Learning Generative Models for Protein Fold Families

Sivaraman Balakrishnan[*†] Hetunandan Kamisetty[†‡]
Jaime. G. Carbonell[*‡§] Su-In Lee[¶] Christopher James Langmead[‡§‖]

Abstract

We introduce a new approach to learning statistical models from multiple sequence alignments (MSA) of proteins. Our method, called GREMLIN (Generative REgularized ModeLs of proteINs), learns an undirected probabilistic graphical model of the amino acid composition within the MSA. The resulting model encodes both the position-specific conservation statistics *and* the correlated mutation statistics between sequential and long-range pairs of residues. Existing techniques for learning graphical models from multiple sequence alignments either make strong, and often inappropriate assumptions about the conditional independencies within the MSA (e.g., Hidden Markov Models), or else use sub-optimal algorithms to learn the parameters of the model. In contrast, GREMLIN makes no *a priori* assumptions about the conditional independencies within the MSA. We formulate and solve a *convex* optimization problem, thus guaranteeing that we find a *globally optimal* model at convergence. The resulting model is also generative, allowing for the design of new protein sequences that have the same statistical properties as those in the MSA. We perform a detailed analysis of covariation statistics on the extensively studied WW and PDZ domains and show that our method out-performs an existing algorithm for learning undirected probabilistic graphical models from MSA. We then apply our approach to 71 additional families from the PFAM database and demonstrate that the resulting models significantly out-perform Hidden Markov Models in terms of predictive accuracy.

[*]Language Technologies Institute Carnegie Mellon University, Pittsburgh, PA
[†]These two authors contributed equally to this work.
[‡]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA
[§]Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA
[¶]Department of Computer Science & Engineering and Department of Genome Sciences, University of Washington
[‖]Corresponding author. 5000 Forbes Ave., Pittsburgh, PA 15213. Phone: (412) 268 7571. Fax: (412) 268 5576. E-mail: cjl@cs.cmu.edu

1 Introduction

A protein family[1] is a set of evolutionarily related proteins descended from a common ancestor, generally having similar sequences, three dimensional structures, and functions. By examining the statistical patterns of sequence conservation and diversity within a protein family, we can gain insights into the constraints that determine structure and function. These statistical patterns are often learned from multiple sequence alignments (MSA) and then encoded using probabilistic graphical models (e.g., [1, 2, 3, 4, 5]). The well-known database PFAM [4], for example, contains more than 11,000 profile Hidden Markov Models (HMM) [6] learned from MSAs. The popularity of generative graphical models is due in part to the fact that they can be used to perform important tasks such as structure and function classification (e.g., [2, 5]) and to design new protein sequences (e.g., [7]). Unfortunately, existing methods for learning graphical models from MSAs either make unnecessarily strong assumptions about the nature of the underlying distribution over protein sequences, or else use greedy algorithms that are often sub-optimal. The goal of this paper is to introduce a new algorithm that addresses these two issues simultaneously and to demonstrate the superior performance of the resulting models.

A graphical model encodes a probability distribution over protein sequences in terms of a graph and a set of functions. The nodes of the graph correspond to the columns of the MSA and the edges specify the *conditional independencies* between the columns. Each node is associated with a local function that encodes the column-specific conservation statistics. Similarly, each edge is associated with a function that encodes the correlated mutation statistics between pairs of residues.

The task of learning a graphical model from an MSA can be divided into two sub-problems: (i) learning the topology of the graph (i.e., the set of edges), and (ii) estimating the parameters of the functions. The first problem is especially challenging because the number of unique topologies on a graph consisting of n nodes is $O(2^{n^2})$. For that reason, it is common to simply *impose* a topology on the graph, and then focus on parameter estimation. An HMM, for example, has a simple topology where each column is connected to its immediate neighbors. That is, the model assumes each column is conditionally independent of the rest of the MSA, given its sequential neighbors. This assumption dramatically reduces the complexity of learning the model but is not well justified biologically. In particular, it has been shown by Ranganathan and colleagues that it is necessary to model correlated mutations between non-adjacent residues [8, 9, 10].

Thomas and colleagues [11] demonstrated that correlated mutations between non-adjacent residues can be efficiently modeled using a different kind of graphical model known as a Markov Random Field (MRF). However,

[1] In this paper, the expression *protein family* is synonymous with *domain family*.

when using MRFs one must first identify the conditional independencies within the MSA. That is, one must learn the topology of the model. Thomas and colleagues address that problem using a greedy algorithm, called GMRC, that adds edges between nodes with high mutual information [11, 12, 13, 14]. Unfortunately, their algorithm provides no guarantees as to the optimality of the resulting model.

The algorithm presented in this paper, called GREMLIN (Generative REgularized ModeLs of proteINs), solves the same problem as [11] but does so using a method with strong theoretical guarantees. In particular, our algorithm is *consistent*, i.e. it is guaranteed to yield the true model as the data increases, and it has low *sample-complexity*, i.e. it requires less data to identify the true model than any other known approach. GREMLIN also employs *regularization* to penalize complex models and thus reduce the tendency to over-fit the data. Finally, our algorithm is also computationally efficient and easily parallelizable. We demonstrate GREMLIN by performing a detailed analysis on the well-studied WW and PDZ domains and demonstrate that it produces models with higher predictive accuracy than those produced using the GMRC algorithm. We then apply GREMLIN to 71 other families from the PFAM database and show that our algorithm produces models with consistently higher predictive accuracy than profile HMMs.

2 Materials and Methods

In what follows, we describe our approach to learning the statistical patterns within a given multiple sequence alignment. The resulting model is a probability distribution over amino acid sequences for a particular domain family.

2.1 Modeling Domain Families with Markov Random Fields

Let X_i be a finite discrete random variable representing the amino-acid composition at position i of the MSA of the domain family taking values in $\{1...k\}$ where the number of states, k, is 21 (20 amino acids with one additional state corresponding to a gap). Let $\mathbf{X} = \{X_1, X_2, ..X_p\}$ be the multi-variate random variable describing the amino acid composition of an MSA of length p. Our goal is to model $P(\mathbf{X})$, the amino-acid composition of the domain family.

Unfortunately, P($\mathbf{X}$) is a distribution over a space of size k^p, rendering the explicit modeling of the joint distribution computationally intractable for naturally occurring domains. However, by exploiting the properties of the distribution, one can significantly decrease the number of parameters required to represent this distribution.

To see the kinds of properties that we can exploit, let us consider a toy domain family represented by an MSA as shown in Fig. 1-(A). A close examination of the MSA reveals the following statistical properties of its composition: (i) the Tyrosine ('Y') at position 2 is conserved across the family; (ii) positions 1 and 4 are co-evolving – sequences with a (S) at position 1 have a Histidine (H) at position 4, while sequences with a Phenylalanine (F) at position 1 have a Tryptophan (W) at position 4; (iii) the remaining positions appear to evolve independently of each other. In probabilistic terms we say that X_1, X_3 are co-varying, and that the remaining X_i's are statistically independent. We can therefore encode the joint distribution over all positions in the MSA by storing one joint distribution $P(X_1, X_4)$, and the univariate distributions $P(X_i)$, for the remaining positions (since they are all statistically independent of every other variable).

The ability to factor the full joint distribution, $P(\mathbf{X})$, in this fashion has an important consequence in terms of space complexity. Namely, we can reduce the space requirements from 21^7 to $21^2 + 7 * 21$ parameters. This drastic reduction in space complexity translates to a corresponding reduction in time complexity for computations over the distribution. While this simple example utilizes independencies in the distribution; this kind of reduction is possible in the more general case of *conditional independencies*. A Probabilistic Graphical Model (PGM) exploits these (conditional) independence properties to store the joint probability distribution using a small number of parameters.

Intuitively, a PGM stores the joint distribution of a multivariate random variable in a graph; while any distribution can be modeled by a PGM with a complete graph, exploiting the conditional independencies in the distribution leads to a PGM with a (structurally) sparse graph. Following [12], we use a specific type of probabilistic graphical model called a Markov Random Field (MRF). In its commonly defined form with pair-wise log-linear potentials, a Markov Random Field (MRF) can be formally defined as a tuple $\mathcal{M} = (\mathbf{X}, \mathcal{E}, \Phi, \Psi)$ where $(\mathbf{X}, \mathcal{E})$ is an undirected graph over the random variables. $\mathbf{X}$ represents the set of vertices and $\mathcal{E}$ is the set of edges of the graph. The graph succinctly represents conditional independencies through its Markov properties, which state for instance that each node is independent of all other nodes given its neighbors. Thus, graph separation in $(\mathbf{X}, \mathcal{E})$ implies conditional independence. Φ, Ψ are a set of node and edge potentials, respectively, usually chosen to be log-linear functions of the form:

$$\phi_s = [e^{v_1^s} \;\; e^{v_2^s} \;\; ... \;\; e^{v_k^s}]; \qquad \psi_{st} = \begin{bmatrix} e^{w_{11}^{st}} & e^{w_{12}^{st}} & ... & e^{w_{1k}^{st}} \\ e^{w_{21}^{st}} & e^{w_{22}^{st}} & ... & e^{w_{2k}^{st}} \\ & ... & & \\ e^{w_{k1}^{st}} & e^{w_{k2}^{st}} & ... & e^{w_{kk}^{st}} \end{bmatrix} \tag{1}$$

where s is a position in the MSA, and (s,t) is an edge between the positions s and t in the MSA. ϕ_s is a $(k \times 1)$ vector and ψ_{st} is a $(k \times k)$ matrix. For future notational simplicity we further define

$$\mathbf{v}^s = [v_1^s \;\; v_2^s \;\; ... \;\; v_k^s] \qquad \mathbf{w}^{st} = \begin{bmatrix} w_{11}^{st} & w_{12}^{st} & ... & w_{1k}^{st} \\ w_{21}^{st} & w_{22}^{st} & ... & w_{2k}^{st} \\ & ... & & \\ w_{k1}^{st} & w_{k2}^{st} & ... & w_{kk}^{st} \end{bmatrix} \tag{2}$$

where $\mathbf{v}^s$ is a $(k \times 1)$ vector and $\mathbf{w}^{st}$ is a $(k \times k)$ matrix. $\mathbf{v} = \{\mathbf{v}^s | s = 1 \ldots p\}$ and $\mathbf{w} = \{\mathbf{w}^{st} | (s,t) \in \mathcal{E}\}$ are node and edge "weights". $\mathbf{v}$ is a collection of p, $(k \times 1)$ vectors and $\mathbf{w}$ is a collection of p, $(k \times k)$ matrices.

