The von Mises Graphical Model: Structure Learning

Narges Sharif Razavian[1],
Hetunandan Kamisetty[2],
Christopher James Langmead[2,3,*]

March 2011
CMU-CS-11-108
CMU-CB-11-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]Language Technologies Institute, [2]Department of Computer Science, [3]Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA, USA 15213.
*E-mail: cjl@cs.cmu.edu

This research was supported by NSF IIS-0905193

Keywords: Von Mises, Structure Learning, Generative Models, Probabilistic Graphical Models, L1-Regularization, Time-Varying, Proteins, Molecular Dynamics

Abstract

The von Mises distribution is a continuous probability distribution on the circle used in directional statistics. In this paper, 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 a simple inference algorithm based on Gibbs sampling. We compare and contrast the von Mises Graphical Model (vGM) with a Gaussian Graphical Model (GGM) on both synthetic data and on data from protein structures and demonstrate that the vGM achieves higher accuracy than the GGM.

1 Introduction

The von Mises distribution is used in directional statistics to model angles and other circularly-distributed variables [3]. It closely approximates the wrapped normal distribution, but has the advantage that it is more tractable, mathematically [9]. Additionally, the von Mises distribution can be generalized to distributions over the $(p-1)$-dimensional sphere in $\mathbb{R}^p$, where it is known as the von Mises-Fisher distribution.

The goals of this paper are to introduce an undirected graphical model for the Mises-Fisher distribution, and an algorithm for learning the structure and parameters of the model in a regularized fashion. We note that the von Mises distribution has been used previously in the context of *directed* graphical models. In particular, Harder and co-workers recently introduced a Dynamic Bayesian Network where each output variable follows a bivariate von Mises distribution[4]. Our model, in contrast, is undirected and isn't limited to modeling sequential data. Moreover, to the best of our knowledge, the challenge of learning the structure and parameters of probabilistic graphical models over von Mises distributed variables has only been examined in the contex of Neural Networks and Boltzmann Machines[14][15], in which unrestricted Boltzmann machines have been trained with univariate Von Mises distributions for the individual nodes. In contrast to these work we learn an L1-regularized graphical model of the form of a multivariate Von Mises distribution, such that the couplings are also modeled as directional variables.

The rest of the paper is organized as follows. In Section 2 we introduce the univariate and bivariate von Mises distributions. In Section 3 we present the multivariate von Mises distribution as an undirected graphical model. We then define a Gibbs sampler for the model in Section 4, which we use later for drawing samples and performing inference on such models. Section 5 presents our structure learning algorithm which employs L_1 regularization to learn the dependency relationships between the variables. We then compare and contrast the von Mises graphical model (vGM) to another popular graphical model, the Gaussian graphical model (GGM), in Section 6 on both synthetic circular, and real protein torsion angle data. We show that the vGM achieves higher accuracy than the GGM when each variable has high marginal variance.

2 Background

The wrapped normal distribution for angle $\theta \in [0, 2\pi]$ is defined as an infinite sum of the wrappings of a normal 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 [8] over θ_1 and θ_2, for example, can be defined as:

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

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, $\vec{K}_1 = [\cos(\theta_1 - \mu_1), \sin(\theta_1 - \mu_1)]$, $\vec{K}_2 = [\cos(\theta_2 - \mu_2), \sin(\theta_2 - \mu_2)]$, $\mathbf{M}$ 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:

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

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.

3 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 [8] 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) [5]. This is significant because learning and inference can be performed analytically for multivariate Gaussian distributions. However, we will show in Section 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 1 shows the factor graph representation of the graphical mode for four variables. Under this representation the node factors are defined as

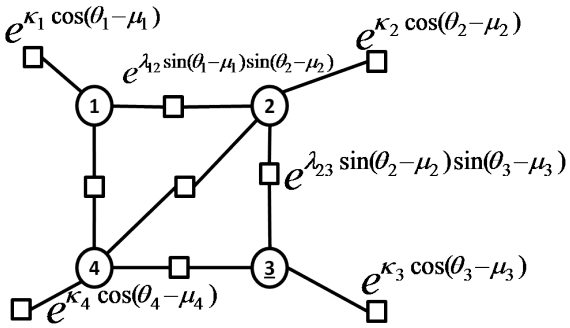

Figure 1: Factor Graph Representation for multivariate von Mises distribution. Each circular node is a variable, and the square nodes are factors.

