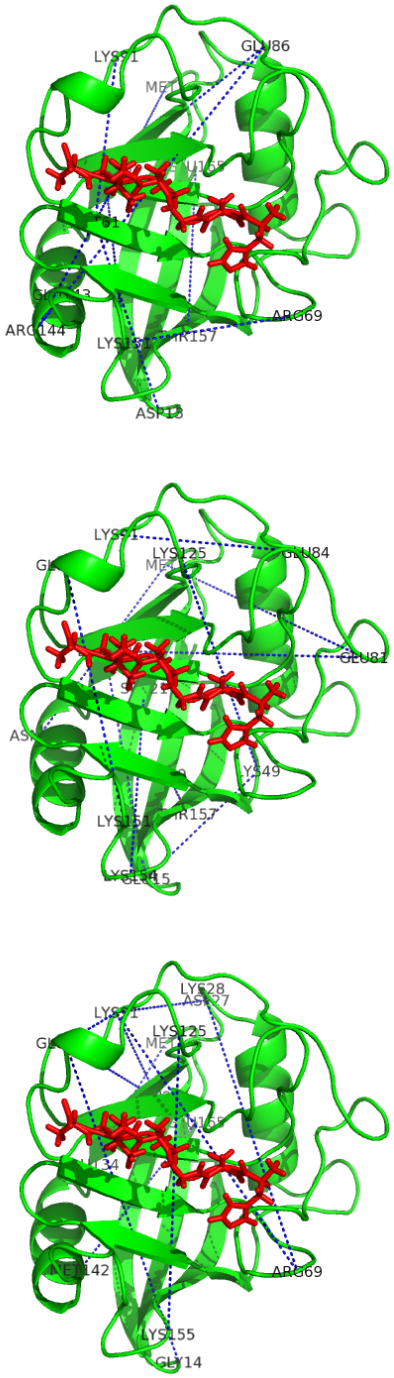


Figure 4.4: Transient Edges. The set of edges seen exclusively in the *trans* (top), transition (middle), and *cis* (bottom) states, respectively. For simplicity, only the top 10 couplings are shown.

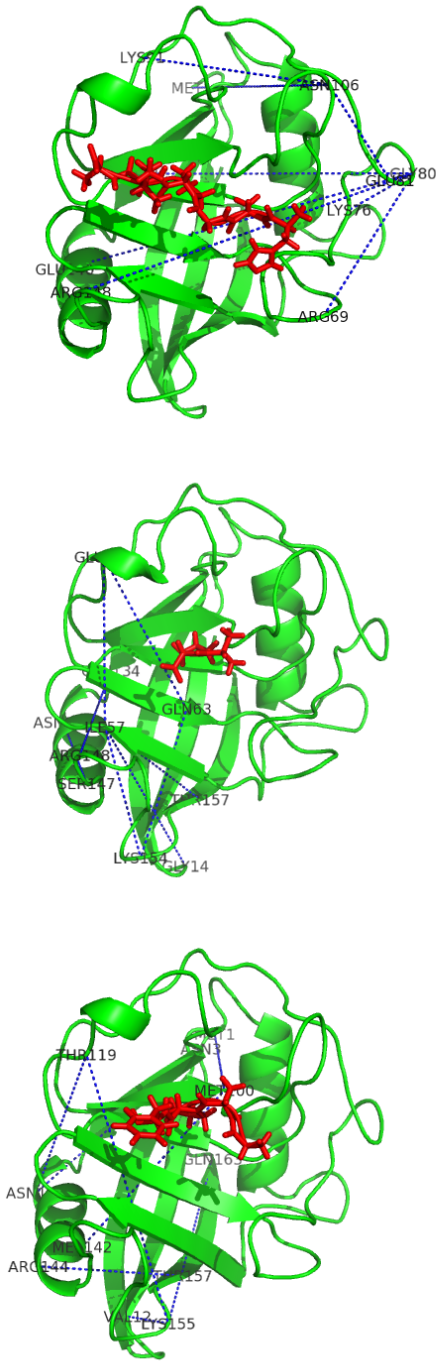


Figure 4.5: Substrate-specific Edges. The set of edges seen exclusively in the AWQ (top) CHY (middle), and RMH (bottom) substrates. For simplicity, only the top 10 couplings are shown.