The probability of a particular sequence $x = \{x_1, x_2, \ldots, x_p\}$ according to $\mathcal{M}$ is defined as:

$$P_{\mathcal{M}}(X) = \frac{1}{Z} \prod_{s \in V} \phi_s(X_s) \prod_{(s,t) \in E} \psi_{st}(X_s, X_t) \tag{3}$$

where Z, the so-called partition function, is a normalizing constant defined as a sum over all possible assignments to $\mathbf{X}$.

$$Z = \sum_{X \in \mathbf{X}} \prod_{s \in V} \phi_s(X_s) \prod_{(s,t) \in E} \psi_{st}(X_s, X_t) \tag{4}$$

The structure of the MRF for the MSA shown in Fig. 1(A) is shown in Fig. 1(B). The edge between variables X_1 and X_4 reflects the statistical coupling between those positions in the MSA.

2.2 Structure learning with L_1 Regularization

In the previous section we outlined how an MRF can parsimoniously model the probability distribution $P(\mathbf{X})$. In this section we consider the problem of *learning* the MRF from an MSA.

Eq. 3 describes the probability of a sequence x for a specific model $\mathcal{M}$. Given a set of independent sequences

$\mathcal{X} = \{\mathbf{X^1}, \mathbf{X^2}, \mathbf{X^3}, ..., \mathbf{X^n}\}$, the log-likelihood of the model parameters $\Theta = (\mathcal{E}, \mathbf{v}, \mathbf{w})$ is then:

$$\text{ll}(\Theta) = \frac{1}{n} \sum_{X^i \in \mathcal{X}} \left[\sum_{s \in V} \log \phi_s(X_s^i) + \sum_{(s,t) \in E} \log \psi_{st}(X_s^i, X_t^i) \right] - \log Z \tag{5}$$

where the term in the braces is the unnormalized likelihood of each sequence, and Z is the global partition function. The problem of learning the structure *and* parameters of the MRF is now simply that of maximizing ll(Θ).

$$MLE(\theta) = \max_{\Theta} \text{ll}(\Theta) \tag{6}$$

This Maximum Likelihood Estimate (MLE) is guaranteed to recover the true parameters as the amount of data increases. However, this formulation suffers from two significant shortcomings: (i) the likelihood involves the computation of the global partition function which is computationally intractable and requires $O(k^p)$ time to compute, and (ii) in the absence of infinite data, the MLE can significantly over-fit the training data due to the potentially large number of parameters in the model.

An overview of our approach to surmount these shortcomings is as follows: first, we approximate the likelihood of the data with an objective function that is easier to compute, yet retains the optimality property of MLE mentioned above. To avoid over-fitting and learning densely connected structures, we then add a regularization term that penalizes complex models to the likelihood objective. The specific regularization we use is particularly attractive because it has high statistical efficiency.

The general regularized learning problem is then formulated as:

$$\max_{\Theta} \text{pll}(\Theta) - \text{R}(\Theta) \tag{7}$$

where the pseudo log-likelihood pll(Θ) is an approximation to the exact log-likelihood and R(Θ) is a regularization term that penalizes complex models.

While this method can be used to jointly estimate both the structure $\mathcal{E}$ and the parameters $\mathbf{v}, \mathbf{w}$, it will be convenient to divide the learning problem into two parts: (i) *structure learning* — which learns the edges of the graph, and (ii) and *parameter estimation* — learning $\mathbf{v}, \mathbf{w}$ given the structure of the graph. We will use a regularization penalty in the structure learning phase that focuses on identifying the correct set of edges. In the parameter estimation phase, we use these edges and learn $\mathbf{v}$ and $\mathbf{w}$ using a different regularization penalty that focuses on estimating $\mathbf{v}$ and $\mathbf{w}$ accurately. We note that once the set of edges has been fixed, the parameter estimation problem can be

solved efficiently. Thus, we will focus on the problem of learning the edges or, equivalently, the set of conditional independencies within the model.

2.2.1 Pseudo Likelihood

The log-likelihood as defined in Eq. 5 is smooth, differentiable, and concave. However, maximizing the log-likelihood requires computing the global partition function Z and its derivatives, which in general can take up to $\mathcal{O}(k^p)$ time. While approximations to the partition function based on Loopy Belief Propagation [15] have been proposed as an alternative, such approximations can lead to inconsistent estimates.

Instead of approximating the true-likelihood using approximate inference techniques, we use a different approximation based on a pseudo-likelihood proposed by [16], and used in [17, 18]. The pseudo-likelihood is defined as:

$$\begin{aligned} \text{pll}(\Theta) &= \frac{1}{n} \sum_{X^i \in \mathcal{X}} \sum_{j=1}^{p} \log(P(X_j^i | X_{-j}^i)) \\ &= \frac{1}{n} \sum_{X^i \in \mathcal{X}} \sum_{j=1}^{p} \left[\log \phi_j(X_j^i) + \sum_{k \in V_j'} \log \psi_{jk}(X_j^i, X_k^i) - \log Z_j \right] \end{aligned}$$

where X_j^i is the residue at the j^{th} position in the i^{th} sequence of our MSA, X_{-j}^i denotes the "Markov blanket" of X_j^i, and Z_j is a local normalization constant for each node in the MRF. The set V_j' is the set of all vertices which connect to vertex j in the PGM. The only difference between the likelihood and pseudo-likelihood is the replacement of a global partition function with local partition functions (which are sums over possible assignments to single nodes rather than a sum over all assignments to *all* nodes of the sequence). This difference makes the pseudo-likelihood significantly easier to compute in general graphical models.

The pseudo-likelihood retains the concavity of the original problem, and this approximation makes the problem tractable. Moreover, this approximation is known to yield a consistent estimate of the parameters under fairly general conditions if the generating distribution is in fact a pairwise MRF defined by a graph over $\mathbf{X}$ [19]. That is, under these conditions, as the number of samples increases, parameter estimates using pseudo-likelihood converge to the true parameters.

2.2.2 L1 Regularization

The study of convex approximations to the complexity and goodness of fit metrics has received considerable attention recently [17, 15, 20, 18]. Of these, those based on L_1 regularization are the most interesting because of their strong theoretical guarantees. In particular methods based on L_1 regularization exhibit consistency in both parameters and structure (i.e., as the number of samples increases we are guaranteed to find the true model), and high statistical efficiency (i.e., the number of samples needed to achieve this guarantee is small). See [21] for a recent review of L_1-regularization. Our algorithm uses L_1-regularization for both structure learning and parameter estimation.

For the specific case of block-L_1 regularization, $R(\Theta)$ usually takes the form:

$$R(\Theta) = \lambda_{node} \sum_{s=1}^{p} \|\mathbf{v}^s\|_2^2 + \lambda_{edge} \sum_{s=1}^{p} \sum_{t=s+1}^{p} \|\mathbf{w}^{st}\|_2 \tag{8}$$

where λ_{node} and λ_{edge} are regularization parameters that determine how strongly we penalize higher (absolute) weights. The value of λ_{node} and λ_{edge} control the trade-off between the log-likelihood term and the regularization term in our objective function.

The regularization described above groups all the parameters that describe an edge together in a *block*. The second term in Eq. 8 is the sum of the L_2 norms of each block. Since the L_2 norm is always positive, our regularization is exactly equivalent to penalizing the L_1 norm of the vector of norms of each block with the penalty increasing with higher values of λ_{edge}. It is important to distinguish the block-L_1 regularization on the edge weights from the more traditional L_2 regularization on the node weights where we sum the *squares* of the L_2 norms.

The L_1 norm is known to encourage sparsity (by setting parameters to be exactly zero), and the *block* L_1 norm we have described above encourages group sparsity (where *groups* of parameters are set to zero). Since, each group corresponds to all the parameters of a single edge, using the block L_1 norm leads to what we refer to as structural sparsity (i.e. sparsity in the edges). In contrast, the L_2 regularization also penalizes high absolute weights, but does not usually set any weights to zero, and thus does not encourage sparsity.

2.2.3 Optimizing Regularized Pseudo-Likelihood

In the previous two sections we described an objective function, and then a tractable and consistent approximation to it, given a set of weights (equivalently, potentials). However, to solve this problem we still need to be able to

find the set of weights that maximizes the likelihood under the block-regularization form of Eq. 7. We note that the objective function associated with block-L_1 regularization is no longer smooth. In particular, its derivative with respect to any parameter is discontinuous at the point where the group containing the parameter is 0. We therefore consider an equivalent formulation where the non-differentiable part of the objective is converted into a constraint making the new objective function differentiable.

$$\begin{aligned} & \max_{\Theta,\alpha} \text{pll}(\Theta) - \lambda_{node} \sum_{s=1}^{p} \|\mathbf{v}^s\|_2^2 - \lambda_{edge} \sum_{s=1}^{p} \sum_{t=s+1}^{p} \alpha_{st} \\ \text{subject to:} \quad & \forall (1 \leq s < t \leq p) : \alpha_{st} \geq \|\mathbf{w}^{st}\|_2 \end{aligned}$$

where the constraints hold with equality at the optimal (Θ,α). Intuitively, α_{st} behaves as a differentiable proxy for the non-differentiable $\|\mathbf{w}^{st}\|_2$, making it possible to solve the problem using techniques from smooth convex optimization. Since the constraints hold with equality at the optimal solution (ie $\alpha_{st} = \|\mathbf{w}^{st}\|_2$), the solutions and therefore, the formulations are identical.

We solve this reformulation through the use of projected gradients. We first ignore the constraints, compute the gradient of the objective, and take a step in this direction. If the step results in any of the constraints being violated we solve an alternative (and simpler) Euclidean projection problem:

$$\begin{aligned} & \min_{\Theta'\alpha'} \| \begin{bmatrix} \Theta' \\ \alpha' \end{bmatrix} - \begin{bmatrix} \Theta \\ \alpha \end{bmatrix} \|_2^2 \\ \text{subject to:} \quad & \forall (1 \leq s < t \leq p) : \alpha_{st} \geq \|\mathbf{w}^{st}\|_2 \end{aligned}$$

which finds the closest parameter vector to the vector obtained by taking the gradient step (in Euclidean distance), which satisfies the original constraints. In this case the projection problem can be solved extremely efficiently (in linear time) using an algorithm described in [18]. Methods based on projected gradients are guaranteed to converge to a stationary point [22], and convexity ensures that this stationary point is globally optimal.

