Learning Biophysically-Motivated Parameters for Alpha Helix Prediction

Blaise Gassend*[1], Charles W. O'Donnell*[1] , William Thies*[1], Andrew Lee[1], Marten van Dijk[1], Srinivas Devadas[1]

[1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

Email: Blaise Gassend*- gassend@mit.edu; Charles W. O'Donnell*- cwo@mit.edu; Wiliam Thies*- thies@mit.edu; Andrew Lee - andy_lee@mit.edu; Marten van Dijk - marten@mit.edu; Srinivas Devadas - devadas@mit.edu;

*Corresponding author

Abstract

Background: Our goal is to develop a state-of-the-art protein secondary structure predictor, with an intuitive and biophysically-motivated energy model. We treat structure prediction as an optimization problem, using parameterizable cost functions representing biological "pseudo-energies." Machine learning methods are applied to estimate the values of the parameters to correctly predict known protein structures.

Results: Focusing on the prediction of alpha helices in proteins, we show that a model with 302 parameters can achieve a Q_α value of 77.6% and an SOV_α value of 73.4%. Such performance numbers are among the best for techniques that do not rely on external databases (such as multiple sequence alignments). Further, it is easier to extract biological significance from a model with so few parameters.

Conclusions: The method presented shows promise for the prediction of protein secondary structure. Biophysically-motivated elementary free-energies can be learned using SVM techniques to construct an energy cost function whose predictive performance rivals state-of-the-art. This method is general and can be extended beyond the all-alpha case described here.

Background

It remains an important and relevant problem to accurately predict the secondary structure of proteins based on their amino acid sequence. The identification of basic secondary structure elements—alpha helices, beta strands, and coils—is a critical prerequisite for many tertiary structure predictors, which consider the complete three-dimensional protein structure. To date, there has been a broad array of approaches to secondary structure prediction, including statistical techniques, neural networks, hidden Markov models, support vector machines, nearest neighbor methods and energy minimization. In terms of prediction accuracy, neural networks are among the most popular methods in use today [1, 2], delivering a pointwise prediction accuracy (Q_3) of about 77% and a segment overlap measure (SOV) [3] of about 74% [4].

However, to improve the long-term performance of secondary structure prediction, it likely will be necessary to develop a cost model that mirrors the underlying biological constraints. While neural networks offer good performance today, their operation is largely opaque. Often containing up to 10,000 parameters and relying on complex layers of non-linear perceptrons, neural networks offer little insight into the patterns learned. Moreover, they mask the shortcomings of the underlying models, rendering it a tedious and ad-hoc process to improve them. In fact, in the past 15 years, the largest improvements in neural network prediction accuracy have been due to the integration of homologous sequence alignments [4] rather than specific changes to the underlying cost model.

In our approach we focus on simpler, more natural cost models that are based on the underlying biophysics. Due to the lack of experimentally determined free energy values, we begin with parameterizable cost functions, and treat parameter value estimation as an optimization problem. Our goal is then to determine the values of these "pseudo-energies" such that they correctly predict known protein structures. An iterative constraint-based optimization method is used to do this machine learning, incorporating the power of Support Vector Machines (SVMs).

Using a cost function based on Hidden Markov Models (HMMs), we develop a secondary structure predictor for all-alpha proteins. With only 302 parameters, representing the energetic benefit for each residue being in a helix or being a certain distance from the N- or C-cap, our predictor achieves a Q_α value of 77.6% and a SOV_α score of 73.4% when applied to a database of all-alpha proteins. Our technique does not depend on any homologous sequence alignments. When compared to other methods that do not utilize alignment information, it appears that our Q_α represents a 3.5% improvement of the previous best [5], while our SOV_α is comparable (0.2% better). However, due to differences in the data set, we emphasize the

novelty of the approach rather than the exact magnitude of the improvements. We are extending our technique to beta strands (and associated data sets) as ongoing work.

