A Bayesian Approach to Protein Model Quality Assessment

Hetunandan Kamisetty HETU@CS.CMU.EDU
5000 Forbes Avenue, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213

Christopher J. Langmead CJL+@CS.CMU.EDU
5000 Forbes Avenue, Computer Science Department and Lane Center for Computational Biology, Carnegie Mellon University, Pittsburgh, PA 15213

Abstract

Given multiple possible models $\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n$ for a protein structure, a common sub-task in *in-silico* Protein Structure Prediction is ranking these models according to their quality. Extant approaches use MLE estimates of parameters $\mathbf{r}_i$ to obtain point estimates of the Model Quality. We describe a Bayesian alternative to assessing the quality of these models that builds an MRF over the parameters of each model and performs approximate inference to integrate over them. Hyper-parameters $\mathbf{w}$ are learnt by optimizing a list-wise loss function over training data. Our results indicate that our Bayesian approach can significantly outperform MLE estimates and that optimizing the hyper-parameters can further improve results.

1. Introduction

The protein structure prediction problem is one of the most challenging unsolved problems in Biology. Informally, it is the task of computing the three dimensional structure of a protein, given its chemical description as a sequence of amino acids. Knowing the three dimensional structure of a protein can provide deep insights into its working, and the mechanisms of its interaction with the environment. This can be used, for example, in the design of new drugs and bio-sensors. Unfortunately, despite significant progress, experimental methods (i.e., X-ray crystallography and Nuclear Magnetic Resonance) to *determine* protein structures still require months of effort and $O(\$100K)$ — per protein. Therefore there has been a lot of focus on *in-silico* approaches to Protein Structure *Prediction*.

Given a protein sequence s, a common feature of structure prediction algorithms is the ability to (stochastically) generate a large number of putative models, $\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n$, and then assess the quality of each of these models using a ranking algorithm. Extant algorithms rank by first computing the optimal set $\mathbf{r}_{i_{opt}}$ of parameters $\mathbf{r}_i$ and then computing a point estimate $E(\mathbf{b}_i, \mathbf{r}_{i_{opt}}, \mathbf{s})$ of the model quality. A natural question to ask, is if a Bayesian estimate of model quality can be computed efficiently. Such an estimate involves computing an integral over all possible $\mathbf{r}_i$, a computation that is infeasible to perform exactly for protein structures.

There are a variety of computational techniques for *estimating* this integral in the structural biology community. The more accurate amongst them require extensive sampling or molecular dynamics simulations (e.g., (Alder & Wainwright, 1959)), which can take hours to days on real-proteins, making them infeasible for the task of *in-silico* Protein Structure Prediction. Faster coarse-grained methods exist, e.g., (Muegge, 2006), but it has been argued (Thomas & Dill, 1994) that they are not accurate enough.

In contrast to these techniques, we estimate the integral for each $\mathbf{b}_i$ by first discretizing $\mathbf{r}_i$ and then performing approximate inference on a discrete Markov Random Field constructed over $\mathbf{r}_i, \mathbf{s}$. The act of discretizing induces an error; we show how using a prior distribution eliminates this source of error. We then learn the hyper-parameters of our model by minimizing a loss-function for ranking over training data, using gradient descent.

While we won't discuss it in detail in this paper, there is a very strong motivation based on statistical physics to compute a Bayesian estimate. These Bayesian esti-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.
This research was sponsored in part by a grant from Microsoft Research to CJL.

mates are referred to as free-energies in that literature and are quantities that govern the behavior of physical systems. Significantly, the approximate inference algorithms that we use to compute these Bayesian estimates are mathematically equivalent to specific free-energy approximations introduced by statistical physicists. For example, it is now known that Pearl's *Belief Propagation* (BP) algorithm (Pearl, 1986) that we use in this paper computes the Bethe approximation (Bethe, 1935) of the free energy. Thus, there is reason to believe that these approximations are physically valid.

Our results on a database of *in-silico* models for 32 proteins show that moving from a point estimate to a Bayesian estimate improves the accuracy of ranking by multiple criteria: the average rank of best model improves from nearly 27 to less than 7, the model ranked first is significantly better in quality, and the rank correlation improves by nearly 0.3 over point estimates.

To summarize,

- We describe a Bayesian approach to assessing Protein Model Quality by using approximate inference in a discrete MRF to integrate out model parameters.
- We identify and address an important issue that arises when computing partition functions of discretized configuration spaces.
- We develop an algorithm for learning to rank partition functions based on optimizing a list-wise loss function.
- We establish the utility of our approach by showing that ranking accuracies significantly improve on a dataset of models for 32 different proteins.

2. A Markov Random Field Model for Proteins

In this section, we review some basic information about protein structures and describe our approach to computing the Bayesian estimate G($\mathbf{b}$) of quality of a model $\mathbf{b}$.