$f_i = \kappa_i cos(\theta_i - \mu_i)$ 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.

4 Sampling

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. We note that it is possible to perform inference in the vGM using an Expectation-Propagation style algorithm, which we will present in a future publication due to space considerations. In this paper, we will instead use a Gibbs sampler to perform approximate inference.

Gibbs sampling assumes that it is easy to sample from the univariate conditionals. Fortunately, as shown in [5] the univariate von Mises $conditionals$ are $univariate$ von Mises distributions themselves, and this makes Gibbs sampling a feasible option. In particular

$f(\theta_p|\theta_1, \theta_2, \ldots \theta_{p-1}) \propto$

$$\begin{aligned} &\exp\left\{\kappa_p cos(\theta_p - \mu_p) + \sum_{j=1}^{p-1} \lambda_{jp} sin(\theta_j - \mu_j) sin(\theta_p - \mu_p)\right\} \\ =&\exp\left\{\kappa^* cos(\theta_p - \mu^*)\right\}, \end{aligned}$$

where

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

$$\mu^* = \mu_p + \arctan(\frac{1}{\kappa_p} \sum_{j=1}^{p-1} \lambda_{jp} sin(\theta_j - \mu_j)) \tag{2}$$

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

5 Learning

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. Given a set of $i.i.d.$ training samples, $D = \{\Theta_1, \Theta_2, \cdots \Theta_n\}$, the likelihood function is:

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

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.

5.1 Full pseudo-likelihood for von Mises Graphical Model

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

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

As discussed in section 4, each univariate conditional term for the vGM is itself a univariate von Mises distribution. Thus, the full pseudo likelihood can be re-written as:

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

where

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

$$\kappa_{\setminus j}^{(i)} = \sqrt{\kappa_j^2 + (\sum_{l\neq j}\lambda_{j,l}sin(\theta_{i,l}-\mu_l))^2}.$$

5.2 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 [2]. This property does hold true for von Mises.

Proof sketch: Consider two conditionals which have different parameters ($\vec{\kappa}_1^*$ and $\vec{\kappa}_2^*$, and $\vec{\mu}_1^*$ and $\vec{\mu}_2^*$), but have the same joint distribution. Then, by taking the derivative of the two conditionals based on Θ, and equating the two derivatives, it can be shown that $\vec{\kappa}_1^*$=$\vec{\kappa}_2^*$, and $\vec{\mu}_1^*$=$\vec{\mu}_2^*$. From the equality of the $\vec{\kappa}^*$ and $\vec{\mu}^*$, we can then form a system of equations using the definitions (1) and (2). The system has a single solution in which the vGM parameters $\vec{\kappa}$ and $\vec{\mu}$ are equal. Thus, based on the theorem discussed in [2], the Full Pseudo Likelihood is a consistent estimator for the vGM.

5.3 Structure learning for vGM

When the topology of the graph is not given or known, we must also learn the structure of the model, as well as the parameters. The study of the so-called structure learning problem has received considerable attention recently (e.g., [13, 7, 6, 11]). Structure learning algorithms based on L_1 regularization are particularly interesting because they exhibit consistency and high statistical efficiency (see [12] for a review). We use an algorithm introduced by Schmidt *et.al* [11] 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. [11] 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.

In the rest of this section, the notation x^+ refers to $max(x, 0)$ and x^- refers to $-min(0, x)$. Having defined a differentiable approximation for absolute values, the learning problem can be re-written as finding the optimal solution to:

$$min_{x^+,x^-} L(x^+ - x^-) + \rho \sum_{i=1}^{n} [x_i^+ - x_i^-]$$
$$s.t. \forall\, i, x_i^+ \geq 0, \text{and } x_i^- \geq 0$$

This objective can be minimized using projected gradients. Briefly, projected gradients optimize variables with inactive constraints. If $x^* = [x_i^+ x_i^-]^T$, the set of active constraints is defined as: $\{i | x_i^* = 0, \nabla L(x_i^+ - x_i^-) + \lambda > 0\}$. The algorithm takes gradient descent steps of the form $x^* := [x^* - t\nabla L(x_i^+ - x_i^-)]^+$ where t represents the step size. We note that methods based on projected gradients are guaranteed to converge to a stationary point [1].