4.1.6 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 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. 4.2, 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[88]. 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. 4.4). 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. 4.5). 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 section 4.2 we will describe our proposed solution to take advantage of more accurate graphical models in our 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. In section 4.3 we discuss our proposed model to adjust our kernel length accordingly, using nonparametric clustering via Dirichlet processes, befor optimizing the weighted log likelihood.

4.2 Proposed Work: Time varying nonparametric graphical models

Zhou et. al. [89] provide formulations to model a smoothly varying graphical model, via weighted log likelihood. As we saw in section 4.1, for Gaussian graphical models sparse parameter estimation is very efficient, but Gaussian graphical models do not fit most datasets. We propose to improve the graphical model by using non-paranormal graphical models as reviewed in chapter 2.4.1. The non-paranormal graphical model can be optimized via log likelihood analytically, and has the advantage that the data is in feature space, and thus we are able to model more sophisticated graphical models.

Time permitting, we will also develop models that take advantage of other non-parametric graphical models (i.e. RKHS graphical models) which we have worked on in the static domain.

[talk about the experiments and datasets]

4.3 Proposed Work: Dirichlet process and hierarchical Dirichlet process for time varying graphical models

As mentioned in section 4.1, current models of time varying graphical models provide many options for re-weighting of the data, to optimize the weighted log likelihood. In modeling of protein folding dynamics, each span of the samples corresponds to one local minima in the free energy landscape, which is important to locate and analyze.

Previously in [**?**], after using fixed-length symmetric kernel to perform structure learning (as in section 4.1.3), we have used linkage clustering algorithm to cluster different graphical models learned for different time frames, and then we have built a state-transition matrix on top of the clusters, to be able to understand the energy landscape. We have selected the number of our clusters heuristically, 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.

As the next step, we propose to perform the clustering of the data non-parametrically, based on Dirichlet process (DP) models [14], so as to solve the issue of model selection. Dirichlet processes are probabilistic processes that allow potentially infinite clusters[65] to be defined in the model, but for a given dataset, will converge to a finite set of clusters. We have previously investigated Dirichlet processes and hierarchical Dirichlet processes [69], and propose to create local optima sub-state and transition sub-state models, based on the initial clustering of the data, and weighted log likelihood.

Describing transition states in the folding is generally difficult, due to the fact that the high (i.e. non optimal) energy of the state makes the protein very unstable at the transition state, and we can collect fewer samples in these states. As an example, Figure 4.6 shows the free energy of Engrailed protein during one folding experiment, in two sub-states and one transition state. In this thesis, we propose to estimate the transition states using weighted log likelihood, where the weights are calculated based on the $likelihood\ ratio$ of each sample, under the two consecutive sub-state models. We propose to perform our experiments on three MD simulation datasets for Engrailed protein folding, each in a different temperature, and analyze the results.

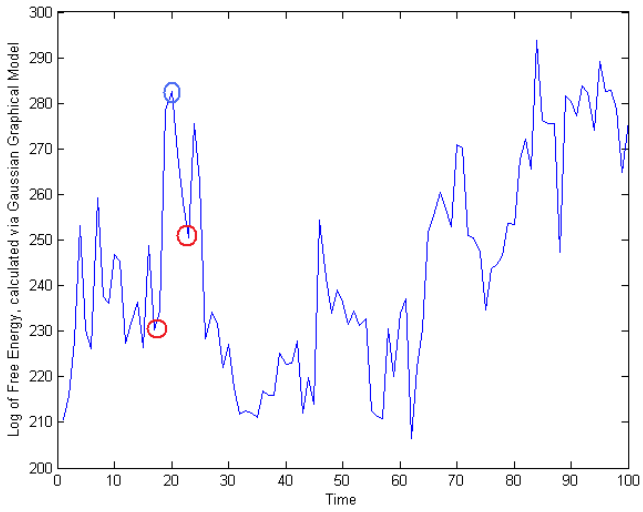


Figure 4.6: An example of transition states, in protein folding trajectory. Red circles indicate two local optimal sub-states, and the blue circle indicates the transition state between them.

Most methods of Protein folding energy landscape analysis, try to model a single folding trajectory for protein at a time. However, it is usually the case where the protein can be simulated in multiple temperatures and under different conditions such as in presence or the absence of a ligand. Hierarchical Dirichlet processes[81] allow the clustering algorithm to learn different DP-based clustering models (i.e. multiple sub-states in the protein folding process) for each folding trajectory, such that we take advantage of the fact that many sub-states are shared between these trajectories. In the final stage of this thesis we propose to apply hierarchical Dirichlet process models to discover protein folding sub-states from multiple datasets of folding trajectories.