A protein consists of some number of amino acids across one or more polypeptide *chains*. Each amino acid comprises of some number of atoms. It is customary to partition the set of atoms into two disjoint sets: *backbone* and *side-chain*. *Backbone atoms* refer to those that are common to all 20 amino acid types, while *side-chain* atoms are those that differ among the different kinds of amino acids. A *configuration* of the protein corresponds to the geometry of each of its constituent atoms. We will use $\mathbf{s}$ to denote the amino acid sequence of a protein and $\mathbf{b}, \mathbf{r}$ to denote the configuration of all backbone and side-chain atoms in the protein respectively. Additionally, we will use superscripts (s^u) to denote the corresponding variables at a specific position.

A common and convenient approach to modeling the protein structure prediction problem is to first determine a possible configuration of the backbone atoms $\mathbf{b}$ for protein sequence $\mathbf{s}$, and then determine the optimal configuration of the side-chain atoms $\mathbf{r}_{opt}$ for this set of backbone atoms The quality of the model is then estimated using a function $E(\mathbf{b}, \mathbf{r}_{opt}, \mathbf{s})$ which computes an energy for the configuration. Given multiple possible configurations $\mathbf{b}_1, \mathbf{b}_2, \ldots . \mathbf{b}_n$, the configuration with the best (i.e., lowest) $E(\mathbf{b}_i, \mathbf{r}_{i_{opt}}, \mathbf{s})$ is considered the best guess for the (unknown) structure of the protein.

From a statistical standpoint, each of these $\mathbf{b}_i$ can be considered a model for the data $\mathbf{s}$, $\mathbf{r}_i$ can be considered the parameters of the model, and $E(\mathbf{b}_i, \mathbf{r}_{i_{opt}})$, a point estimate of the model quality. A natural question to ask then, is if a Bayesian estimate of model quality is possible, and if so, if it could be computed efficiently. This paper addresses these questions.

To do this, using Boltzmann's law, we first define a probability distribution over the $\mathbf{r}, \mathbf{s}$ variables:

$$P(\mathbf{r}, \mathbf{s}|\mathbf{b}) \propto \exp(-E(\mathbf{b}, \mathbf{r}, \mathbf{s})) \tag{1}$$

For a Bayesian estimate of model quality, we need to integrate over all parameters $\mathbf{r}$, i.e. we need to compute $\int_{\mathbf{r}} P(\mathbf{r}, \mathbf{s}|\mathbf{b}) d\mathbf{r} \propto \int_{\mathbf{r}} \exp(-E(\mathbf{b}, \mathbf{r}, \mathbf{s})) = Z_{\mathbf{b}} = \exp(-G(\mathbf{b}))$ i.e, we need to compute the partition function $Z_{\mathbf{b}}$ over the side-chain conformational space consistent with $\mathbf{s}$ given a model $\mathbf{b}$.

Computing $Z_{\mathbf{b}}$, or equivalently, the negative log partition function $G(\mathbf{b})$ thus involves computing an integral over an extremely large state space. In what follows, we will approximate this integral by first discretizing the space and representing the probability distribution using a discrete Markov Random Field (MRF), and then using approximate inference to approximate the discrete summation over the distribution.

The assumption of a discrete library of possible side-chain configurations (called *rotamers*) is common, and well-founded physically (cf. (Canutescu et al., 2003)). Yanover and Weiss (2003) use this discretization in a graphical model to determine the optimal set of parameters (i.e. $\mathbf{r}_{i_{opt}}$), which can then be used to deter-

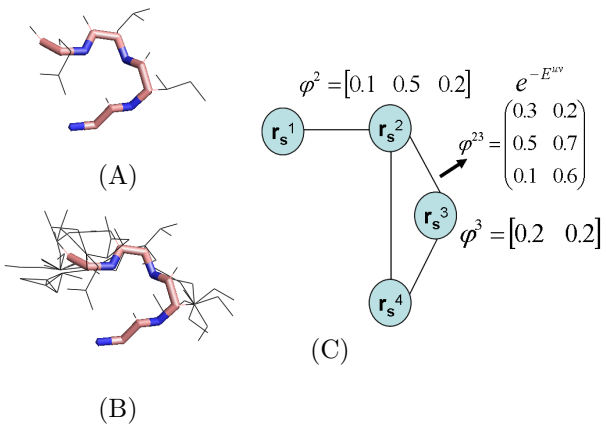

Figure 1. (A) An example **b**(in color), **r** (in black) for a small, 4 amino acid fragment of a protein. (B) Figure showing configurations of **r** consistent with **s**. (C) A graphical model encoding the conditional Boltzmann distribution over $\mathbf{r_s}$. For notational, and visual clarity, the **b** variable that is observed, is not shown in the MRF.

mine a point estimate of model quality. In contrast, we use a graphical model to compute a *Bayesian* estimate by marginalizing out $\mathbf{r}_i$. (Canutescu et al., 2003) provide a discrete set of conformations for each of the 20 naturally occurring amino acids. The number of discrete conformations varies from amino acid to amino acid – some amino acids can have as many as 81 configurations while others might have only one.