In order to scale the method to significantly larger domains, we can sub-divide the structure learning problem into two steps. In the first step, each node is considered separately to identify its neighbors. This may lead to an asymmetric adjacency matrix, and so in the second step the adjacency matrix is made symmetric. This two-

step approach to structure learning has been extensively compared to the single step approach by [20] and has been found to have almost identical performance. The two-step approach however has several computational advantages. The problem of learning the neighbors of a node is exactly equivalent to solving a logistic regression problem with block-L_1 regularization, and this problem can be solved quickly and with low memory requirements. Additionally, the problem of estimating the graph can now be trivially parallelized across nodes of the graph since these logistic regression problems are completely decoupled. Parameter learning of the graph with just L_2 regularization can then be solved *extremely* efficiently using quasi-Newton methods [23].

3 Results

The probabilistic framework defined in Sec. 2.1 and the optimization objectives and algorithms defined in Sec. 2.2 constitute a method for learning a graphical model from a given MSA. The optimization framework has two major penalty parameters that can be varied (λ_v, λ_e). To understand the effects of these parameters, we first evaluated GREMLIN on artificial protein families whose sequence records were generated from known, randomly generated models. This lets us evaluate the success of the various components of GREMLIN in a controlled setting where the ground truth was known.

Our experiments involve comparing the performance of ranking edges and learning a graph structure using a variety of techniques, including: (i) our algorithm, GREMLIN; (ii) the greedy algorithm of [11, 12], denoted GMRC method'; and (iii) a simpler greedy algorithm that uses the metric suggested in [8], denoted $\Delta\Delta G^{stat}$. We also compare our performance with the Profile Hidden Markov Models [6] used by [4].

We note that the GMRC method only considers edges that meet certain coupling criteria (see [11, 12] for details). In particular, we found that it returns sparse graphs (fewer than 100 edges), regardless of choice of run-time parameters. GREMLIN, in contrast, returns a full spectrum from disconnected to completely connected graphs depending on the choice of the regularization parameter. In our experiments, we use our parameter estimation code on their graphs, and compare ourselves to the best graph they return.

In the remainder of this section, we demonstrate that GREMLIN significantly out-performs other algorithms. In particular, we show that GREMLIN achieves higher goodness of fit to the test set, and has lower prediction error than the GMRC method - *even when we learn models of similar sparsity*. Finally, we show that GREMLIN also significantly out performs profile HMM-based models for 71 real protein families, in terms of goodness of fit. These results demonstrate that the use of block-regularized structure learning algorithms can result in higher-

quality MRFs than those learnt by the GMRC method, and that MRFs produce higher quality models than HMMs.

3.1 Simulations

We generated 32-node graphs. Each node had a cardinality of 21 states, and each edge was included with probability ρ. Ten different values of ρ varying from 0.01 and 0.45 were used; for each value of ρ, twenty different graphs were generated resulting in a total of 200 graphs. For each edge that was included in a graph, edge and node weights were drawn from a Normal distribution (weights $\sim \mathcal{N}(0,1)$). Since each edge involves sampling 441 weights from this distribution, the edges tend to have many small weights and a few large ones. This reflects the observation that in positions with known correlated mutations, a few favorable pairs of amino acids are usually much more frequent than most other pairs. When we sample from our simulated graphs using these parameters, we therefore tend to generate such sequences.

For each of these 200 graphical models, we then sampled 1000 sequences using a Gibbs sampler with a burn-in of 10,000 samples and discarding 1,000 samples between each accepted sequence. These 1000 sequences were then partitioned into two sets: a training set containing 500 sequences and a held-out set of 500 sequences used to test the model. The training set was then used to train a model using the block regularization norm.

We first test our accuracy on structure learning. We measure accuracy by the F-score which is defined as

$$\text{F-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Precision and recall are in turn defined in terms of the number of true positives (tp), false positives (fp) and false negatives (fn) as precision $= \frac{tp}{tp+fp}$ and recall $= \frac{tp}{tp+fn}$.

Since the structure of the model directly depends only on the regularization weight on the edges, the structures were learnt for each norm and each training set with different values of λ_e (between 1 and 500), keeping λ_v fixed at 1.

Figure 2-A compares our structure learning method with the algorithm in [12]. We evaluate their method over a wide range of parameter settings and select the best model. Figure 2-A shows that our method significantly out-performs their method for *all* values of ρ. We see that over all settings our best model has an average F-score of *at least* 0.63. We conclude that we are able to infer accurate structures given the proper choice of settings.

Figure 2-B, shows the error in our parameter estimates given the true graph as a function of ρ. We also find that parameter estimation is reasonably robust to the choice of the regularization weights, as long as the regularization

weights are non-zero.

Fig. 3-A shows a qualitative analysis of edges missed by each method (we consider all simulated graphs and the best learnt graph of each method). We divide the missed edges into three groups (weak, intermediate and strong) based on their true L_2 norm. We see again that the three norms perform comparably, significantly out-performing the GMRC method in all three groups.

Finally, Fig. 3-B shows the sensitivity of our structure learning algorithms to the size of training set. In particular, we see that for the simulated graphs around 400 sequences results in us learning very accurate structures. However, as few as 50 sequences are enough to infer reasonable structures.

3.2 Evaluating Structure and Parameters Jointly

In a simulated setting, structure and parameter estimates can be compared against known ground truth. However, for real domain families we need other evaluation methods. We evaluate the structure and parameters for real domain families by measuring the imputation error of the learnt models. Informally, the imputation error measures the probability of *not* being able to "generate" a complete sequence, given an incomplete one. The imputation error of a column is measured by erasing it in the test MSA, and then computing the probability that the true (known) residues would be predicted by the learnt model. This probability is calculated by performing inference on the erased columns, conditioned on the rest of the MSA. The imputation error of a model is the average of its imputation error over columns.

Using imputation error directly for model selection generally gives us models that are too dense. Intuitively, once we have identified the true model, adding extra edges decreases the imputation error by a very small amount, probably a reflection of the finite-sample bias. We evaluated the modified AIC (Akaike Information Criteria) and BIC (Bayesian Information Criteria) for model selection due to their theoretically appealing properties. In the finite sample case we find that BIC performs well when the true graph is sparse, while AIC performs well when the true graph is dense. We discuss the information criteria in detail in the supplemental material, and provide some general suggestions for their use. Unfortunately, neither method performs well over the entire range of graphs. For this reason, we considered an approach to model selection based on finite sample error control. We chose to control the false discovery rate (FDR) in the following way. Consider permuting the each column of the MSA independently (and randomly). Intuitively, the true graph is now a graph with no edges. Thus, one approach to selecting the regularization parameter is to find the value that yields no edges on the permuted MSA. A more robust method, which we use, is to use the average regularization parameter obtained from multiple random permutations as in

[24]. In the results that follow we use 20 random permutations.

Given the success of GREMLIN on simulated data, and equipped with a method for model selection described above, we proceed to apply GREMLIN to real protein MSAs. We consider the WW and PDZ families in some detail since the extensive literature on these families allows us to draw meaningful conclusions about the learnt models.

3.3 A generative model for the WW domain

The WW domain family (Pfam id: PF00397 [4]) is a small protein interaction module with two highly conserved tryptophans that adopts a curved three-stranded β-sheet structure with a binding site for proline-containing peptides. In [9] and [10], the authors determine, using Statistical Coupling Analysis (SCA), that the residues can be divided into two clusters: the first cluster contains a set of 8 strongly coupled residues and the second cluster contains everything else. Based on this finding, the authors then designed 44 sequences that satisfy co-evolution constraints of the first cluster, of which 12 actually fold *in vitro*. An alternative set of control sequences, which did not satisfy the constraints, failed to fold.

We first constructed an MSA by starting with the PFAM alignment and removing sequences to construct a non-redundant alignment (no pair of sequences was greater than 80% similar). This resulted in an MSA with 700 sequences of which two thirds were used as a training set and the rest were used as a test set. Each sequence in the alignment had 30 positions. The training set was used to learn the model, for multiple values of λ_e. Given the structure of the graph, parameters were learned using $\lambda_v = 1, \lambda_e = 1$. The learnt model is presented in Figure 4.

Figure 5 compares the imputation errors of our approach (in red and yellow) with the GMRC method of [12] and Profile HMMs [6]. The model in red was learnt using λ_e selected by performing a permutation study. Since this model had more edges than the model learnt by GMRC, we used a higher λ_e to learn a model that had fewer edges than the GMRC model. The x-intercept was based on a loose lower bound on the error and was estimated by computing the imputation error on the test-data of a completely connected model *learnt on the test data*. Due to over-fitting, this is likely to be a very loose estimate of the lower bound. We find that our imputation errors are lower than the methods we compare to (even at comparable levels of sparsity).

To see which residues are affected by these edges, we construct a "coupling profile" (Fig. 4-C). We construct a shuffled MSA by taking the natural MSA and randomly permuting the amino acids within the same position (column of MSA) for each position. The new MSA now contains no co-evolving residues but has the same conservation profile as the original MSA. To build a coupling profile, we calculate the difference in the imputation error of sequences in a held-out test set and the shuffled MSA. Intuitively, having a high imputation error difference

means that the position was indeed co-evolving with some other positions in the MSA. The other positions would also have a high imputation error difference in the coupling profile.

We also performed a retrospective analysis of the artificial sequences designed by [10]. We attempt to distinguish sequences that folded from those that didn't. Although this is a discriminative test (folded or not) of a generative model, we nevertheless achieve a high AUC of 0.87 (the ROC curve is shown and described in the supplemental material). We therefore postulate that the additional constraints we identify are indeed critical to the stability of the WW fold. In comparing our AUC to the published results of [12] (AUC of 0.82) and the Profile HMM (AUC of 0.83) we see that we are able to better distinguish artificial sequences that fold from those that don't.

3.4 Allosteric regulation in the PDZ domain

The PDZ domain is a family of small, evolutionarily well represented protein binding motifs. The domain is most commonly found in signaling proteins and helps to anchor trans-membrane proteins to the cytoskeleton and hold together signaling complexes. The PDZ domain is also interesting because it is considered an *allosteric* protein. The domain, and its members have been studied extensively, in multiple studies, using a wide range of techniques ranging from computational approaches based on statistical coupling ([8]) and Molecular Dynamics simulations [25], to NMR based experimental studies ([26]).

We use the MSA from [8]. The MSA is an alignment of 240 non-redundant sequences, with 92 positions. We chose a random sub-sample with two-thirds of the sequences as the training set and use the rest as a test set. Using this training set, we learnt generative models for each of the block regularizers, and choosing the smallest value of λ_e that gave zero edges for 20 permuted MSAs as explained previously. The resulting model had 112 edges (Fig. 6). Figure 5 summarizes the imputation errors on the PDZ domain. We again observe that the model we learn is denser than that learnt by GMRC and has lower imputation error. However, even at comparable sparsity GREMLIN out-performs the Profile HMM and GMRC.