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 5.1:

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

where

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

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

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

Here, $A_0(\kappa)$ is defined as $\frac{I_1(\kappa)}{I_0(\kappa)}$ as described in [8].

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

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

such that

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

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.

6 Experiments

We have implemented the Gibbs sampler and structure learning algorithm for the vGM and used them to perform experiments using both synthetic and real data. The synthetic data were generated using our Gibbs sampler on randomly generated vGM models. The accuracy of our structure learning algorithm was evaluated by comparing to ground truth and to the structures learned using an algorithm for learning Gaussian Graphical Models (GGM). The real data come from a molecular dynamics (MD) simulation of the protein ubiquitin. A MD simulation involves integrating Newton's laws of motion for a set of atoms (in our case, those of the protein and the surrounding water molecules). The result is a time-series of conformational snapshots of the protein. The goal is to learn a model of the joint distribution over the protein's backbone torsion angles (which collectively determine the three dimensional structure of the molecule). The accuracy of our learning algorithm was evaluated on the MD data using cross-validation and compared to the GGM learning algorithm.

6.1 Parameter Learning on Synthetic Data

We generated random vGM graphs for different parameter configurations by systematically varying the followings: (a) the number of nodes of graph from 8 to 128; (b) the density of edges of the graph from 0.1% to 100%; and (c) the von Mises parameters $\vec{\kappa}$ and Λ. For each parameter configuration, we generated 50 vGMs by randomly generating the elements of $\vec{\kappa}$ using a uniform distribution on $[0, S_\kappa]$. Here, S_κ ranged from 10^{-2} to 10^2. Elements of the Λ matrix were drawn from a Gaussian distribution $\mathcal{N}(0, S_\Lambda)$ where S_Λ ranged from 10^{-2} to 10^2. 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. 4) to randomly generate 50 data sets from each of the randomly generated vGM configurations. Each dataset contained 3000 fully observed samples. Next, we used our structure learning algorithm (Sec 5) to learn a vGM from each data set. For comparison, we also used the structure learning algorithm presented in [10] to learn a GGM from the same data.

Evaluation Metrics We used two metrics to compare the results of the two structure learning algorithms. First, the average $F1$-score was computed for each randomly generated vGMs, where the average is taken over the 50 data sets. The $F1$-score quantifies the ability of each learning algorithm to identify the true correlations in the data. It is defined as $\frac{2.precision*recall}{precision+recall}$.

The second metric was the $cosine$ of the angle between the true and estimated model parameters ($\vec{\kappa}$ and Λ). The cosine of angle between vectors A and B is defined as $\frac{A^T B}{\|A\|_2 \|B\|_2}$ and values closer to one indicate higher similarity. We used this metric as an indicator of the quality of algorithm in learning not only the structure, but the strength of the links in the graph. The Λ matrix was first reshaped to be a vector, and the $kappa$ values were also appended to this vector to create one single vector of parameters. Then the cosine of the angle between this vector and the true parameter vector was computed.

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. The same procedure was used to find the penatly for the GGM.

Results Figures 2 through 4 present surface plots depicting the cosine angles between the true and learned parameters for varying edge densities. In each figure, the x and y axes correspond to the log of S_κ and S_Λ (defined above), while the z axis and the color is the average cosine angle between the true and learned parameters, averaged over the 50 data sets of that configuration.

At very low edge density (Fig. 2), the variables are mostly independent and the algorithm successfully learns the $\vec{\kappa}$ values over all combinations of the true values of $\vec{\kappa}$ and Λ. At 50% and 100% edge density (Figs. 3 and 4), the effect of the magnitude of the κ and Λ values becomes

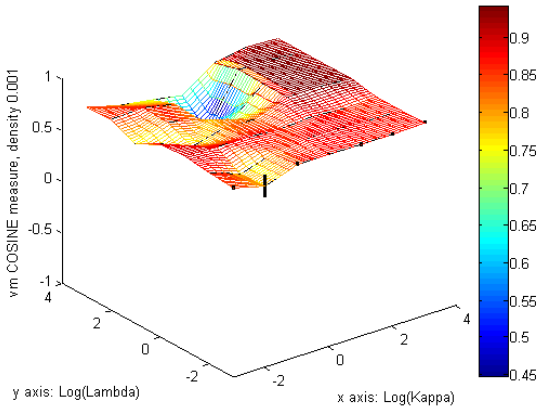

Figure 2: Cosine angles between true and learnt parameters at 0.1% edge density (i.e. mostly independent variables). Standard error bars are shown as black bars.