Given a specific backbone configuration **b**, due to the nature of the physical forces in action, pairs of amino-acids distally located according to **b** are expected to exert very little direct influence on one another. In statistical terms, we say that such residues are conditionally independent of each other given **b**. We will exploit these conditional independencies present in $P(\mathbf{r}, \mathbf{s}|\mathbf{b})$ to compactly encode it as a Markov Random Field over the $\mathbf{r_s}$ variables[1]. Since only a subset of the **r** conformations are possible for a given sequence, we build an MRF over $\mathbf{r_s}$, the set of side-chain conformations that have non-zero probability for sequence **s**.

The MRF has single and pair-wise factors ϕ^u, ϕ^{uv} of the form $\phi^u(\mathbf{r}_\mathbf{s}^u) = \exp\left(-E(\mathbf{b}, \mathbf{r}_\mathbf{s}^u)\right)$ and $\phi^{uv}(\mathbf{r}_\mathbf{s}^u, \mathbf{r}_\mathbf{s}^v) = \exp(-E(\mathbf{b}, \mathbf{r}_\mathbf{s}^u, \mathbf{r}_\mathbf{s}^v))$. where $E(\cdot)$ is the energy of those atoms as defined by the ROSETTA force-field $E_{Rosetta}$. $E_{Rosetta}$ is a linear combination $\mathbf{w}^T\mathbf{E} = w_{ljatr}E_{ljatr} + w_{ljrep}E_{ljrep} + w_{sol}E_{sol} + w_{hb}E_{hb} + w_{dun}E_{dun}$ where E_{ljatr}, E_{ljrep}, are the attractive and repulsive parts of a $6-12$ Lennard-Jones potential used to model van der Waals interactions; E_{sol}, is the Lazardus-Karplus solvation energy and E_{hb}, is the Hydrogen bond energy. The vector **w** that defines the linear combination is a hyper-parameter of the model, which we will learn using training data.

Notice that the due to the choice of the Boltzmann factor for Φs, this distribution is consistent with the Boltzmann distribution of Eq. 1.

Fig. 1 illustrates the construction of $\mathcal{G}$ using a toy protein with 4 amino acids. The MRF has one vertex for each amino acid, and edges between vertices that "interact". The MRF has single-node and pair-wise potentials, each defined in terms of the Boltzmann factor $\exp(-E^{uv})$ as shown in the figure.

3. Approximating the log-partition function

Computing the log-partition function is computationally intractable in the general case (Dagum & Chavez, 1993). However, there exist a number of efficient approximation algorithms for performing probabilistic inference over MRFs which can be used to compute an approximation to the log-partition function.

Probabilistic inference in an MRF involves computing marginal distributions over the random variables in the graph. Inference algorithms implicitly or explicitly obtain an estimate of the log-partition function. For example, sum-product BP, which we use in this paper, performs inference by minimizing the difference between the log-partition function and a functional F_P of the form $-\langle E\rangle_P + S_P$ where $\langle E\rangle_p$ and S_P are, respectively, the expected energy and the entropy of the current distribution P. The value of this functional equals the discrete log-partition function exactly when the current distribution equals the actual distribution.

Therefore, by using Loopy Belief Propagation on the MRF that encodes the conditional distribution, and computing the functional at convergence, we can obtain an estimate of the partition function $Z_\mathbf{b}$ for each backbone configuration **b**. While Loopy BP is not guaranteed to converge, it has always done so in our experiments.

On a simpler model, we have previously shown the efficacy of Loopy BP on computing folding free energies of protein structures. We have also demonstrated the utility of the entropic component of the functional in *distinguishing* the native structure from near-native models (Kamisetty et al., 2008; Kamisetty et al., 2007). Our current approach extends this to

[1]Since **b** is observed, its energetic contribution can be moved into other factors (Kamisetty et al., 2008). Thus we will drop its explicit mention in the MRF, in order to improve notational clarity.

the *ranking* problem by significantly enhancing the model and introducing a learning algorithm to learn the hyper-parameters.

3.1. Discretization

We now briefly discuss a subtle, yet important issue that we have glossed over so far in our presentation: the effects of discretizing the parameter space over $\mathbf{r}$.