The SCA based approach of [8] identified a set of residues that were coupled to a residue near the active site (HIS-70) including a residue at a distal site on the other end of the protein (GLY-49 in this case). Since the SCA approach can only determine the presence of a dependence but cannot distinguish between direct and indirect couplings, only a cluster of residues was identified. Our model also identifies this interaction, but more importantly, it determines that this interaction is mediated by ALA-74 with position 74 *directly* interacting with both these positions. By providing such a list of sparse interactions our model can provide a small list of hypotheses

to an experimentalist looking for possible mechanisms of such allosteric behavior.

In addition to the pathway between HIS-70 and GLY-49, we also identify residues not on the pathway that are connected to other parts of the protein including, for example ASN-61 of the protein. This position is connected to ALA-88 and VAL-60 in our model, and does not appear in the network suggested by [8], but has been implicated by NMR experiments [26] as being dynamically linked to the active site.

From our studies on the PDZ and WW families we find that GREMLIN produces higher quality models than GMRC and profile HMMs, and identifies richer sets of interactions. In the following section we consider the application of GREMLIN to a larger subset of the PFAM database. Since the greedy algorithm of GMRC does not scale to large families, our experiments are restricted to comparing the performance of GREMLIN with that of profile HMMs.

3.5 Large-scale analysis of families from Pfam

We selected all protein families from PFAM [4] that had at least 300 sequences in their seed alignment. We restricted ourselves to such families because the seed alignments are manually curated before depositing and are therefore expected to have higher quality than the whole alignments. We pre-processed these alignments to remove redundant sequences (sequence similarity $> 80\%$) in order to generate non-redundant alignments. From each alignment, we then removed columns that had gaps in more than half the sequences, and then removed sequences in the alignment that had more than insertions at more than 10% of these columns. Finally, we removed sequences that had more than 20% gaps in their alignment. If this post-processing resulted in an alignment with less than 300 sequences, it was dropped from our analysis. 71 families remained at the end of this process. These families varied greatly in their length with the shortest family having 15 positions and the longest having more than 450 positions and the median length being 78 positions. Figure 7 shows the distribution of lengths.

For each of these families, we created a random partition of the alignment into training (with 2/3 of the sequences) and test (with 1/3 of the sequences) alignments and trained an MRF using our algorithm. As mentioned earlier, we chose λ_e by performing 20 random permutations of each column and choosing the smallest λ_e that gave zero edges on all 20 permutations. As a baseline comparison, we also trained a profile-HMM using the Bioinformatics toolkit in Matlab on the training alignments. We then used the learnt models to impute the composition of each position of the test MSA and computed the overall and per-position imputation errors for both models. Due to space constraints, we provide the models and detailed analyses for each family on a supporting website (details in appendix) and focus on overall trends in the rest of this section.

Figure 8 shows the histograms of the distance between residues connected by an edge and the degree of the nodes. Approximately 30% of the edges are between residues that are more than 10 Å of each other. That is, GREMLIN learns edges that are different than those that would be obtained from a contact map. Despite the presence of long-range edges, GREMLIN does learn a sparse graph; most nodes have degree less than 5, and the majority have 1 or fewer edges.

Fig. 9-(A) shows a boxplot demonstrating the effect of incorporating co-evolution information according to our model. The y-axis shows the decrease in the per-position imputation error when moving from a profile-HMM model to the corresponding MRF, while the x-axis bins this improvement according to the number of edges in the MRF at that position. In each box, the central red line is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually with red '+' marks. As the figure shows, moving from a profile-HMM model to an MRF never hurts: for positions with 0 edges, there is no difference in imputation; for positions with at least one edge, the MRF model *always* results in lower error. While this is not completely surprising given that the MRF has more parameters and is therefore more expressive, it is not obvious that these parameters can be learnt from such little data. Our results demonstrate that this is indeed possible. While there are individual variations within each box, the median improvement in imputation error shows a clear linear relationship to the number of neighbors of the position in the model. This linear effect falls off towards the right in the high-degree vertices where the relationship is sub-linear. Fig. 9-(B) shows the effect of this behavior on the improvement in overall imputation error across all positions for a family.

3.6 Computational efficiency

In this subsection we briefly discuss the computational efficiency of GREMLIN . The efficiency of GREMLIN was measured based on the running time (i.e. CPU seconds until a solution to the convex optimization problem is found). GREMLIN was run on a 64 node cluster. Each node had 16GB DRAM and 2xquad-cores (each with 2.8-3 GHZ), allowing us to run 512 jobs in parallel with an average of 2GB RAM per job.

Fig. 10 shows a plot of the running time for a given λ_e on all the PFAM MSAs. Fig. 10-(A) plots the running time for learning the neighbors of a position, against the number of columns (positions) in the MSA (A) while 10-(B) plots it against number of rows (sequences) in the training MSA. In both, the average running time *per column* is shown in red circles. While learning the neighbors at a position, since GREMLIN is run in parallel for each column of the MSA, the actual time to completion for each protein depends on the maximum running time across

these columns. This number is shown in blue squares. Fig. 10-(C) plots the running time for parameter learning against the maximum running time to learn the neighbors at a position. Recall that this task is performed serially. As the figure demonstrates, GREMLIN takes roughly similar amounts of time in its parallel stage (neighborhood learning) as it does in its serial stage (parameter learning).

The plots show that the running time has an increasing trend as the size of the MSA increases (number of positions and number of sequences). Also, the dependence of the running time on the number of columns is stronger than its dependence on the number of rows. This is consistent with the analysis in [17] which shows that a similar algorithm for structure learning with a pure L_1 penalty has a computational complexity that scales as $\mathcal{O}(\max(n,p)p^3)$, where n corresponds to the number of rows and p to the number of columns in the MSA.

4 DISCUSSION

4.1 Related Work

The study of co-evolving residues in proteins has been a problem of much interest due to its wide utility. Much of the early work focused on detecting such pairs in order to predict contacts in a protein in the absence of a solved structure [27, 28] and to perform fold recognition. The pioneering work of [8] used an approach to determine probabilistic dependencies they call SCA and observed that analyzing such patterns could provide insights into the allosteric behavior of the proteins and be used to design new sequences [9]. Others have since developed similar methods [29, 30, 31]. By focusing on co-variation or probabilistic *dependencies* between residues, such methods conflate direct and indirect influences and can lead to incorrect estimates. In contrast, [12] developed an algorithm for learning a Markov Random Field over sequences. Their constraint-based algorithm proceeds by identifying conditional independencies and adding edges in a greedy fashion. However, the algorithm can provide no guarantees on the correctness of the networks it learns. They then extended this approach to incorporate interaction data to learn models over pairs of interacting proteins [13] and also develop a sampling algorithm for protein design using such models [14]. More recently, [32] use a similar approach to determine residue contacts at a protein-protein interface. Their method uses a gradient descent approach using Loopy Belief Propagation to approximate likelihoods. Additionally, their algorithm does not regularize the model and may therefore be prone to over-fitting. In contrast, we use a Pseudo-Likelihood as our objective function thereby avoiding problems of convergence that Loopy BP based methods can face and regularize the model using block regularization to prevent over-fitting.

Block regularization is most similar in spirit to the group Lasso [33] and the multi-task Lasso [34]. Lasso [35] is the problem of finding a linear predictor, by minimizing the squared loss of the predictor with an L_1 penalty. It is well known that the shrinkage properties of the L_1 penalty lead to sparse predictors. The group Lasso extends this idea by grouping the weights of some features of the predictor using an L_2 norm, [33] show that this leads to sparse selection of groups. The multi-task Lasso solves the problem of multiple separate (but similar) regression problems by grouping the weight of a single feature across the multiple tasks. Intuitively, we solve a problem similar to a group Lasso, replacing the squared loss with an approximation to the negative log-likelihood, where we group all the feature weights of an edge in an undirected graphical model. Thus, sparse selection of groups gives our graphs the property of structural sparsity.

Lee and co-workers [36] introduced structure learning in MRFs with a pure L_1 penalty, but do not go further to explore block regularization. They also use a different approximation to the likelihood term, using Loopy Belief Propagation. Schmidt and co-workers [18] apply block-regularized structure learning to the problem of detecting abnormalities in heart motion. They also developed an efficient algorithm for tractably solving the convex structure learning problem based on projected gradients.

4.2 Mutual Information performs poorly in the structure learning task

One of the key advantages of a graphical model based approach to modeling protein families is that the graph reveals which interactions are direct and which are indirect. One might assume that alternative quantities, like Mutual Information, might yield similar results. We now demonstrate with an example that a simple Mutual Information based metric cannot distinguish well between direct and indirect interactions. Fig. 11-(A) shows the adjacency matrix of a Probabilistic Graphical Model. The elements of the matrix are color-coded by the strength of their interaction: blue represents the weakest interaction (of strength 0, i.e. a non-interaction) and red the strongest interaction in this distribution. Fig. 11-(B) shows the mutual information induced between the variables by this distribution as measured from 500 sequences sampled from the graphical model (the diagonal elements of the mutual information matrix have been omitted to highlight the information between different positions). While it may appear visually that (B) shares a lot of structure with (A), it isn't actually the case. In particular, the edges with the highest mutual information indeed tend to be direct interactions; however a large fraction of the direct interactions might not have high MI. This is demonstrated in Fig. 11-(C) where MI is used as a metric to classify edges into direct and indirect interactions. The blue line shows the ROC curve using MI as a metric and has only moderate discriminatory power for this task (AUC: 0.71). In contrast, our approach, shown in red, is much more

successful at discriminating between direct and indirect interactions: the AUC of our approach is a near-perfect 0.98.