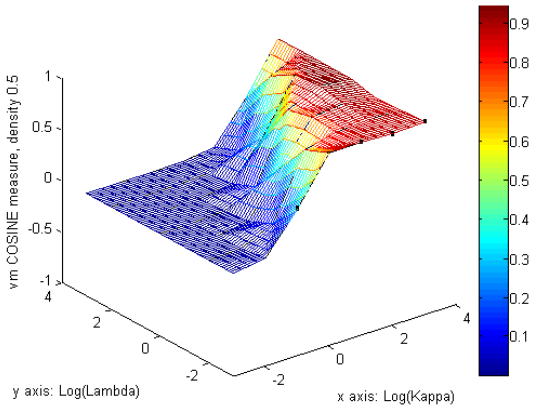

Figure 3: Cosine angles between true and learnt parameters at 50% edge density.

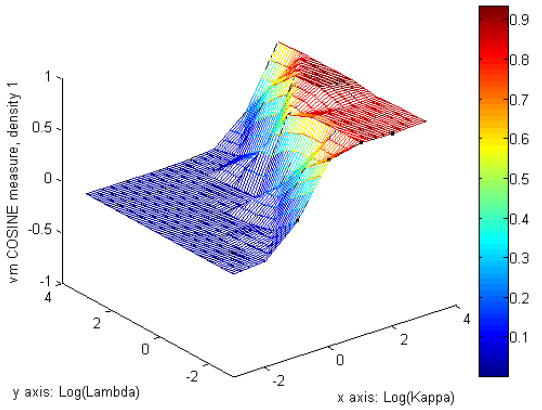

Figure 4: Cosine angles between true and learnt parameters at 100% edge density (i.e., fully connected).

evident. In particular, accuracy is positively correlated with the magnitude of the κ's and inversely correlated with the magnitude of the λ's. Recall that κ is inversely related to the marginal variances, and that the elements of the Λ matrix correspond to the strength of the coupling/correlation between variables. Thus, accuracy of the learning algorithm decreases under high variance, and/or strong couplings.

Comparison to Gaussian Graphical Model As previously mentioned (Sec. 3), a vGM can be well-approximated with a GGM when the variables have low variance (i.e., high values of 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:

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

and

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

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

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

and

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

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, as follows:

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

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

We ran the GGM regularized learning algorithm [10] to determine if it has lower accuracy than the vGM when the variance of the variables is higher.

Figure 5 shows the average *difference* in the performance of the two learning algorithms for different parameter combinations (S_κ and S_Λ) at a fixed density of 50%. Each point in the plot is calculated by computing the cosine of the angle between the true and estimated model parameters obtained using the vGM algorithm minus the same quantity for the GGM algorithm. Thus, the peak in the contour plot corresponds to parameter combinations where vGM outperforms the GGM the most.

Interestingly, the surface in Figure 5 is not monotonic. In the lower right corner, where the concentration is high (i.e., low variance) *and* the coupling between variables is low, the GGM's performance is essentially the same as the vGM. This is expected. However, in the upper right

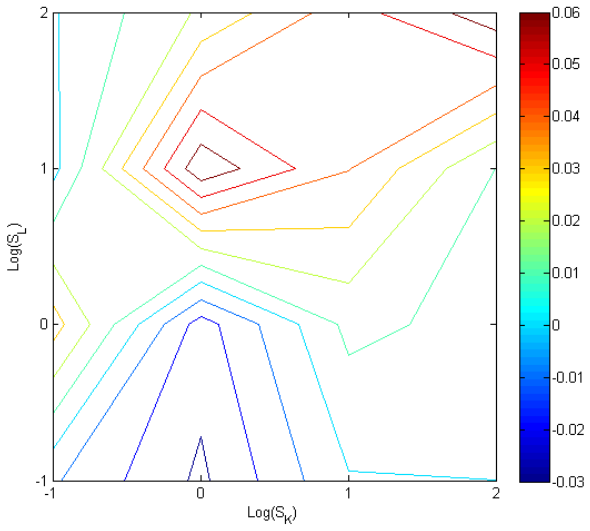

Figure 5: Performance of the vGM learning algorithm versus the GGM learning algorithm for different parameter combinations. See text for details.