Chapter 5

Time Line

October 2012 - January 2013 - Completion of (1) Structure learning for reproducing kernel Hilbert space embedded graphical models and related experiments, and (2) Scalability improvement for nonparametric graphical models.

January - May 2013 - Completion of (1) time-varying nonparametric graphical models, (2) [Time-permitting] Dirichlet process and hierarchical Dirichlet process clustering for time varying graphical models of multiple folding trajectories, and (3) Job search.

May - August 2013 - Thesis writing, and defense.

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. 4.1.6

[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. 4.1.5

[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. 4.1.5, 4.1.5

[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. 4.1.4, 4.1.4

[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, 2.3.2, 2.4.1, 3

[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. 4.1.2

[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. 4.1.1

[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. 2.3.3

[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. 4.1.5

[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. 1, 4.1.1

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. 3.1.3

[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. 2.3.3

[13] 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. 2.4.3

[14] J B Macqueen D Blackwell. Ferguson distributions via polya urn schemes, 1973. 4.3

[15] Hal Daumé III. From zero to reproducing kernel hilbert spaces in twelve pages or less. Available at http://www.isi.edu/~hdaume/docs/daume04rkhs.ps, February 2004. 2.4.3

[16] 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. 3.1.6

[17] Joshua Dillon and Guy Lebanon. Statistical and computational tradeoffs in stochastic composite likelihood. 2009. 3.1.3

[18] Finale Doshi, David Wingate, Joshua B. Tenenbaum, and Nicholas Roy. Infinite dynamic bayesian networks. In *ICML*, pages 913–920, 2011. 2.5

[19] N.I. Fisher. *Statistical Analysis of Circular Data*. Cambridge University Press, 1993. 1, 2.3.3

[20] 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, 2.3.2

[21] 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. 3.1.6

[22] 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. 3.1.6

[23] Zoubin Ghahramani. Learning dynamic bayesian networks. In *Adaptive Processing of Sequences and Data Structures*, pages 168–197. Springer-Verlag, 1998. 2.5

[24] 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. 2.3.1

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

2.3.3, 3.1.1, 3.1.2

[26] 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. 3.1.3

[27] Jian Huang, Joel L. Horowitz, and Fengrong Wei. Variable selection in nonparametric additive models. *Journal of Annual Statistics*, 3:2282–2313, August 2010. 3.2.4

[28] 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: W&CP 5. 2.1

[29] 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. 2.3.1

[30] M. Karplus and J. N. Kushick. Method for estimating the configurational entropy of macromolecules. *Macromolecules*, 14(2):325–332, 1981. 4.1.2

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

[32] Seyoung Kim, Kyung-Ah Sohn, and Eric P. Xing. A multivariate regression approach to association analysis of a quantitative trait network. *Bioinformatics*, 25(12):i204–i212, 2009. doi: 10.1093/bioinformatics/btp218. 2.5

[33] Mladen Kolar, Le Song, Amr Ahmed, and Eric P. Xing. Estimating time-varying networks. 2008. 2.5, 2.5

[34] 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`. 2.4.2, 2.4.3

[35] J. Lafferty and L. Wasserman. Rodeo: Sparse, greedy nonparametric regression. *Annual of Statistics*, 36(1):28–63, 2008. 2.4.2, 3, 3.2, 3.2.4

[36] J. Lafferty, H. Liu, and L. Wasserman. Sparse Nonparametric Graphical Models. *ArXiv e-prints*, January 2012. 1, 2.4.2, 3.2.4

[37] 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. 3.1.3

[38] David M. Leitner. Energy flow in proteins. *Annu. Rev. Phys. Chem.*, 59:233–259, 2008. 4.1.1

[39] 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. 4.1.2

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

2006. URL http://videolectures.net/mlss06tw_lin_svm/. 2.4.3

[41] Yi Lin and Hao Helen Zhang. Component selection and smoothing in multivariate nonparametric regression. *Journal of Annual Statistics*, pages 2272–2297, 2006. 3.2.4

[42] Yuanqing Lin, Shenghuo Zhu, Daniel Lee, and Ben Taskar. Learning sparse markov network structure via ensemble-of-trees models. In *AISTAT'09: The Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 360–367, 2009. URL http://jmlr.csail.mit.edu/proceedings/papers/v5/lin09a/lin09a.pdf. 3, 3.2, 3.2.4

