Markov Logic Improves Protein β-Partners Prediction

Marco Lippi LIPPI@DSI.UNIFI.IT
Paolo Frasconi P-F@DSI.UNIFI.IT
Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze, Italy

Keywords: Protein structure, Markov logic networks, link prediction

Abstract

Protein β-partners prediction is an important problem in protein structure that can be naturally formulated as supervised link prediction. We show that prediction performance can be improved using a hybrid solution based on Markov logic networks with grounding-specific weights.

1. Introduction

Protein secondary structure is determined by hydrogen bonds between side-chain atoms. The two most common structures are α-helices, where amino acids are closely coiled, and β-sheets, where two or more extended strands of the macro-molecule are arranged in a flat conformation. Adjacent strands can either run in the same (parallel) or in the opposite (anti-parallel) direction. In most cases, each β-residue is linked by hydrogen bonds to one or two β-residues in different strands of the same chain, called its β-partners. An example of a β-partnership graph is shown in Fig. 1a.

Prediction of β-partners from protein sequence is an important task towards protein structure prediction and can be naturally formulated as supervised link prediction in a graph whose nodes are protein residues. Baldi et al. (2000) started from amino acid sequences and trained feedforward neural networks as binary classifiers on residue pairs, with inputs consisting of two windows of residues, each centered around one of the target residues. A difficulty of casting link prediction into a binary classification problem on pairs is that the resulting data set will be highly imbalanced. For examples, Baldi et al. (2000) reported 37,000 positive examples and 44 million negative examples for a data set of 826 chains (the number of non-partner pairs grows quadratically with the chain length). Additionally, treating pairs of residues as iid examples is prone to higher generalization error since linkage between targets is not taken into account in the learning process. Cheng and Baldi (2005) assumed both the amino acid and the secondary structure sequences to be known; they employed a different architecture called 2D-recursive neural network (2D-RNN), in which local inputs are also pairs of profile vectors and outputs associated to pairs of residues. A 2D-RNN has the structure of a two-dimensional grid and is trained with binary targets that correspond to the adjacency matrix of the β-partnership graph. Cheng and Baldi (2005) added a non-adaptive post-processor that collectively reassigns β links by means of an efficient graph matching algorithm enforcing some physical constraints derived from background knowledge. The resulting BetaPro predictor (Cheng & Baldi, 2005) reaches state-of-the-art performance.

However, BetaPro predictions can still contradict background knowledge and in some cases several errors are introduced while attempting to satisfy generic constraints. An example is shown in Fig. 1b for PDB entry 1BIK, where a single misprediction from the BetaPro first stage is amplified by the second stage resulting in a completely wrong set of partner assignments between strands A and C, which are in facts spatially far away. In order to further improve prediction accuracy for this class of bioinformatics problems, we advocate the use of statistical relational learning algorithms that can be collectively trained and that enforce constraints derived from background knowledge during learning. In particular, we focus here on Markov logic.

2. Methods

Given a set of first-order formulas $\mathcal{F}$, a Markov logic network (MLN; see (Domingos et al., 2008) for a review) defines a probability on each possible world. In the case of discriminant learning we are interested in here, we distinguish between evidence atoms (associated with predictive variables) and query atoms (associated with predictions). In this setting, an MLN is a model for the conditional distribution of the set of query atoms Y given the set of evidence atoms X, that

Preliminary work. Under review by the International Workshop on Mining and Learning with Graphs (MLG). Do not distribute.

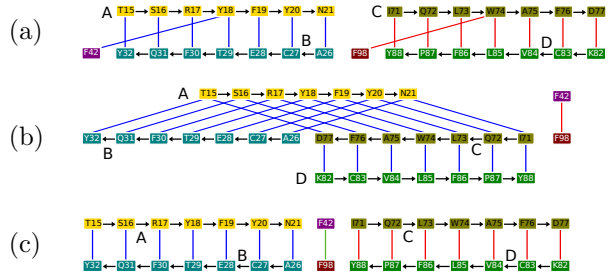

Figure 1. (a) β-partnership graph of PDB entry 1BIK. (b) β-partners predicted by BetaPro. (c) refined prediction obtained with Markov logic and grounding-specific weights derived from BetaPro predictions. Node colors correspond to strands.

can be conveniently expressed as a log-linear model:

$$P(Y=y|X=x) = \frac{e^{\sum_{F_i \in \mathcal{F}_y} w_i n_i(x,y)}}{Z_x} \quad (1)$$

being w_i a real-valued weight attached to formula F_i, $\mathcal{F}_y$ the set of formulas that contain query atoms, and $n_i(x,y)$ the total number of groundings of F_i that are satisfied in world (x,y).