corner, we see that the vGM can outperform the GGM when variance is low, provided that the average coupling is high. The peak in the plot occurs at approximately the point where $\log S_\kappa$ is zero and $\log S_\Lambda$ is one. Thus, the relative performance of the vGM increases as variance increases. But this is only true up to a point. At very low concentrations (i.e., high variances) the performance of both algorithms is about the same (left-most edge). Note that the GGM outperforms the vGM at approximately the point where $\log S_\kappa$ is zero and $\log S_\Lambda$ is negative one, but that the depth of the trough (≈ -0.03) isn't as deep as the hight of the peak (≈ 0.06). Thus, we conclude that the vGM performs as well, or better than the GMM over the majority of the parameter combinations we considered.

We also compared the accuracy of the two algorithms according to the F1-score, to determine if either does a better job at learning the structure of the model. We did not discover any specific pattern in the F1-scores. Thus, there is no evidence that the vGM algorithm outperforms GGM algorithm on in terms of learning the structure of the graph.

6.2 Parameter Learning Cross Validation on Protein Torsion Angle Data

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 (aka torsion) angles. For example, Figure 6 depicts a toy protein consisting of two amino acids (real proteins have dozens to thousands of amino acids). The dihedral angles in this figure are denoted using the conventional names used in biochemistry: ϕ, ψ, ω, χ_1, χ_2, χ_3, and χ_4. Unlike the figure, a protein's structure isn't static. Rather, each protein samples from an underlying distribution over configu-

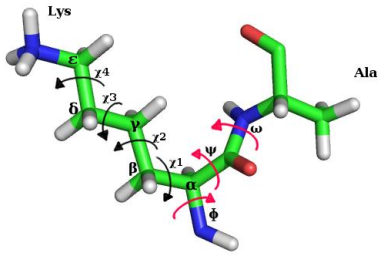

Figure 6: Backbone and side-chain dihedral angles of a di-peptide Lys-Ala protein.

rations (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. Thus, a vGM is a suitable choice for modeling a protein's structure.

We applied our von Mises graphical model learning algorithm to learn a model of the joint distribution over a subset of the dihedral angles in the protein ubiquitin, which has 76 amino acids. The data set consisted of 15000 observations obtained via molecular dynamics simulation. Each observation consists of a vector of dihedral angles defining the structure of the protein. We learned a vGM of these data using our structure learning algorithm. Model selection for selecting the regularization penalty was performed as described in Section 6.1.

To learn the data, we performed a 5-fold cross validation, each time learning the vGM model on 12000 samples, and computing the probability of the remaining 3000 structures given the existing model. Since the full joint probability calculation requires inference, we compared the probability of each variable conditioned on the rest of the observations, given the learned parameters. This conditional probability was calculated using the formula derived in section 4.

We also learned a Gaussian Graphical Model on the same protein data, using the algorithm described in [10] and we performed the same cross validation procedure as with the vGM .

Figure 7 shows the log probability of each dihedral angle in Test set under the model learned on the Training set. Each dot in the plot corresponds to one dihedral angle log probability. For a large number of dihedral angles, the log likelihood of the test set under vMM is higher than their likelihood under GGM. There is also a large number of positions where both models perform comparably – these correspond to positions in the protein that are fairly constrained and do not fluctuate significantly. This echoes the intuition that when the angles are relatively constrained (i.e. have low variance), both von Mises and Gaussians have similar behavior. This result also demonstrates that overall the vMM is a better fit of directional data than the GMM with significant benefits when the variables have high variance.

Figure 8 shows the analog of the precision matrix of the model discovered using the vGM learning algorithm. Figure 9 overlays these edges on the the a depiction of the structure of ubiquitin. Notice that there are many edges between distant parts of the protein. While beyond the scope of this paper, we note that such long-range dependencies are consistent with the biological function of the protein which binds to other molecules via a gripping motion.