Related Work

King and Sternberg share our goal of identifying a small and intuitive set of parameters in the design of the DSC predictor [6]. DSC is largely based on the classic GOR technique [7], which tabulates (during training) the frequency with which each residue appears at a given offset (-8 to +8) from a given structure element (helix, strand, coil). During prediction, each residue is assigned the structure that is most likely given the recorded frequencies for the surrounding residues. King and Sternberg augment the GOR algorithm with several parameters, including the distance to the end of the chain and local patterns of hydrophobicity. They use linear discrimination to derive a statistically favorable weighting of the parameters, resulting in a simple linear cost function; they also perform homologous sequence alignment and minor smoothing and filtering. Using about 1,000 parameters, they estimate an accuracy of $Q_\alpha = 73.5\%$ for DSC. The primary difference between our predictor and DSC is that we achieve comparable accuracy (our $Q_\alpha = 77.6\%$) without providing alignment information. Incorporating an alignment profile is often responsible for 5-7% improvement in accuracy [8, 9, 10]. In addition, we learn the position-specific residue affinities rather than using the GOR frequency count. We also consider multiple predictions simultaneously and maintain a global context rather than predicting each residue independently.

Many researchers have developed Hidden Markov Models (HMMs) for secondary structure prediction. Once it has been trained, our predictor could be converted to an HMM without losing any predictive power, as our dynamic programming procedure parallels the Viterbi algorithm for reconstructing the most likely hidden states. However, for the training phase, our system represents a soft-margin Hidden Markov SVM [11] rather than a traditional HMM. Unlike an HMM, a Hidden Markov SVM has a discriminative learning procedure based on a maximum margin criterion and can incorporate "overlapping features", driving the learning based on the overall predicted structure rather than via local propagation.

Tsochantaridis, Altun and Hofmann apply an integrated HMM and SVM framework for secondary structure prediction [12]. The technique may be similar to ours, as we are using their SVM implementation; unfortunately, there are few details published. Nguyen and Rajapakse also present a hybrid scheme in which the output of a Bayesian predictor is further refined by an SVM classifier [13]. The Q_α score is 74.1% for the Bayesian predictor alone and 77.0% for the Bayesian/SVM hybrid; the SOV_α score is 73.2% for the Bayesian predictor and a comparable 73.0% for the Bayesian/SVM hybrid. To the

best of our knowledge, these are the highest Q_α and SOV_α scores to date (as tested on Rost and Sander's data set [9]) for a method that does not utilize alignment information.

Bystroff, Thorsson, and Baker design an HMM to recognize specific structural motifs and assemble them into protein secondary structure predictions [14]. Using alignment profiles, they report an overall Q_3 value of 74.3%. Our approach may use fewer parameters, as they manually encode each target motif into a separate set of states. Martin, Gibrat, and Rodolphe develop a 21-state HMM model with 471 parameters that achieves an overall Q_3 value of 65.3% (without alignment profiles) and 72% (with alignment profiles) [15]. Alpha helices are identified based on an amphiphilic motif: a succession of two polar residues and two non-polar residues. Won, Hamelryck, Prügel-Bennet and Krogh give a genetic algorithm that automatically evolves an HMM for secondary structure prediction [16, 17]. Using alignment profiles, they report an overall Q_3 value of 75% (only 69.4% for helices). They claim that the resulting 41-state HMM is better than any previous hand-designed HMM. While they restrict their HMM building blocks to "biologically meaningful primitives", it is unclear if there is a natural energetic interpretation of the final HMM.

Schmidler, Liu, and Brutlag develop a segmental semi-Markov Model (a generalization of the HMM), allowing each hidden state to produce a variable-length sequence of the observations [18, 19]. They report a Q_3 value of 68.8% without using alignment profiles. Chu and Ghahramani push further in the same direction, merging with the structure of a neural network and demonstrating modest ($\sim$1%) improvements over Schmidler et al. [20].

While our technique is currently limited to an alpha helix predictor, for this task it performs better ($Q_\alpha = 77.6\%$) than any of the HMM-based methods described above; furthermore, it does so without any alignment information. Our technique is fundamentally different in its use of Hidden Markov SVMs for the learning stage. Lastly, some groups have applied HMM-based predictors to the specific case of transmembrane proteins, where much higher accuracy can be obtained at the expense of generality [21].