This approach of using a set of discrete rotameric states to compute the entropy faces a subtle problem. If we call $\log Z_{cont}$ the log-partition obtained by computing an integral, and $\log Z_{discrete}$ the estimate of the log-partition function obtained by computing the functional F, we have the following theorem:

Theorem 1. *$|\log Z_{cont} - \log Z_{discrete}|$ is unbounded from above*

Proof. Consider a single random variable X with a pdf of the form $\frac{1}{Z}e^0$, i.e. with a uniform distribution over its state space. Consider its discretized counterpart, a discrete random variable X_{disc} having state space of size n, each representing an equal fraction of the continuous state space with a uniform distribution U over this state space.

Consider the functional $F_U = -\langle E \rangle_U + S_U$. It is easy to see that $\langle E \rangle_U = \langle 0 \rangle_U = 0$. The entropy S_U, on the other hand $= -\sum_{i=1}^{n} -\frac{1}{n}\log(\frac{1}{n}) = -\log(n)$.

Thus $n \longrightarrow \infty$, $S \longrightarrow -\infty$ leading to the theorem. □

In other words, as the granularity of the discretization increases arbitrarily, the discrete entropy increases arbitrarily. This is, of course, at odds with our intuition that the original continuous variables have finite amount of information, or entropy. This problem arises in many scenarios, most notably for our purposes, in information-theoretic treatments of statistical physics (Jaynes, 1963; Jaynes, 1968). Fortunately, a solution to this problem is available, which to the best of our knowledge is due to E.T. Jaynes (1963).

Theorem 2 (Relative Entropy)**.** *The discrete relative entropy $S_n^{rel} = -\sum_{i=1}^{n} P(x_i)\log\frac{P(x_i)}{m(x_i)}$, with respect to a measure m approaches $S_{continuous}^{rel}$ as $n \longrightarrow \infty$.*

Proof. Proof due to E.T. Jaynes (1963). □

Thus, by using a measure m over the configurational space and replacing the discrete entropy by the relative entropy $S = -\sum_{\mathbf{r}} P(\mathbf{r})\log\frac{P(\mathbf{r})}{m(\mathbf{r})}$, we now obtain a quantity that behaves correctly in the limit. To use this for our purposes, we point out that the library of discrete conformations that we use (Canutescu et al., 2003) provides such a measure m_{dun}, which we utilize.

Our earlier treatment of inference can be modified to use the relative entropy instead of the discrete entropy, by observing that

$$S = -\sum_{\mathbf{r}}(P(\mathbf{r})\log P(\mathbf{r}) - P(\mathbf{r})\log m(\mathbf{r}))$$

and therefore, $F_P =$

$$\sum_{\mathbf{r}} P(\mathbf{r})E_{\mathbf{r}} + \sum_{\mathbf{r}}(P(\mathbf{r})\log P(\mathbf{r}) - P(\mathbf{r})\log P(m(\mathbf{r})))$$
$$= \sum_{\mathbf{r}} P(\mathbf{r})(E_{\mathbf{r}} - \log m(\mathbf{r})) - \sum_{\mathbf{r}} P(\mathbf{r})\log P(\mathbf{r})$$

In other words, the move from the discrete entropy to the discrete relative entropy can be made by adding a $-\log m(\mathbf{r})$ term to the energy function. Furthermore, due to the properties of the measure m_{dun} we use in practice, any $(m_{dun})^{w_{dun}}$ is also a valid measure; we can therefore use any linear combination $w_{dun}E_{dun}$ in the energy function leading to E, the "pseudo" energy function being $E_{Rosetta} + w_{dun}E_{dun}$.

4. Learning to Rank (log) Partition Functions

Given the partition function (or equivalently, $G_{\mathbf{b}}$) for each model, our next step is to learn a set of hyper-parameters $\mathbf{w}$ that performs well on ranking tasks.

Given models $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_n\}$, and a ranking (permutation over $1 \ldots n$) $\mathbf{y}$ for these models, the learning task involves finding a function G that computes a numerical score for each model that minimizes some loss-function $\mathcal{L}$ between G and $\mathbf{y}$ on $\mathbf{B}$. Many approaches have been developed for the task of learning to rank, especially in IR tasks like document-retrieval (Herbrich et al., 2000) and web-search (Joachims, 2002). These tasks differ in their choice of the loss function $\mathcal{L}$ and the algorithms used to minimize it. While initial approaches to ranking approached the ranking problem as a large number of pair-wise classifications (Herbrich et al., 2000; Joachims, 2002), recent approaches have shown the utility of using loss-functions based on the entire rank, or the so-called "list-wise" approaches (Cao et al., 2007; Xia et al., 2008). Further, a "soft" approach to ranking (Burges et al., 2005) has allowed the use of gradient-based continuous optimization techniques instead of combinatorial optimization.

We use a "list-wise" soft-ranking approach to ranking since it has been shown to have good performance. We

study the properties of two loss-functions under this approach – the negative log-likelihood and the cross-entropy as described below.