Apart from its utility as a generative model, the parameters of the vGM model provide a suc-

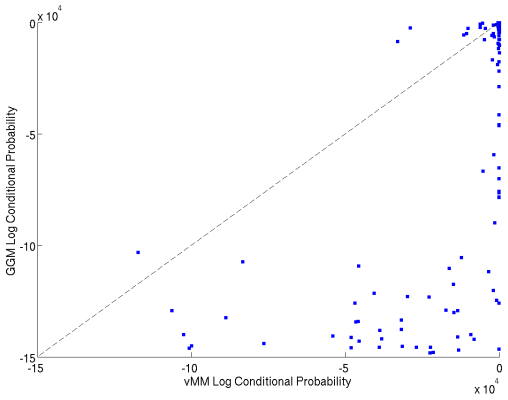

Figure 7: The log conditional probability of each variable under the vGM and GGM estimated model

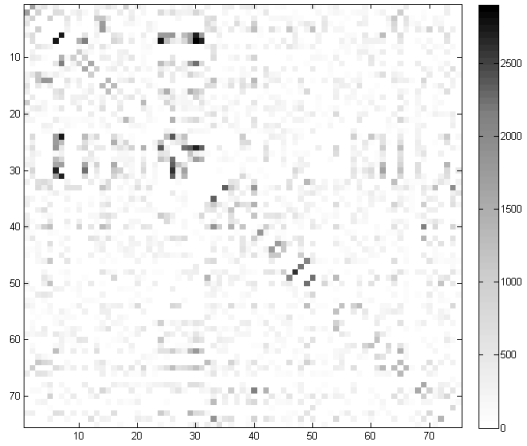

Figure 8: The network of couplings discovered by the vGM algorithm between Ubiquitin residues

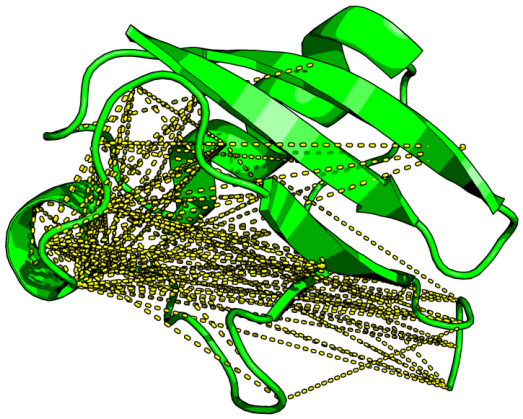

Figure 9: The dependency links between the torsion angles of the Ubiquitin Protein backbone.

cinct summary of the relations between the various angular fluctuations observed in the simulation, and this can then be used as a starting point by biologists to gain insights into the working of the protein.

7 Conclusion and Future Work

In this paper we presented the first multivariate von Mises graphical model and introduced algorithms for sampling and structure learning. 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, despite being a better fit for the circular data. We used a gradient based algorithm to estimate the parameters of such graphical models from data, and we showed the consistency of the maximum full pseudo likelihood estimator for Von Mises models.

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. Here we used two measures: F1 to evalute the accuracy of the estimator to learn the structure of the graph (i.e. the dependency network), and the Cosine of the angle between parameter vectors, to evaluate the parameter values. Von Mises model did not outperform the Gaussian model under F1 measure. However in the parameter estimation, we 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 protein ubiquitin. We computed the conditional probabilities of each variable conditioned on the rest of the angles of each sample in the the test set, after learning the parameters of vGM and GGM from the training set. We observed

that Von Mises is a better fit for the data, and can recover long distance dependencies between the movements of residues.

Finally, we note that we have recently derived the update equations for an Expectation-Propagation style inference algorithm for the vGM. It was not presented here due to space limitations. We intend to present that algorithm in a future publication.

Acknowledgements

We would like to thank the members of the Langmead lab for helpful comments and suggestions.

References

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

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

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

[4] 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.

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

[6] 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.

[7] 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.

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

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

[10] Mark Schmidt, Glenn Fung, and Rmer Rosales. Fast optimization methods for l1 regularization: A comparative study and two new approaches. In *In Proceedings of European Conference on Machine Learning*, pages 286–297, 2007.

[11] 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.

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

[13] 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.

[14] Richard S. Zemel, Christopher K. I. Williams, and Michael Mozer. Directional-unit boltzmann machines. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 172–179, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[15] Richard S. Zemel, Christopher K. I. Williams, and Michael C. Mozer. Lending direction to neural networks. *NEURAL NETWORKS*, 8:503–512, 1995.