4.3 Influence of Phylogeny

One limitation associated with a sequence-only approach to learning a statistical model for a domain family is that the correlations observed in the MSA can be inflated due to phylogeny [37, 38]. A pair of co-incident mutations at the root of the tree can appear as a significant dependency even though they correspond to just once co-incident mutation event. To test if this was the case with the WW domain, we constructed a phylogenetic tree from the MSA using Junes-Cantor measure of sequence dissimilarity. In the case of WW, this resulted in a tree with two clear sub-trees, corresponding to two distinct (nearly equal-sized) clusters in sequence space. Since each sub-tree had a number of sequences, we re-learnt MRFs for each sub-tree separately. The resulting models for each sub-tree did not vary significantly from our original models – a case that would have occurred if there were co-incident mutations at the root that lead to spurious dependencies. Indeed the only difference between the models was in the C-terminal end was an edge between positions 1 and 2 that was present in sequences from the first sub-tree but was absent in the second sub-tree. This occurred because in the second sub-tree, these positions were completely conserved due to which our model was not able to determine the dependency between them. While this does not eliminate the possibility of confounding due to phylogeny, we have reason to believe that our dependencies are robust to significant phylogenetic confounding in this family. A similar analysis for the PDZ domain, found 3 sub-trees, and again we found that the strongest dependencies were consistent across models learnt on each sub-tree separately. Nevertheless, we believe that incorporating phylogenetic information into our method is an important direction for future research.

5 CONCLUSION

In this paper we have proposed a new algorithm for discovering and modeling the statistical patterns contained in a given MSA. Overall, we find that by employing sound probabilistic modeling and convex structure (and parameter) learning, we are able to find a good balance between structural sparsity (simplicity) and goodness of fit. One of the key advantages of a graphical model approach is that the graph reveals the direct and indirect constraints that can further our understanding of protein function and regulation.

MRFs are generative models, and can therefore be used design new protein sequences via sampling and infer-

ence. However, we expect that the utility of our model in the context of protein design could be greatly enhanced by incorporating structure based information which explicitly models the physical constraints of the protein. We have previously shown in [39], that it is possible to construct MRFs that integrate both sequence and structure information. We believe an interesting direction for future work is to apply structure learning to MSAs enhanced with physical constraints (e.g., interactions energies) in the form of informative priors or as edge features. The learning algorithm would then select the type of constraint (i.e., sequence vs structure) that best explains the covariation in the MSA.

Finally, we note that there are a number of other ways to incorporate phylogenetic information directly into our model. For example, given a phylogenetic clustering of sequences, we can incorporate a single additional node in the graphical model reflecting the cluster to which the sequence belongs. This would allow us to distinguish functional coupling from coupling caused due to phylogenetic variations.

Acknowledgment

Funding: This research was supported by NSF IIS-0905193 and an award from Microsoft Research to CJL.

Supplemental Material

Comparison of structures learnt at different regularization levels

Fig. 12 shows our performance in predicting the true structure by using L_1-L_2 (Fig. 12) The accuracy is measured using the F-score (the harmonic mean of precision and recall) of the edge set. We observe that for all settings of ρ GREMLIN learns fairly accurate graphs at some value of λ_e.

Model selection using information criteria

We consider modifications to two widely used model selection strategies. The Bayesian Information Criterion (BIC) [40], is used to select parsimonious models and is known to be asymptotically consistent in selecting the true model. The Akaike Information Criterion (AIC) [41], typically selects denser models than the BIC, but is known to be asymptotically consistent in selecting the model with lowest predictive error (risk). In general, they do not however select the same model [42].

We use the following definitions:

$$\begin{aligned} \text{pseudo-BIC}(\lambda) &= -2\text{pll}(\lambda) + \log(\text{n})\text{df}(\lambda) \\ \text{pseudo-AIC}(\lambda) &= -2\text{pll}(\lambda) + 2\text{df}(\lambda) \end{aligned}$$

Where we use the pseudo log-likelihood approximation to the log-likelihood. While it may be expected that using the pseudo log-likelihood instead of the true log-likelihood may in fact lead to inconsistent selection a somewhat surprising result [43] shows that in the case of BIC using pseudo log-likelihood is in fact also consistent for model selection. Although we aren't aware of the result, we expect a similar result to hold for the risk consistency of the pseudo-AIC.

We evaluate the likelihood on the *training* sample to score the different models. n is the number of training sequences.

Estimating the degrees of freedom of a general estimator is quite hard in practice. This has lead to use of various heuristics in practice. For the LASSO estimator which uses a pure-L1 penalty, it is known that the number of non-zeros in the regression vector is a good estimate of the degrees of freedom. A natural extension when using a *block*-L1 penalty is the number of non-zero blocks (i.e. edges). Since this does not differentiate between weak

and strong edges, we used the block-L1 norm as an estimate of the degress of freedom. In our simulations, we find that choice often results in good model selection.

Figure 13 shows the performance of the two model selection strategies at different sparsity levels. We evaluate the performance by learning several graphs (at different levels of regularization) and comparing the Spearman rank-correlation between the F-score of the graphs and their rank. We can clearly see that when the true graph is sparse the modified BIC has a high rank-correlation, whereas when the true graph is dense the modified AIC does well, with neither method providing reliable model selection for all graphs.

Receiver operating characteristic curve

We consider the task of distinguishing artificial sequences that were found to take the WW fold from those that did not. All sequences and their labels (folded in vivo or not) are from [10]. The ROC curve (Figure 14) is obtained by varying a threshold on scores (we use the unnormalized likelihood as the score). Sequences above the threshold are predicted to fold. For each threshold we calculate the sensitivity and specificity and show the resulting curve.

Supporting Website

Details of the models for the 71 protein families along with analyses of coupling profiles are provided on our supporting website: `http://www.cs.cmu.edu/~cjl/gremlin/`.

References

[1] Anders Krogh, Michael Brown, I. Saira Mian, Kimmen Sjlander, and David Haussler. Hidden markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[2] Kevin Karplus, Kimmen Sjlander, Christian Barrett, Melissa Cline, David Haussler, Richard Hughey, Liisa Holm, Chris Sander, Ebi England, and Ebi England. Predicting protein structure using hidden markov models. In *Proteins: Structure, Function, and Genetics*, pages 134–139, 1997.

[3] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1998.

[4] A. Bateman, E. Birney, L. Cerruti, R. Durbin, L. Etwiller, S.R. Eddy, S. Griffiths-Jones, K.L. Howe, M. Marshall, and E.L.L. Sonnhammer. The Pfam protein families database. *Nucleic acids research*, 30(1):276, 2002.

[5] Yan Liu, Jaime G. Carbonell, Peter Weigele, and Vanathi Gopalakrishnan. Protein fold recognition using segmentation conditional random fields. *Journal of Computational Biology*, 13(2):394–406, 2006.

[6] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14:755–763, 1998.

[7] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Protein design by sampling an undirected graphical model of residue constraints. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 6(3):506–516, 2009.

[8] S. W. Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286:295–299, Oct 1999.

[9] M. Socolich, S. W. Lockless, W. P. Russ, H. Lee, K. H. Gardner, and R. Ranganathan. Evolutionary information for specifying a protein fold. *Nature*, 437:512–518, Sep 2005.

[10] W. P. Russ, D. M. Lowery, P. Mishra, M. B. Yaffe, and R. Ranganathan. Natural-like function in artificial WW domains. *Nature*, 437:579–583, Sep 2005.

[11] John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg. Graphical models of residue coupling in protein families. In *BIOKDD '05: Proceedings of the 5th international workshop on Bioinformatics*, pages 12–20, New York, NY, USA, 2005. ACM.

[12] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Graphical models of residue coupling in protein families. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 5(2):183–197, 2008.

[13] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Graphical models of protein-protein interaction specificity from correlated mutations and interaction data. *Proteins: Structure, Function, and Bioinformatics*, 76(4):911–29, 2009.

[14] J. Thomas, N. Ramakrishnan, and C. Bailey-Kellogg. Protein Design by Sampling an Undirected Graphical Model of Residue Constraints. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 6(3):506–516, 2009.

[15] 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.

[16] J. Besag. Efficiency of pseudolikelihood estimation for simple Gaussian fields. *Biometrika*, 64(3):616–618, 1977.

[17] 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.

[18] 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.

[19] B. Gidas. Consistency of maximum likelihood and pseudo-likelihood estimators for Gibbs Distributions. *Institute for Mathematics and Its Applications*, 10:129–+, 1988.

[20] 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.

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

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

[23] Dong C. Liu, Jorge Nocedal, Dong C. Liu, and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[24] Jennifer Listgarten and David Heckerman. Determining the number of non-spurious arcs in a learned dag model: Investigation of a bayesian and a frequentist approach. *23rd annual conference on Uncertainity in Artificial Intelligence*, 2007.

[25] Anne Dhulesia, Joerg Gsponer, and Michele Vendruscolo. Mapping of two networks of residues that exhibit structural and dynamical changes upon binding in a pdz domain protein. *Journal of the American Chemical Society*, 130(28):8931–8939, July 2008.

[26] E.J. Fuentes, C.J. Der, and A.L. Lee. Ligand-dependent dynamics and intramolecular signaling in a PDZ domain. *Journal of molecular biology*, 335(4):1105–1115, 2004.

[27] D. Altschuh, T. Vernet, P. Berti, D. Moras, and K. Nagai. Coordinated amino acid changes in homologous protein families. *Protein Eng.*, 2(3):193–199, September 1988.

[28] Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Genetics*, 18(4):309–317, April 1994.

[29] S. N. Fatakia, S. Costanzi, and C. C. Chow. Computing highly correlated positions using mutual information and graph theory for g protein-coupled receptors. *PLoS ONE*, 4(3):e4681, 03 2009.

[30] Anthony A. Fodor and Richard W. Aldrich. On evolutionary conservation of thermodynamic coupling in proteins. *Journal of Biological Chemistry*, 279(18):19046–19050, April 2004.

[31] Angelika Fuchs, Antonio J. Martin-Galiano, Matan Kalman, Sarel Fleishman, Nir Ben-Tal, and Dmitrij Frishman. Co-evolving residues in membrane proteins. *Bioinformatics*, 23(24):3312–3319, December 2007.

[32] M. Weigt, R. A. White, H. Szurmant, J. A. Hoch, and T. Hwa. Identification of direct residue contacts in protein-protein interaction by message passing. *Proc. Natl. Acad. Sci. U.S.A.*, 106:67–72, Jan 2009.

[33] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67, 2006.

[34] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

[35] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[36] 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.

[37] D. D. Pollock and W. R. Taylor. Effectiveness of correlation analysis in identifying protein residues undergoing correlated evolution. *Protein Eng.*, 10(6):647–657, June 1997.

[38] Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associates, September 2003.

[39] H. Kamisetty, B. Ghosh, C. Bailey-Kellogg, and C.J. Langmead. Modeling and Inference of Sequence-Structure Specificity. In *Proc. of the 8th International Conference on Computational Systems Bioinformatics (CSB)*, pages 91–101, 2009.

[40] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[41] H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, January 2003.

[42] Yuhong Yang. Can the strengths of aic and bic be shared? *Biometrica*, 92:2003, 2003.