4.1. Negative Log-Likelihood

The negative log-likelihood loss defines a probability distribution over all rankings (permutations) and attempts to maximize the likelihood of the true ranking. The loss-function for the models $\mathbf{B}$ of a single protein is defined as $\mathcal{L}_{negLL}(G(\mathbf{B}), \mathbf{y}) = -\log P(\mathbf{y}|\mathbf{B}; G)$ where

$$P(\mathbf{y}|\mathbf{B}; G) = \prod_{i=1}^{n} \frac{\exp(-G(\mathbf{b}_{y_i}))}{\sum_{k=1}^{n} \exp(-G(\mathbf{b}_{y_k}))}$$

and $P(\mathbf{y}|\mathbf{b}; G)$ is the probability of seeing the observed ranking $\mathbf{y}$. The probability distribution defined over the permutations follows a Plackett-Luce model (Marden, 1995).

The notable advantages of this loss-function are that it is convex, consistent, sound, and efficient to compute (Xia et al., 2008).

4.2. Cross-Entropy

The cross-entropy loss assumes the existence of a scoring function ψ, that retains the order of the permutation $\psi(y_1) < \psi(y_2) < \ldots < \psi(y_n)$.

This scoring function is then used to compute a probability distribution over all permutations π.

$$P(\pi|\mathbf{b}; \psi_{\mathbf{y}}) = \prod_{i=1}^{n} \frac{\exp(-\psi_{\mathbf{y}}(\mathbf{b}_{\pi_i}))}{\sum_{k=1}^{n} \exp(-\psi_{\mathbf{y}}(\mathbf{b}_{\pi_k}))}$$

Using a similar probability distribution this time derived using G,

$$P(\pi|\mathbf{b}; G) = \prod_{i=1}^{n} \frac{\exp(-G(\mathbf{b}_{\pi_i}))}{\sum_{k=1}^{n} \exp(-G(\mathbf{b}_{\pi_k}))}$$

we can then compute a loss-function based on the KL-Divergence between the two distributions.

Since ψ and $\mathbf{y}$ are given, the KL-Divergence can be simplified by dropping to self-entropy term, leading to the cross-entropy loss function: $\mathcal{L}(G(\mathbf{b}), \mathbf{y}) = D(P(\pi|\mathbf{b}; \psi_{\mathbf{y}})||P(\pi|\mathbf{b}, G))$

Note however, that the cross-entropy between distributions over all permutations cannot be computed efficiently in practice since there are $n!$ of them. A common practice is to therefore use a probability distribution (the so-called top-one distribution) over the objects as follows $P_\psi(j) = \frac{\exp(-\psi(j))}{\sum_{i=1}^{n} \exp(-\psi(i))}$ and a similar distribution P_G and use the cross-entropy between them: $\mathcal{L}_{crossEnt}(\mathbf{y}, G_{\mathbf{B}}) = \sum_{i=1}^{n} P_\psi(\mathbf{b}_i) \log P_G(\mathbf{b}_i)$.

Cross-entropy as defined in this manner is also convex, sound, and efficient to compute. However, it is not consistent, implying that in the limit of the amount of data tending to ∞, it is not guaranteed to converge to the true values of the hyper-parameters. In contrast, a useful feature of this loss function is the ability to incorporate additional information about the models using ψ. We believe this is a strong advantage of this loss function. (Nallapati, 2006) studies the properties of this loss function in more detail.

4.3. Gradient Descent

In order to optimize the loss functions using gradient descent, we need to compute the derivative $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$. Utilizing the fact that

$$\frac{\partial G}{\partial \mathbf{w}} = -\frac{\partial \log Z_{\mathbf{b}}}{\partial \mathbf{w}} = -\frac{1}{Z_{\mathbf{b}}} \frac{\partial \sum_{\mathbf{r}} \exp(-\mathbf{w}^T E(\mathbf{r}, \mathbf{s}))}{\partial \mathbf{w}} = \langle \mathbf{E} \rangle_{\mathbf{b}} \tag{2}$$

where $\mathbf{E_b}$ is the vector containing the average of $\mathbf{E}(\mathbf{b}, \mathbf{r}, \mathbf{s})$ over all $\mathbf{r_s}$.

The required derivatives are then as follows:

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}_{negLL}(G(\mathbf{B}), \mathbf{y}) = \sum_{i=1}^{n} (\langle \mathbf{E} \rangle_{\mathbf{b}_i} - \frac{\sum_{k=1}^{n} Z_{\mathbf{b}_k} \langle E \rangle_{\mathbf{b}_k}}{\sum_{k=1}^{n} Z_{\mathbf{b}_k}}) \tag{3}$$

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}_{crossEnt}(G(\mathbf{B}), \mathbf{y}) = \sum_{i=1}^{n} -P_\psi(i) \langle \mathbf{E} \rangle_{\mathbf{b}_i} - \sum_{i=1}^{n} \frac{Z_{\mathbf{b}_i} \langle E \rangle_{\mathbf{b}_i}}{\sum_{k=1}^{n} Z_{\mathbf{b}_k}} \tag{4}$$

Given sets of models $\mathbf{B}_1, \ldots \mathbf{B}_d, \ldots \mathbf{B}_D$ for D distinct protein sequences in a database, the two loss functions (and therefore their derivatives) over the entire database are defined as the sum of the corresponding functions over each $\mathbf{B}_d$.