It is often useful to declare formulae with constant-specific weights. This can be done for example in the Alchemy system (`http://alchemy.cs.washington.edu/`) on a per-variable basis, simply by macro expanding formulae containing variable names prefixed by the plus sign. In our application (like several other in bioinformatics), accurate predictions need a non-linear combination of several features obtained from windows of amino acids. One of the key formulae in this case would be

$$\texttt{Window(i,wi)} \wedge \texttt{Window(j,wj)} \rightarrow \texttt{Partners(i,j)} \quad (2)$$

where variables `wi` and `wj` represent the two windows of amino acids centered around positions i and `j`, respectively. Unfortunately, associating constant-specific weights to variables whose configurations take values on a large set is not practical. In (2) this would lead to a number of parameters growing exponentially with the windows size. Splitting the above formula into separate formulae for each residue position would reduce the number of weights but effectively generate a linear model in the amino acid features, while nonlinear combinations are known to be important for achieving high accuracy. In addition, performance is known to increase by replacing amino acid symbols in the window with real vectors encoding evolutionary information (e.g. in the form of multiple alignment profiles), but constant-specific weights cannot be immediately associated with real-valued variables.

We propose here to re-parameterize the MLN by computing each weight as a function of the specific grounding of selected variables in the formula:

$$P(Y=y|X=x) = \frac{e^{\sum_{F_i \in \mathcal{F}_y} \sum_j \omega_i(\mathbf{c}_{ij},\theta_i) n_{ij}(x,y)}}{Z_x} \quad (3)$$

where $\mathbf{c}_{ij}$ is the j-th ground configuration of the selected variables in the i-th formula, ω_i a parameterized function returning the weight attached to each ground formula where the selected variables are replaced by $\mathbf{c}_{ij}$, and $n_{ij}(x,y)$ is the number of true groundings in (x,y) matching $\mathbf{c}_{ij}$. Function ω_i can be implemented in many ways, for example as a kernel machine or as multilayered perceptron with input $\mathbf{c}_{ij}$ and weights θ_i. Standard MLN are recovered when ω_i is the constant function. In this formulation, the number of free parameters can be small even if the number of distinct configurations $\mathbf{c}_{ij}$ grows exponentially. Also, dealing with real-valued variables becomes straightforward.

Inference and learning algorithms are similar to those used for standard MLN. In particular, the MC-SAT algorithm can be applied for computing conditional probabilities and the (lazy) MaxWalkSAT algorithm for MAP inference (Domingos et al., 2008). When $\omega_i(\cdot)$ is realized by a multi-layered perceptron, it is possible to develop a learning procedure by first computing the gradient of the log-likelihood with respect to formula weights (as the difference between counts and expected counts of true groundings) and then propagating this gradient backward through the neural network. In the case of MAP inference, truth assignments rather than probabilities are computed. Therefore, only cases where the truth values of groundings in the data and in the inference are different contribute non-zero gradients. Interestingly, when applied to formulas that emulate the behavior of a traditional propositional learner as in (2), this setting can be interpreted as a form of active learning where MAP inference effectively selects training examples at each iteration.

In our experiments, we took the even simpler and pragmatic approach of using grounding-specific weights derived from BetaPro first stage (the 2D-RNN), which returns for each pair (i,j) a number $p_{ij} \in [0,1]$ interpreted as the Bernoulli conditional probability of partnership between residues i and j given the corresponding profile windows. In order to avoid over-predictions, BetaPro assigns a link whenever $p_{ij} > \tau$ with $\tau < 0.5$. We therefore rescaled p_{ij} by mapping the interval $[0,\tau]$ to $[0,0.5]$ and $[\tau,1]$ to $[0.5,1]$, and then applying the logit function to the result to obtain the weight in (2). Although BetaPro was trained independently and not taking into account other relational constraints such as those developed below, it is a highly engineered system producing state-of-the-art predictions. Our present aim is therefore to demonstrate how these predictions can be better exploited

in a statistical relational learning setting, leaving for future work the complete integration between the two systems with joint training.

A trick that we found useful to speedup the training procedure is the use of stochastic gradient ascent, where formula weights are updated after computing the gradient on the mini-batch of groundings corresponding to an individual protein chain (note that chains are independent in this application).