[43] Imre Csiszar and Zsolt Talata. Consistent estimation of the basic neighborhood of markov random fields. *The Annals of Statistics*, 34(1):123–145, 2006.

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

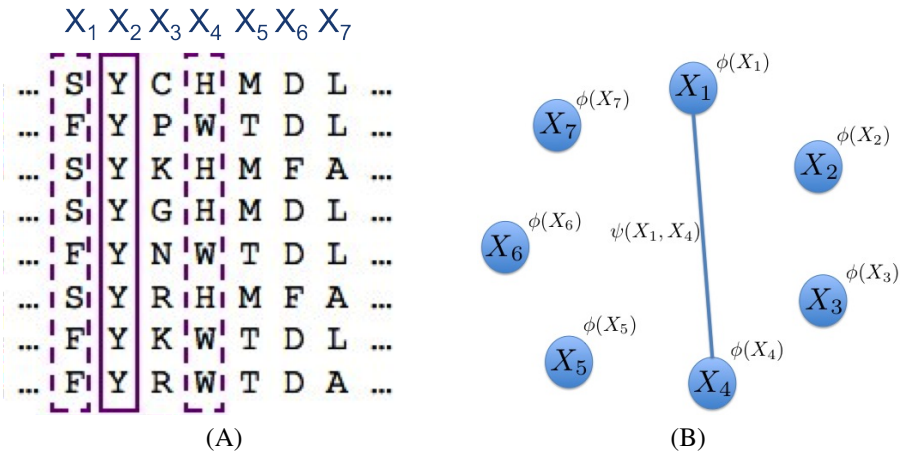
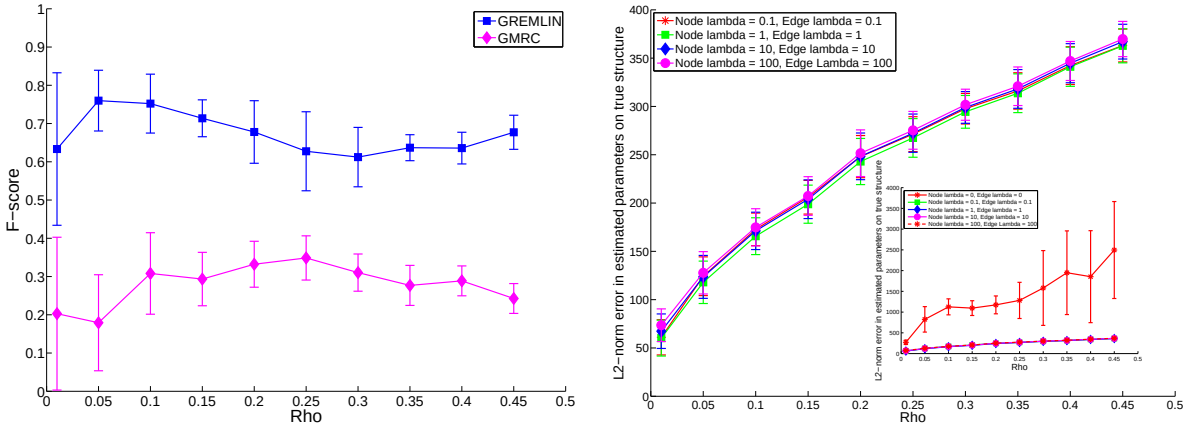
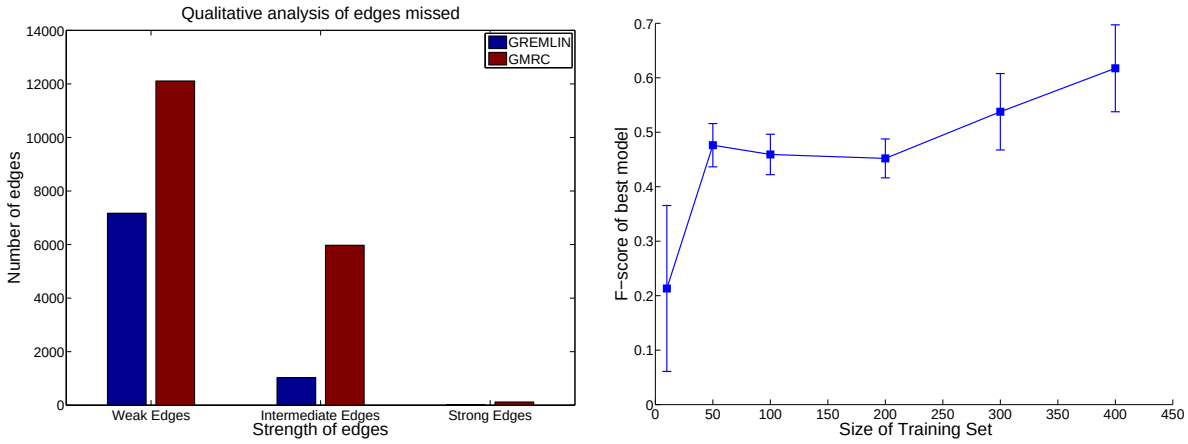


Figure 1: (A) A multiple sequence alignment (MSA) for a hypothetical domain family. (B) The Markov Random Field encoding the conservation in and the coupling in the MSA. The edge between random variables X_1 and X_4 reflects the coupling between positions 1 and 4 in the MSA.



(A) (B)

Figure 2: (A) Edge occurrence probability ρ versus F-score for the structure learning methods we propose, and the method proposed in [12]. (B) L_2 norm of the error in the estimated parameters as a function of the weight of the regularization in stage two. The inset shows the case when no regularization is used in stage two. The much higher parameter estimation error in this case highlights the need for regularization in *both* stages.



(A) (B)

Figure 3: (A) Qualitative grouping of edges missed by GREMLIN and the GMRC method (B) Sensitivity of structure learning to size of training set.

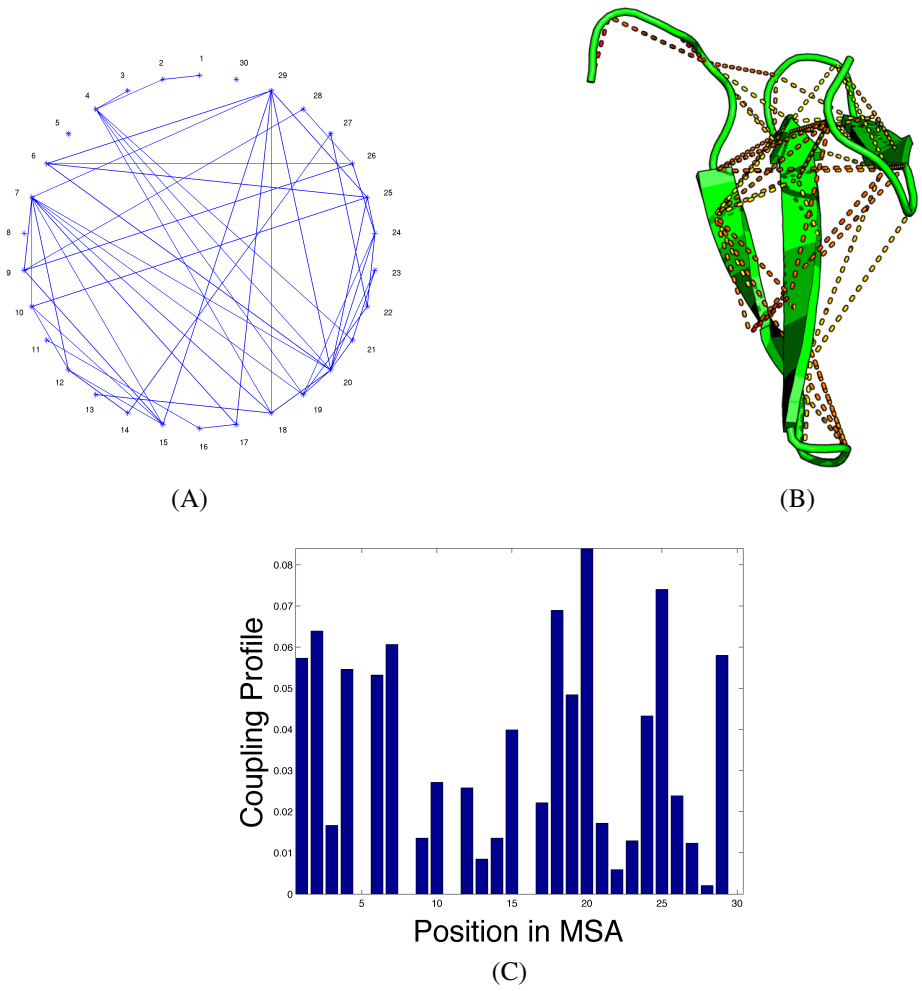


Figure 4: **WW domain model**. Edges returned by GREMLIN overlayed on a circle (**a**) and on the structure (**b**) of the WW domain of Transcription Elongation Factor 1 (PDB id: 2DK7) [44]. (**c**) Coupling profile (see text).

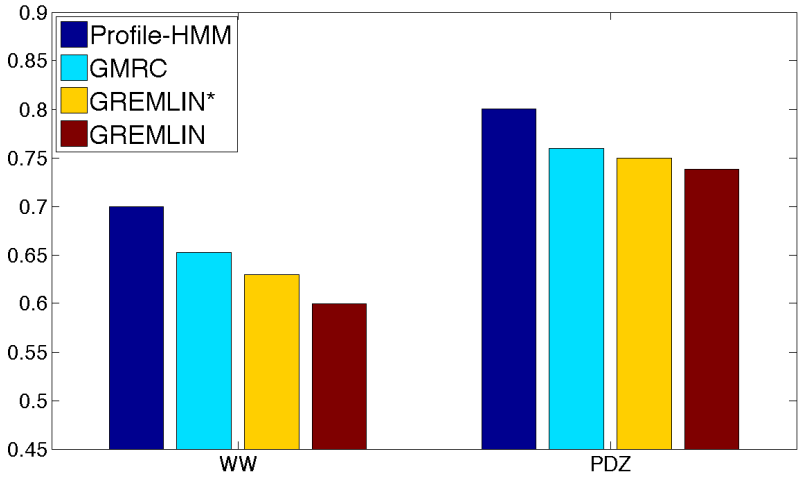
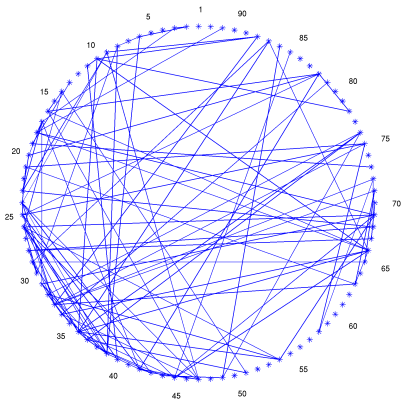
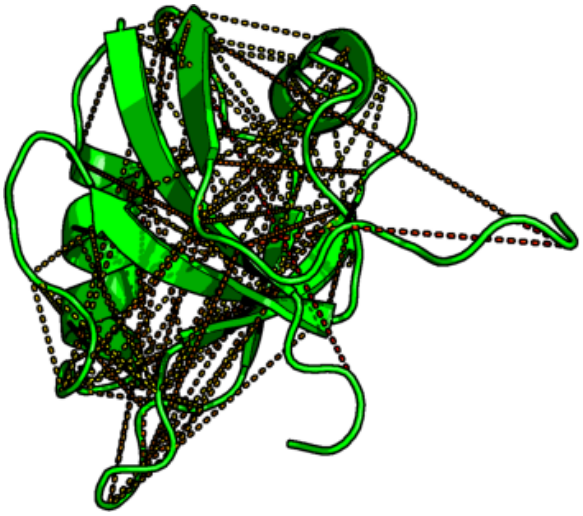