5. Implementation and Results

We computed energies using our implementation of the ROSETTA force-field as specified in (Kortemme & Baker, 2002). We used the `soft-rep` force-field setting since previous studies (Yanover et al., 2007) have indicated that it is better suited for computations with discrete conformations.

Recall that in our approach, the functions in $\mathcal{G}$ (i.e., ϕ^u, ϕ^{uv}) are defined in terms of a Boltzmann factor of $\exp(-\mathbf{w}^T\mathbf{E})$ where $\mathbf{w}^T\mathbf{E} = E_{Rosetta} + w_{dun}E_{dun}$ and $E_{Rosetta} = w_{ljatr}E_{ljatr} + w_{ljrep}E_{ljrep} + w_{sol}E_{sol} + w_{hb}E_{hb} + w_{dun}E_{dun}$. The vector of hyper-parameters

$\mathbf{w}$ that defines the linear combination is learnt by minimizing a loss function, as described in Sec. 4.

We studied the efficacy of our approach on a database of 32 proteins selected from (Wroblewska & Skolnick, 2007). For each protein, this database contains a set of 50 plausible models generated *in-silico* and the actual structure of the protein ("native"). Each of them contain $\mathbf{b}$ and $\mathbf{r}_{opt}$ and an associated "distance" – the root mean square displacement (RMSD) in Å (angstroms) between the coordinates of the atoms in the model to the native. When minimizing the cross-entropy loss function, we use the RMSD as the score, ψ, of the model. This dataset covers all four classes of proteins according to the CATH classification, and the models for each protein cover a large fraction of the model space – very close (< 2 RMSD) to very distant (> 10 RMSD). This is in contrast to the well-known and commonly used "Decoys R Us" collection of datasets, where most datasets are close homologs of each other and/or have very low variation in model space. Thus, we believe that we have chosen the most representative of the datasets available for this purpose[2].

We split this database of 32 proteins into five, randomly generated, equal-sized train/test partitions. Thus, each training set contains 16 proteins, containing the native and 50 plausible models for each of these proteins, along with the RMSD of these models to the native. Each test set contains 16 proteins containing 50+1 (in-silico+native) models which have to be ranked based on their quality. Ideally, the native should always be ranked first, along with the models in increasing order of RMSD to ground truth.

To perform approximate inference, we used Belief Propagation with synchronous updates. We used gradient descent to learn the hyper-parameters $\mathbf{w}$ for both loss functions listed earlier – Cross Entropy and Negative Log-likelihood. For gradient descent, η was set to $\eta_0/\sqrt{(i)}$ at iteration i with $\eta_0 = 0.1$. To test the sensitivity of our results to the choice of η_0, we performed gradient descent on one of the training sets increasing the value of η_0 from 0.05 to 0.25 in increments of 0.05. On this set, our final solution did not change appreciably, though the number of steps taken for convergence changed. On this basis, we believe that our solutions are fairly robust to the choice of η.

We compare the performance of four methods: (i) a point estimate obtained by computing $E_{Rosetta}$ on the optimized parameters, (ii) a Bayesian estimate G obtained using default hyper-parameters, and the Bayesian estimates of G using the learnt hyper-parameters with the two loss functions, which we shall refer to as (iii) G-neg LL and (iv) G-Cross Entropy.

[2]Supplementary information is available at `http://www.cs.cmu.edu/~cjl/papers/icml09suppl/`

Table 1. Rank of Native and Quality of Best Decoy across 5 test sets

Method	Rank of Native (out of 51 total)	Best - closest (RMSD)
Point Estimate	26.75	3.413
G	8.05	1.85
G-NegLL	21.19	3.71
G-Cross Entropy	6.60	1.743

Tab. 1 compares the average rank of the native structure across the 5 test sets and the average difference between the RMSD of the best *in-silico* model as predicted by the method, to the RMSD of the actual closest model. The average rank of the native structure is significantly improved by moving from a point estimate (26.75) to a Bayesian estimate (8.05). Further, by optimizing the hyper-parameters using the cross entropy loss function, this can be further improved to 6.6.

While the rank of the native is a useful metric of comparison, in practice, a more important metric than the rank of the native is the quality of the best *in-silico* model. The average difference in RMSD between the predicted best model and the actual closest model is significantly reduced, from 3.4 Å to nearly half its value – 1.85 Å using G and 1.74 Å using G-Cross Entropy.