[43] 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, 2.4.1

[44] 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. 4.1.2

[45] 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. 4.1.2

[46] K. V. Mardia. Statistics of directional data. *J. Royal Statistical Society. Series B*, 37(3): 349–393, 1975. 2.3.3, 3.1.1, 3.1.3

[47] 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. 2.3.3

[48] 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. 2.3.3

[49] K.V. Mardia and P.E. Jupp. *Directional statistics*. Wiley Chichester, 2000. 2.3.3, 3.1.5

[50] 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. 3.1.6, 3.1.6

[51] 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. 3.1.6, 3.1.6

[52] 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. 2.4.3, 3, 3.2, 3.2.1, 3.2.4

[53] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Uncertainty in Artificial Intelligence*, pages 362–369, 2001. 2.1, 3.1.4

[54] Kevin P. Murphy. Dynamic bayesian networks. 2002. 2.5

[55] 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

[56] E. Nadaraya. On estimating regression. *Theory of Prob. and Appl.*, 9:141–142, 1964. 2.4.3

[57] Hetu Kamisetty Arvind Ramanathan Christopher J. Langmead Narges Razavian, Subhodeep Moitra. Time-varying gaussian graphical models of molecular dynamics data. *Proceedings of 3DSIG 2010 Structural Bioinformatics and Computational Biophysics*, 2010. (document), 2.2

[58] 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. 1, 4.1.1

[59] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076, 1962. ISSN 00034851. 2.4.3

[60] 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, 3.1.4

[61] 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. 4.1.1

[62] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R.D. Skeel, L. Kale, and K. Schulten. Scalable molecular dynamics with namd. *J. Comp. Chem.*, 26:1781–1802, 2005. 1

[63] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957. 2.4.2, 2.4.3

[64] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *In Advances in Neural Information Processing Systems (NIPS*, 2007. 3.2.5

[65] C. E. Rasmussen. The infinite gaussian mixture model. In S.A. et al. Solla, editor, *Advances in information processing systems 12*, pages 554–560. MIT Press, 2000. 4.3

[66] Volker Roth. Sparse kernel regressors. 2130:339–346, 2001. 2.4.3

[67] Mark Schmidt, Glenn Fung, and Romer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. pages 286–297, 2007. 3.1.6

[68] 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. 3.1.3

[69] Narges Sharif-razavian and Andreas Zollmann. An overview of nonparametric bayesian models and applications to natural language processing, 2009. 4.3

[70] 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. 1, 3.1.6, 4.1.1

[71] Qinfeng Shi, James Petterson, John Langford, Alex Smola, and Alex Strehl. Hash kernels. In *Proc. Intl. Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics*, 2009. 3.2.5

[72] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *Algorithmic Learning Theory*. Springer, 2007. Invited paper. 2.4.3

[73] L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions. In *International Conference on Machine Learning*, 2009. 2.4.3

[74] L. Song, A. Gretton, and C. Guestrin. Nonparametric tree graphical models. In *Artificial Intelligence and Statistics (AISTATS)*, 2010. 1, 2.4.3, 2.4.3

[75] L. Song, M. Kolar, and E. Xing. Time-varying dynamic bayesian networks. In *Advances in Neural Information Processing Systems 22 (NIPS)*, 2010. 2.5, 2.5

[76] 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, 2.4.3, 2.4.3, 3.2.5

[77] L. Song, A. Parikh, and E. Xing. Kernel embeddings of latent tree graphical models. In *Neural Information Processing Systems (NIPS)*, 2011. 2.4.3

[78] 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`. 2.4.3

[79] 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. 1, 4.1.1

[80] 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, 2.3.4

[81] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. doi: 10.1198/016214506000000302. URL http://www.tandfonline.com/doi/abs/10.1198/016214506000000302. 4.3

[82] 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. 3.2.1

[83] JA Tropp. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory*, 52(3):1030–1051, 2006. 3.1.3

[84] 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. 4.1.4

[85] 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. 3.1.3

[86] 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. 2.4.3

[87] 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

[88] 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. 4.1.6

[89] Shuheng Zhou, John Lafferty, and Larry Wasserman. Time varying undirected graphs. 2008. URL http://arxiv.org/abs/0802.2758. 2.5, 4.2

[90] Shuheng Zhou, John D. Lafferty, and Larry A. Wasserman. Time varying undirected graphs. In *COLT*, pages 455–466, 2008. 4.1.3