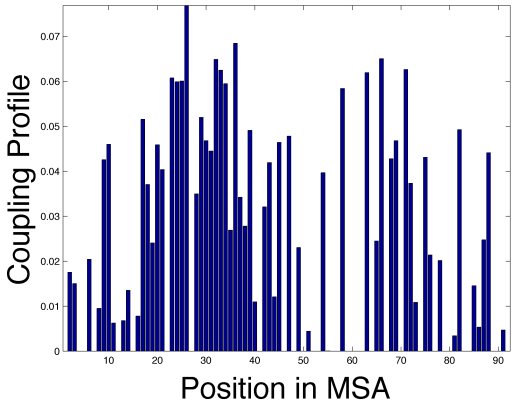
Figure 5: Comparison of Imputation errors on WW and PDZ families. We consider two variants of GREMLIN - with the regularization parameter selected either to produce a model with a smaller number of edges than GMRC (third bar in each group, shown in yellow) or to have zero edges on 20 permuted MSAs (last bar, shown in red). The x-intercept was chosed by estimating a lower bound on the imputation error as described in the text.



(A)



(B)



(C)

Figure 6: **PDZ domain model**. Edges returned by GREMLIN overlayed on a circle (**a**) and on the structure (**b**) of PDZ domain of PSD-95 (PDB id:1BE9). (**c**) Coupling profile (see text).

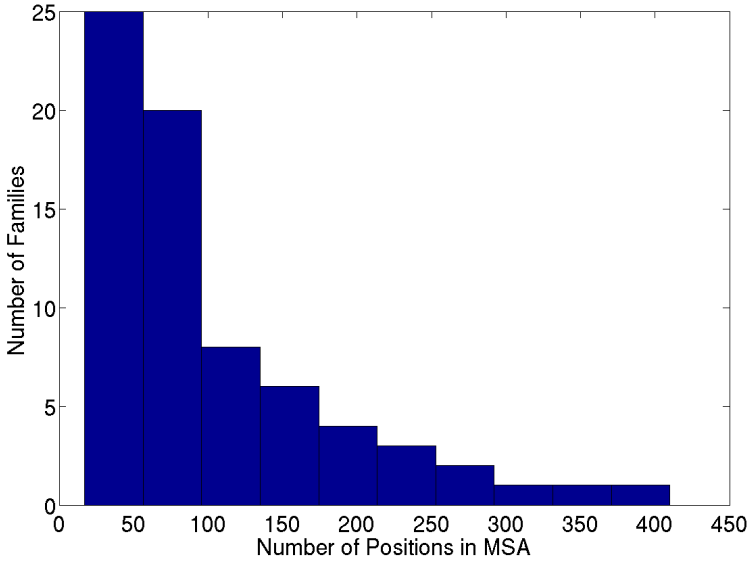
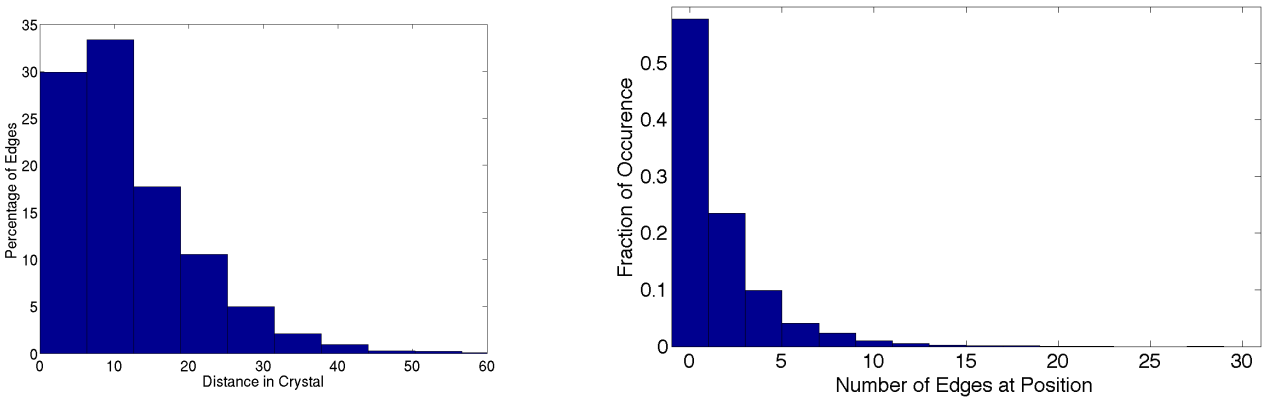
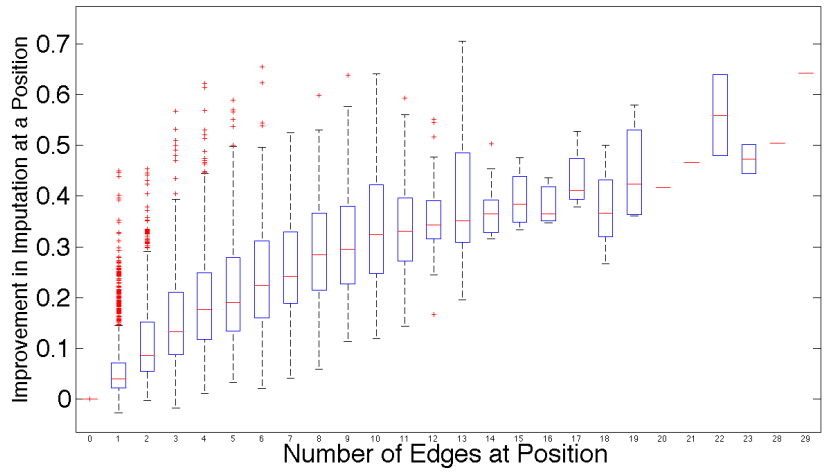


Figure 7: Histogram of MSA lengths of the 73 PFAM families in our study.

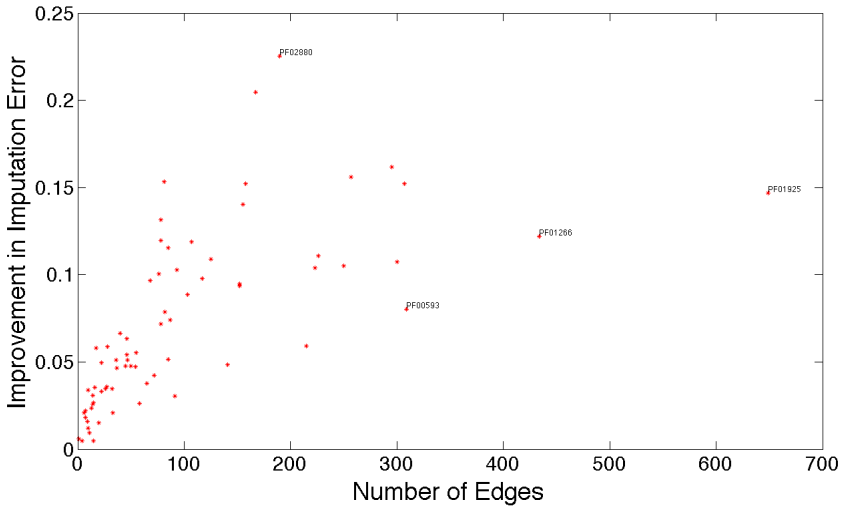


(A) (B)

Figure 8: (A) Histogram of the distance in crystal structure. (B) Degree distribution across all proteins.

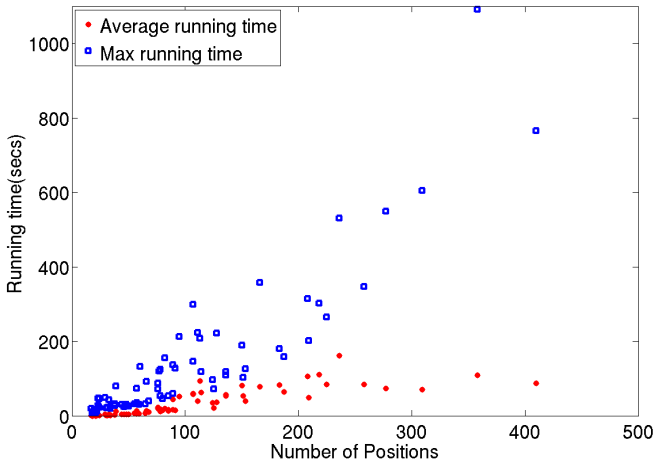


(A)

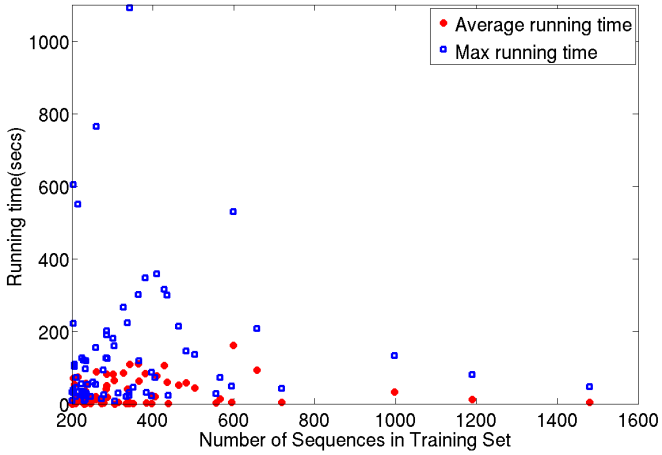


(B)

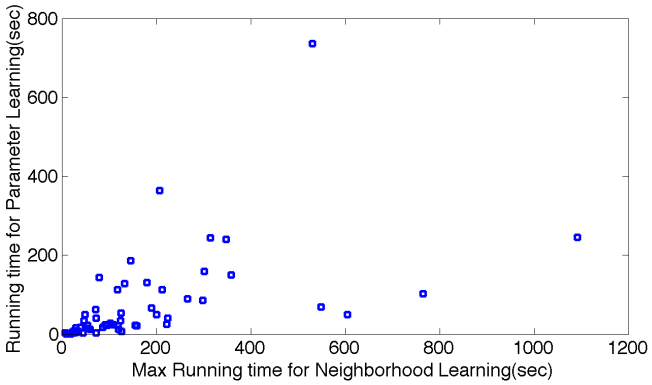
Figure 9: (A) Boxplot displaying the effect of coupling on improvement in imputation error at a position when compared to a profile-HMM. The median imputation error shows a near-linear decrease as the number of neighbors learnt by the model increases. (B) Improvement in overall imputation error across all positions for each family.