There has been a rich and highly successful body of work applying neural networks to secondary structure prediction. The efforts date back to Quian and Sejnowski, who design a simple feed-forward network for the problem [22]. Rost and Sander pioneered the automatic use of multiple sequence alignments to improve the accuracy as part of their PHD predictor [9], which was the top performer at CASP2. More recently, Jones employed the PSI-BLAST tool to efficiently perform the alignments, boosting his PSIPred predictor [4] to the top of CASP3. Baldi and colleagues employ bidirectional recurrent networks in SSPro [23], a system that provided the foundation for Pollastri and McLysaght's Porter server [24].

Petersen describes a ballotting system containing as many as 800 neural networks; while an ensemble of predictors is commonly used to gather more information, this effort is distinguished by its size [25]. A neural network has been followed by an HMM, resulting in a simple and fast system [26]; neural networks have also been used as a post-processing step for GOR predictors [27].

The PSIPred predictor [4] is among the highest scoring neural network techniques. While it achieves an overall Q_3 of about 77% and an SOV of 74%, its performance for alpha helices is even higher: for recent targets on EVA, an open and automatic testing platform [28], PSIPred offers an SOV_α of 78.6% (EVA does not publish a Q_α value comparable to ours).

Though state-of-the-art neural network predictors such as PSIPred currently out-perform our method by about 5%, they incorporate multiple sequence alignments and are often impervious to analysis and understanding. In particular, the number of parameters in a neural network can be an order of magnitude higher than that of an HMM-based approach (see Table 1). A notable exception is the network of Riis and Krogh, which is structured by hand to reduce the parameter count to as low as 311 (prediction accuracy is reported at $Q_3 = 71.3\%$ with alignment profiles, a good number for its time).

Recently, Support Vector Machines (SVMs) have also been used as a standalone tool for secondary structure prediction [29, 30, 31, 32, 33, 34]. In contrast to our technique, which uses an SVM only for learning the parameters of an HMM, these methods apply an SVM directly to a window of residues and classify the central residue into a given secondary structure class. The number of parameters in these techniques depends on the number of support vectors; in one instance, the support vectors occupy 680MB of memory [30]. Regardless of the number of parameters, it can be difficult to obtain a biological intuition for an SVM, given the non-linear kernel functions and numerous support vectors. Nonetheless, these techniques appear to have significant promise, as Nguyen and Rajapakse report an overall Q_3 of 79.5% and an SOV of 76.3% on the PSIPred database [29].

Results and Discussion

We have applied our method to the problem of all-alpha protein secondary structure prediction. We worked with a set of 300 non-homologous all-alpha proteins taken from EVA's largest sequence-unique subset [35] of the PDB at the end of July 2005. The sequences and structures have been extracted from PDB data processed by DSSP [36]. Only alpha helices have been considered (`H` residues in DSSP files); everything else has been lumped as *coil* regions.

In our experiments, we split our 300 proteins into two 150 protein subsets. The first set is used to train our

parameterizable cost function; the second set is used to evaluate the cost function once its parameters have been learned. Since the results vary a bit depending on how the proteins are split in two sets, we train the cost function on 20 random partitions into training and test sets, and report the average performance.

Our predictor minimizes the free-energy function G using the Viterbi algorithm on a simple 7-state Finite State Machine (shown in Figure 1). The Finite State Machine recognizes alpha helices of length greater than 3 amino acids using 302 elementary free-energies as learned weights. These weigh each amino acid's propensity to be within a helix (20 energies), or within three residues of an N- or C-cap of a helix (20x7x2 energies). Two weights also penalize 1 and 2 length coils. The motivation for and implementation of the Finite State Machine is described in more detail later.

Table 2 presents our total results using both the Q_α and SOV_α metrics. Figures 2 and 3 show histograms detailing the distribution of each score. The Q_α metric is simply the number of correctly predicted residues divided by sequence length. SOV_α is a more elaborate metric that has been designed to ignore small errors in helix-coil transition position, but heavily penalize more fundamental errors such as gaps appearing in a helix [3].