Surprisingly, optimizing the hyper-parameters using the likelihood loss function is almost as bad as the point estimate, indicating that the likelihood loss function isn't suitable for this task. We believe that this is due to the fact the likelihood function neglects the RMSD information while computing the likelihood of a ranking. In a data-scarce setting such as ours, this could lead to a significant difference in the optimal solution.

Fig. 2 compares the hyper-parameters learnt for the two loss functions. In both cases, the weights of E_{ljatr}, E_{ljrep} and E_{lk} are significantly lower than the weights of E_{hb}, E_{dun}. Additionally, the $neg - ll$ loss function learns a significantly lower weight for E_{dun}. It must be pointed out that on typical proteins, the numerical contribution of the first three terms is one or two orders of magnitude more than the numerical contribution of the other two terms. Thus, while the weights for these terms are similar, the difference between them is significant enough to cause a large difference predictions. We believe that this could be one

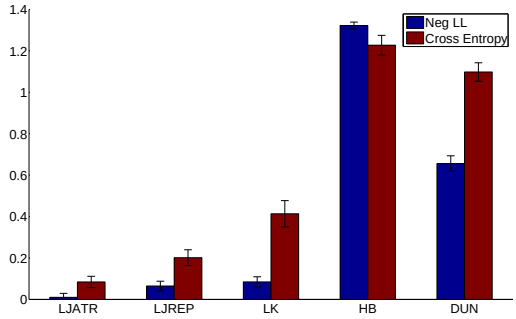

Figure 2. A comparison of the weights learnt using the two different loss-functions

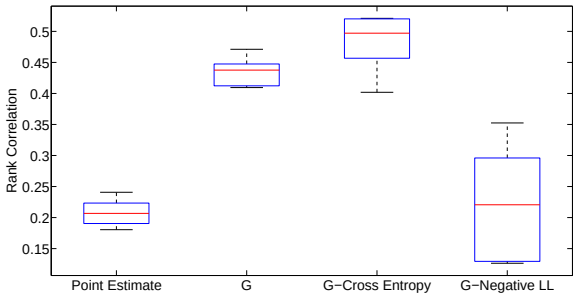

Figure 3. Pearson's rank correlation coefficient between predicted ranking and actual ranking for the four methods listed.

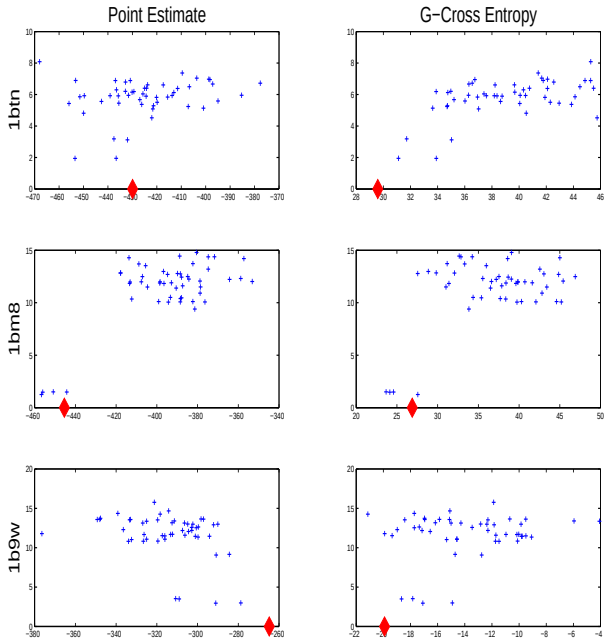

Figure 4. Scatter plots showing ranking for three different proteins in a test set, using the point estimate and the Bayesian estimate. The x-axis is the value of the corresponding estimate, while the y-axis shows the RMSD from the ground truth (shown with a red diamond)

of the reasons the negative-log likelihood loss function did not perform well.

While the rank of the top structure and the quality of top prediction are important metrics of performance, the quality of the overall ranking is also important. This is because, often, the models are iteratively generated, ranked, and selectively refined. Thus, it is important that the ranking is reasonably accurate at all positions.

To measure this, we compute the rank-correlation of the ranks with the ranks obtained by ranking according to RMSD from native. Fig. 3 shows the rank-correlation of the ranks computed in this manner. Again, it can be seen that the performance improves significantly by moving from a point-estimate to a Bayesian estimate, and learning using the cross-entropy loss function improves it further.

Fig. 4 shows the values of the point estimates (first column) and the Bayesian estimates learnt using the cross-entropy (second column), for three protein structures (rows) in a particular test set. The native structure in each of these proteins is shown as a red asterisk while the 50 in-silico models are shown in blue.