The query predicates of our inference process are both residue-level partnership and strand-level partnership, distinguishing the latter in two predicates one for parallel and one for anti-parallel strands. Both hard and soft constraints were integrated in the MLN used in the experiments. Hard constraints include simple properties of the `Partners` query predicate, like anti-reflexivity and symmetry. In addition, we forbade bonds between two residues in the same strand, and between a residue and two different residues belonging to same other strand. We finally prevented *crossing edges*: if $(\mathtt{i}, \mathtt{j})$ and $(\mathtt{i}+1, \mathtt{j}+1)$ are partner pairs, then $(\mathtt{i}+2, \mathtt{j}-1)$ is not.

Soft rules include some more specific properties of β-sheet structures. Adjacency rules assert that if `i` and `j` are partners, then also $\mathtt{i}+1$ and $\mathtt{j}+1$ [$\mathtt{i}+1$ and $\mathtt{j}-1$] are likely to be partners in parallel [anti-parallel] strands. The rule for the frequent β-hairpin pattern asserts that two strands separated by less than six residues (one of which is glycine or proline) are likely to be anti-parallel partners. Another rule is derived from the β-α-β pattern: if two strands surround a helix, then they are likely to be parallel.

3. Results and Discussion

We used the same data set as in (Cheng & Baldi, 2005) which consists of 916 protein chains, containing a total number of 48,996 β-residues participating in 31,638 interstrand residue pairs. The assignment of residues to secondary structure classes is given as evidence in this data set. The data set is already split in ten folds, so we could reproduce the same experiments described in (Cheng & Baldi, 2005). By re-training BetaPro, we could retrieve its predicted conditional probabilities and then use them as grounding-specific weights for training our MLN. We used a modified version of Alchemy with discriminative learning, and chose $3/n^2$ as learning rate, being n the number of β residues.

In Table 1 we report a paired 10-fold cross validation comparison between the MLN and BetaPro. We report precision (P), recall (R) and $F_1 = 2PR/(P+R)$ for β-partner assignments at the residue level. The

Table 1. Results of 10-fold cross validation.

	BetaPro	MLN
P	38.03 ± 0.02	48.03 ± 0.03
R	44.18 ± 0.02	38.87 ± 0.02
F_1	40.87 ± 0.02	42.96 ± 0.03
C_{10}	$46.6 \pm 0.04\%$	$54.8 \pm 0.07\%$
C_{20}	$84.3 \pm 0.03\%$	$87.3 \pm 0.03\%$
C_{50}	100%	100%

difference between the F_1 measures is significant with $p < 0.01$. The F_1 measure of BetaPro is essentially identical to that of its first stage (the 2D-RNN only). The second stage of BetaPro significantly improves the perfect strand matches (PSM; i.e. the number of β-strand pairs for which all the β-partnership predictions are correct), from $F_1 = 16.34$ to $F_1 = 30.91$. In this case, the denominator of precision is the number of strand pairs linked by at least one predicted β-partnership. The MLN also significantly improves the PSM of the underlying 2D-RNN, achieving $F_1 = 28.18$, but is outperformed in this task by the second stage of BetaPro. We finally measured the quality of the coarse partnership prediction at the strand level. The measure C_x in Table 1 represents the percentage of protein chains for which less than x% errors are made in the prediction of coarse links. Focusing on the high quality predition range (error less than 10%), the MLN correctly predicts about 8% more chains than BetaPro.

While encouraging, these results still do not fully take advantage of the potentials of the method. An ongoing line of investigation is the implementation of a hybrid model where multi-layered perceptrons are used to compute the grounding-specific weights. Another interesting direction is the implementation of a multi-task learning scheme where β-partners are jointly predicted together with other structural properties of proteins like secondary structure and solvent accessibility.

References

Baldi, P., Pollastri, G., Andersen, C. A., & Brunak, S. (2000). Matching protein beta-sheet partners by feedforward and recurrent neural networks. *Proc Int Conf Intell Syst Mol Biol*, *8*, 25–36.

Cheng, J., & Baldi, P. (2005). Three-stage prediction of protein beta-sheets by neural networks, alignments and graph algorithms. *Bioinformatics*, *21 Suppl 1*, i75–84.

Domingos, P., Kok, S., Lowd, D., Poon, H., Richardson, M., & Singla, P. (2008). Markov logic. In L. D. Raedt, P. Frasconi, K. Kersting and S. Muggleton (Eds.), *Probabilistic inductive logic programming*, 92–117. New York: Springer.