On average, our method predicts helices in all-alpha proteins with an accuracy of 77.6% (Q_α) or 73.4% (SOV_α). Unfortunately, these results are difficult to compare with existing prediction methods which usually do predictions on both alpha helices and beta strands. Rost and Sanders caution that restricting the test set to all-alpha proteins can result in up to a 3% gain in accuracy [9]. Nonetheless, if one does compare our technique with the previous best amongst methods that do not utilize alignment information [5], our results represent a 3.5% improvement in Q_α and a 0.2% improvement in SOV_α. Additional care should be taken in comparing these numbers to recent techniques such as PSIPred [4], which consider 3_{10} helices (the DSSP state 'G') to be part of a helix rather than a loop; they report gains of about 2% in overall Q_3 if helices are restricted to 4-helices (as in most HMM techniques, including ours).

Apart from prediction accuracy, our technique is distinguished from others by its emphasis on an intuitive and biophysically-motivated cost function. While some of techniques require upwards of 10,000 parameters (see Table 1), our predictor achieves competitive accuracy using only 302 parameters.

The real power of the machine learning method we use is its applicability beyond HMM models. As will become evident in the description of the method, we could describe protein structures as a parse tree of a context-free grammar (or multi-tape grammar) rather than as a sequence of HMM states. With these enriched descriptions, we should be able to include in the cost function interactions between adjacent strands of a beta sheet. This should allow us to incorporate beta sheet prediction into our algorithm.

Unlike most secondary structure methods, we would then be able to predict not only which residues participate in a beta sheet, but also which residues are forming hydrogen bonds between adjacent sheets.

Conclusions

This work is a promising first pass at using SVM techniques to find the elementary free-energies needed to predict protein secondary structure. The method we use is general and can be extended beyond the all-alpha case described here. In future work, we plan to extend this method to super-secondary structure prediction, generating contact maps of individual hydrogen bonds in beta sheets.

Methods

It is widely believed that when a protein is folded, its free-energy approaches a thermodynamic minimum. We therefore treat structure prediction as an optimization problem.

Formal Optimization Problem

In our technique, we define a free-energy function $G(\mathbf{x}, \mathbf{y})$ that estimates the free-energy of an amino acid sequence $\mathbf{x}$ when folded into a candidate secondary structure $\mathbf{y}$. Our predictor outputs the secondary structure $\hat{\mathbf{y}}$ that has the minimal free-energy according to G:

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} G(\mathbf{x}, \mathbf{y}). \tag{1}$$

To go from this general statement to a working algorithm, we need to a find free-energy function G and a set of structures $\mathcal{Y}$ for which the minimization shown in equation (1) is easy to compute. In choosing G and $\mathcal{Y}$, we tradeoff the ability to efficiently minimize G with the ability to accurately capture the richness and detailed physics of protein structure. Atomistic models are able to capture the whole range of structures, and incorporate all the physical interactions between atoms. However, because of this detail they can only be optimized using heuristic methods. We therefore prefer to consider a simplified set of structures $\mathcal{Y}$, and a cost function G with lumped parameters that try to approach physical reality. These lumped parameters are difficult to determine experimentally. We will therefore define a class $\mathcal{G}$ of candidate free-energy functions that are easy to optimize over some set of structures $\mathcal{Y}$. Then we will use machine learning techniques to pick a good G from all the candidates in $\mathcal{G}$. The machine learning will use structure information from the Protein Data Bank (PDB) [37] to determine which G to pick. Given a set

of training examples $\{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, \ldots, k\}$, the learning algorithm needs to find a $G \in \mathcal{G}$ such that:

$$\forall i : \mathbf{y}_i = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} G(\mathbf{x}_i, \mathbf{y}). \tag{2}$$

In practice, this G may not exist or may not be unique, so the machine learning algorithm may have to pick a good approximation, or select a G that is more likely to generalize well to proteins not in the training set. We will now look more closely at how a good G is selected, and later, in Section , we will be more specific about what $\mathcal{G}$ and $\mathcal{Y}$ are.