(A)

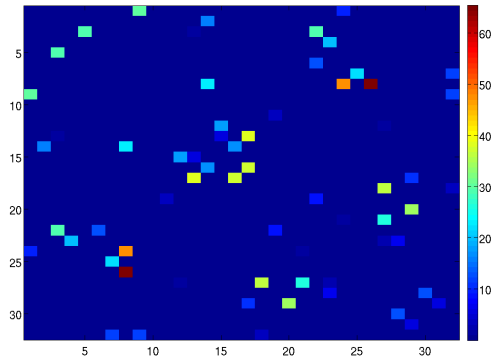


(B)

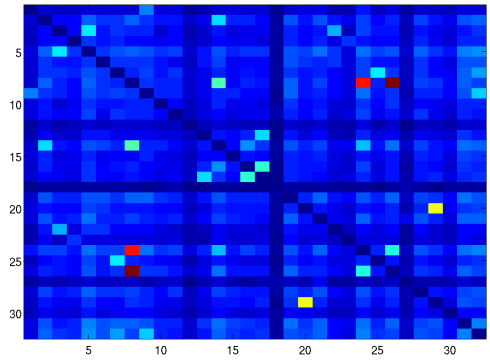


(C)

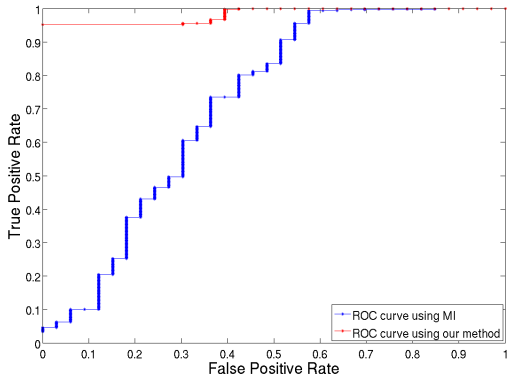
Figure 10: (A) Number of Positions in the MSA versus runtime of Neighborhood learning (in seconds) (B) Number of sequences in the MSA versus runtime of Neighborhood learning (C) Runtime of Neighborhood learning versus runtime of Parameter learning



(A)



(B)



(C)

Figure 11: (A) Adjacency matrix of a Boltzmann distribution colored by edge strength. (B) Mutual Information between positions induced by this Boltzman distribution. While the mutual information of the strongest edges is highest; a large fraction of the edges have MI comparable to many non-interactions. (C) Shows the weak ability of MI to distinguish between edges and indirect interactions in contrast to GREMLIN . AUC using MI: 0.71; AUC using GREMLIN : 0.98.

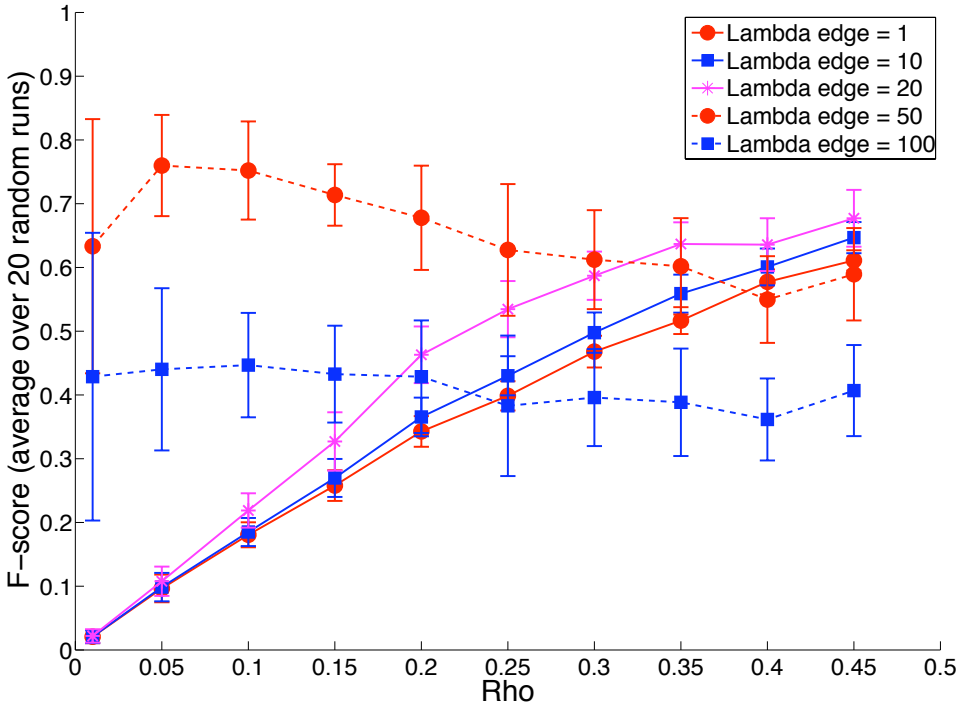
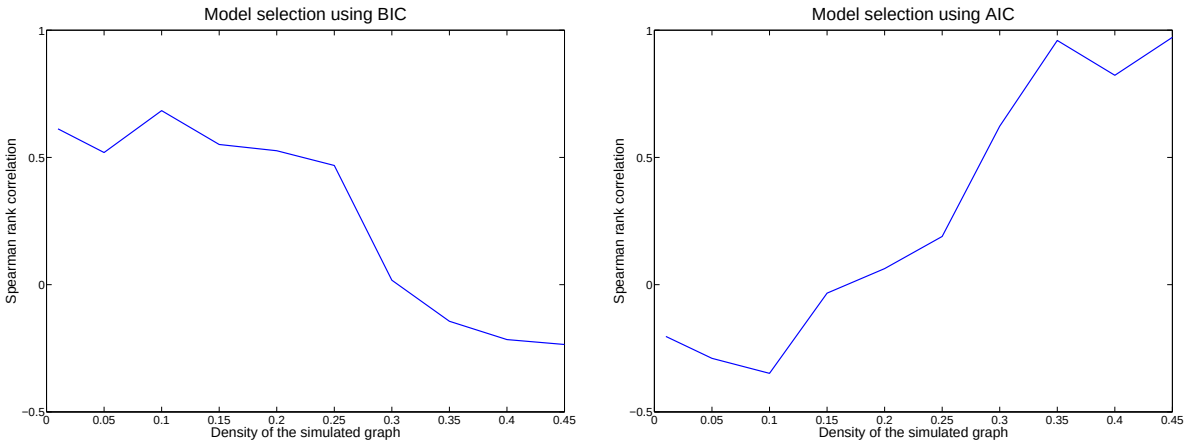


Figure 12: F-scores of structures learnt by using L_1-L_2 norm The figure shows the average and standard deviation of the F-score across 20 different graphs as a function of ρ, the probability of edge-occurrence.



(A) (B)

Figure 13: Graph density versus the rank correlation for ranking and selection using (A) BIC (B) AIC.

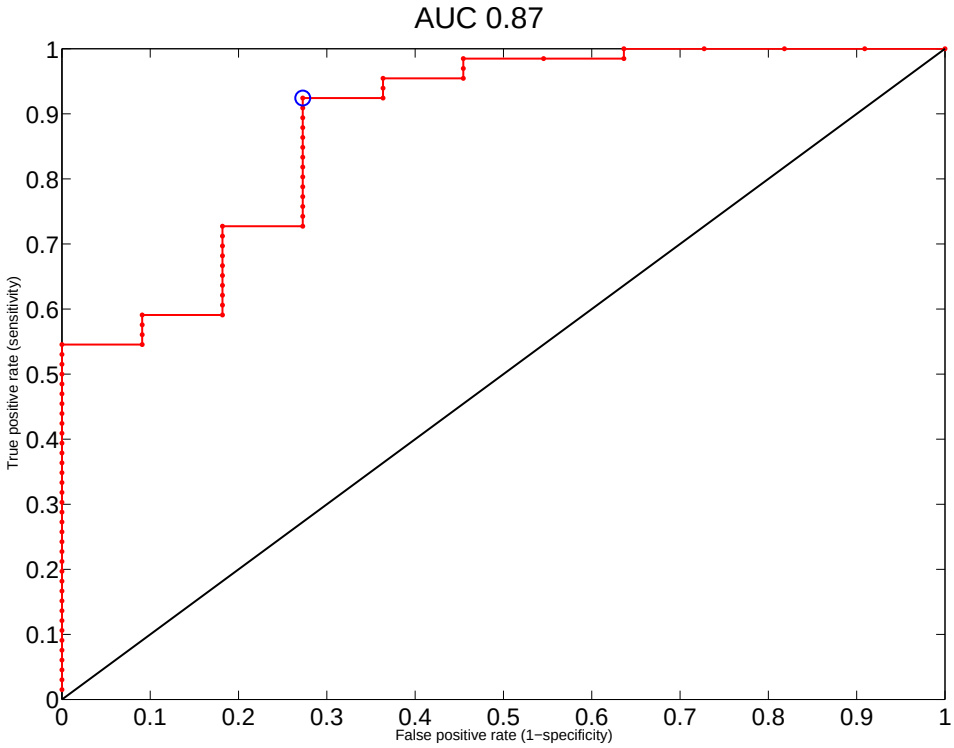


Figure 14: Receiver operating characteristic (ROC) curve of GREMLIN for the task of distinguishing artificial WW sequences that fold from those that don't.