These three proteins were selected to show the different types of behavior in learnt ranking. In 1btn, the first protein, using the Bayesian estimate the native structure is ranked correctly, the best in-silico model is ranked next and there is a good correlation between G and the RMSD. In the second protein, 1bm8, the native structure is not ranked the best, but the best in-silico model is correctly identified. However, there is no strong correlation between RMSD and G for the distant models. Notice also, that in this case, the point estimate performs almost as well. In the third dataset, while the native structure is not ranked at the top, its rank is significantly better than using the point estimate. However, the model closest in RMSD is neither ranked well, nor is the top ranked structure of good quality. It must be noted that while there is variability in the ranking performance across the datasets, in all these cases, there is an improvement in results due to the Bayesian approach.

6. Conclusion

We have presented a Bayesian alternative to traditional methods for evaluating the quality of predicted protein structures. Our experimental results show that our approach significantly out performs MAP estimates of quality assessment. Additionally, we pre-

sented a practical algorithm for learning to rank using partition functions by optimizing a list-wise loss function over training data. We compared two loss functions, the negative log-likelihood and the cross entropy, and found that optimizing the cross-entropy objective function improves on the unoptimized hyperparameters.

Protein structure prediction is an important, and unsolved problem in Biology, and we believe that our method might be able to improve the accuracy of existing techniques. There are a number of areas for future improvement, the most important being incorporating Bayesian Model Averaging by modeling a limited amount of backbone flexibility.

References

Alder, B. J., & Wainwright, T. E. (1959). Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, *31*, 459–466.

Bethe, H. A. (1935). Statistical theory of superlattices. *Proc. Roy. Soc. London A*, *150*, 552–575.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *ICML '05: Proceedings of the 22nd international conference on Machine learning* (pp. 89–96).

Canutescu, A., Shelenkov, A. A., & Dunbrack Jr, R. L. (2003). A graph theory algorithm for protein side-chain prediction. *Protein Sci.*, *12*, 2001–2014.

Cao, Z., Qin, T., Liu, T., Tsai, M., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *Proceedings of the 24th international conference on Machine learning* (pp. 129–136).

Dagum, P., & Chavez, R. M. (1993). Approximating probabilistic inference in bayesian belief networks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, *15*, 246–255.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). *Large margin rank boundaries for ordinal regression.* MIT Press, Cambridge, MA.

Jaynes, E. T. (1963). Information theory and statistical mechanics. *Statistical Physics*, 181–218.

Jaynes, E. T. (1968). Prior probabilities. *Systems Science and Cybernetics, IEEE Transactions on*, *4*, 227–241.

Joachims, T. (2002). Optimizing Search Engines Using Clickthrough Data. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 133–142).

Kamisetty, H., Xing, E., & Langmead, C. (2007). Free Energy Estimates of All-atom Protein Structures Using Generalized Belief Propagation. *Proc. 7th Ann. Intl. Conf. on Research in Comput. Biol. (RECOMB)* (pp. 366–380).

Kamisetty, H., Xing, E. P., & Langmead, C. J. (2008). Free energy estimates of all-atom protein structures using generalized belief propagation. *J. Comp. Biol.*, *15*, 755–766.

Kortemme, T., & Baker, D. (2002). A simple physical model for binding energy hot spots in protein-protein complexes. *Proceedings of the National Academy of Sciences*, *99*, 14116–14121.

Marden, J. (1995). *Analyzing and Modeling Rank Data.* Chapman & Hall/CRC.

Muegge, I. (2006). PMF scoring revisited. *J. Med. Chem.*, *49*, 5895–5902.

Nallapati, R. (2006). *The Smoothed Dirichlet Distribution: Understanding Cross-entropy Ranking in Information Retrieval.* Doctoral dissertation, University of Massachusetts Amherst.

Pearl, J. (1986). Fusion, propagation, and structuring in belief networks. *Artif. Intell.*, *29*, 241–288.

Thomas, P. D., & Dill, K. A. (1994). Statistical potentials extracted from protein structures: How accurate are they? *Journal of Mol. Bio.*, *257*, 457–469.

Wroblewska, L., & Skolnick, J. (2007). Can a physics-based, all-atom potential find a protein's native structure among misfolded structures? i. large scale amber benchmarking. *Journal of Computational Chemistry*, *28*, 2059–2066.

Xia, F., Liu, T. Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. *ICML '08: Proceedings of the 25th international conference on Machine learning* (pp. 1192–1199). New York, NY, USA: ACM.

Yanover, C., Schueler-Furman, O., & Weiss, Y. (2007). Minimizing and learning energy functions for side-chain prediction. *Proc. 7th Ann. Intl. Conf. on Research in Comput. Biol. (RECOMB)* (pp. 381–395).

Yanover, C., & Weiss, Y. (2003). Approximate Inference and Protein Folding. *Advances in Neural Information Processing Systems(NIPS)*, 1481–1488.