Iterative Constraint Based Approach

First, we notice that equation (2) can be rewritten as the problem of finding a function G that satisfies the large set of inequality constraints

$$\forall i, \forall y \in \mathcal{Y} \setminus \{\mathbf{y}_i\} : G(\mathbf{x}_i, \mathbf{y}_i) < G(\mathbf{x}_i, \mathbf{y}). \tag{3}$$

Unfortunately, the set of all secondary structures $\mathcal{Y}$ is exponentially large, so finding a $G \in \mathcal{G}$ that satisfies all these inequalities directly is computationally intractable. Our approach reduces the problem by ignoring as many constraints as possible, only considering the constraints it is "forced" to consider.

In our method, the reduced problem is defined as the problem of finding a function G' that satisfies the set of constraints

$$\forall i, \forall y \in S_i : G'(\mathbf{x}_i, \mathbf{y}_i) < G'(\mathbf{x}_i, \mathbf{y}), \tag{4}$$

for some $S_i \subseteq \mathcal{Y} \setminus \{\mathbf{y}_i\}$.

Initially, we begin with no constraints at all (that is, $S_i = \emptyset$ for all i) and we choose some function $G' \in \mathcal{G}$. Note that, since we start with no constraints, any function $G' \in \mathcal{G}$ initially satisfies equation (4). We then need to check whether G' approximates the solution G to the set of constraints (2). In particular, we verify whether G' can be used to approximate $\mathbf{y}_1$ as the solution $\hat{\mathbf{y}}_1$ of the problem

$$\hat{\mathbf{y}}_1 = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} G'(\mathbf{x}_1, \mathbf{y}).$$

If $G'(\mathbf{x}_1, \mathbf{y}_1) < G'(\mathbf{x}_1, \hat{\mathbf{y}}_1) + \varepsilon$, we say that $\hat{\mathbf{y}}_1$ is "close" to $\mathbf{y}_1$ in the sense that $\hat{\mathbf{y}}_1$ is a close enough approximation of $\mathbf{y}_1$. If $\hat{\mathbf{y}}_1$ is close to $\mathbf{y}_1$, we go on to the next optimization problem,

$$\hat{\mathbf{y}}_2 = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmin}} G'(\mathbf{x}_2, \mathbf{y}).$$

If $\hat{\mathbf{y}}_1$ is not close to $\mathbf{y}_1$, this means the constraint $G'(\mathbf{x}_1, \mathbf{y}_1) < G'(\mathbf{x}_1, \hat{\mathbf{y}}_1)$ in equation (3) has been violated. Therefore we must add this constraint to our reduced problem, replacing S_1 by $S_1 \cup \{\hat{\mathbf{y}}_1\}$. In order to solve

the new reduced problem we need to find a new G' that satisfies the old and new constraints. At all times the number of constraints in the reduced problem is relatively small such that it is computationally feasible to find its solution.

Whenever a prediction $\hat{\mathbf{y}}_i$ is not satisfactorily close to $\mathbf{y}_i$, we add more constraints. For instance, Figure 4 shows our problem reduction for the training example $(\mathbf{x}_1, \mathbf{y}_1)$. Note that the reduced problems lead to the constraints $G'(\mathbf{x}_1, \mathbf{y}_1) < G'(\mathbf{x}_1, \mathbf{y}^1)$, $G'(\mathbf{x}_1, \mathbf{y}_1) < G'(\mathbf{x}_1, \mathbf{y}^7)$, $G'(\mathbf{x}_1, \mathbf{y}_1) < G'(\mathbf{x}_1, \mathbf{y}^{245})$, etc., where $\mathcal{Y} = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^m\}$ (in other words, $S_1 = \{\mathbf{y}^1, \mathbf{y}^7, \mathbf{y}^{245}\}$).

The algorithm terminates if no constraints need to be added. That is, each prediction is a good approximation,

$$\forall i : G'(\mathbf{x}_i, \mathbf{y}_i) < G'(\mathbf{x}_i, \hat{\mathbf{y}}_i) + \varepsilon \text{ where } \hat{\mathbf{y}}_i = \operatorname*{argmin}_{\mathbf{y} \in \mathcal{Y}} G'(\mathbf{x}_i, \mathbf{y}). \tag{5}$$

This is equivalent to

$$\forall i, \forall y \in \mathcal{Y} \setminus \{\mathbf{y}_i\} : G'(\mathbf{x}_i, \mathbf{y}_i) < G'(\mathbf{x}_i, \mathbf{y}) + \varepsilon. \tag{6}$$

This is similar to the full set of constraints on G in equation (3), except that G' need only satisfy each inequality within a distance of ε.

Linear Cost Function

One important assumption we make is that the family of free energy functions $\mathcal{G}$ is linear. That is, the total free energy of the protein is a sum of elementary interactions. This simplification agrees with many mathematical models of the energy force fields that control protein folding. For example, electrostatic, Van der Waals, stretch, bend, and torsion forces can all be described by the sum of energy terms for each pair of molecular elements. Given this, we can formally define the family of functions $\mathcal{G}$ to be

$$\mathcal{G} = \{G_{\mathbf{w}} : (\mathbf{x}, \mathbf{y}) \longrightarrow \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle : \text{ for some } \mathbf{w}\}. \tag{7}$$

Here the feature function Ψ is fixed and known, representing the specific energy characteristics that we are interested in. For example, one element of the vector $\Psi(\mathbf{x}, \mathbf{y})$ might be the number of proline residues from sequence $\mathbf{x}$ that appear within an alpha helix in candidate structure $\mathbf{y}$. Additional details on our design of Ψ appears later. By definition of a linear function, the dot product of the vector $\mathbf{w}$ (notated by $\langle , \rangle$) can then be taken to appropriately weight the importance of individual terms within Ψ. With this assumption, the reduced problem's constraints given by equation (4) can be rewritten as

$$\forall i, \forall y \in S_i : G_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}_i) < G_{\mathbf{w}}(\mathbf{x}_i, \mathbf{y}). \tag{8}$$

In order to solve the reduced problem, we need to find the unknown weight vector $\mathbf{w}$ such that these constraints are satisfied. Again, since $G_{\mathbf{w}}$ is a linear function, this set of constraints can translate into

$$\forall i, \forall \mathbf{y} \in S_i : \langle \mathbf{w}, \Delta\Psi_i(\mathbf{y}) \rangle > 0, \tag{9}$$

where $\Delta\Psi_i(\mathbf{y}) = \Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}_i)$. This reformulation of the constraints allows this problem to be solved in a much more elegant and computationally efficient manner. We use the powerful technique of Support Vector Machines to quickly determine the function $G_{\mathbf{w}}$, although many other techniques are possible.

Iteratively Constraining Support Vector Machines

Support Vector Machines (SVMs) are a fast and effective tool for generating functions from a set of labeled input training data. SVMs are able to determine a set of weights $\mathbf{w}$ for the function $G_{\mathbf{w}}$ that will allow $G_{\mathbf{w}}$ to accurately map all of the training example inputs $\mathbf{x}_i$ to outputs $\mathbf{y}_i$. This problem can be formulated as a quadratic program, in which the variables are the weights $\mathbf{w}$ and a set of "slack variables" ξ_i:

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{10a}$$

under the constraints

$$\forall i, \forall \mathbf{y} \in S_i : \langle \mathbf{w}, \Delta\Psi_i(\mathbf{y}) \rangle \geq 1 - \xi_i \quad \text{with} \quad \forall i : \xi_i \geq 0. \tag{10b}$$

The only differences between these constraints and those in equation (9) is that *(i)* the strict inequality (> 0) is replaced by a non-strict inequality (≥ 1), and *(ii)* slack variables ξ_i are introduced to allow a best-fit solution in the event of unsatisfiable constraints. The objective function minimizes the length of the weight vector (to normalize the constraints across various dimensions of $\mathbf{w}$) and the size of the slack variables. The constant parameter C indicates how much a solution is penalized for violating a constraint. In practice, SVMs solve the dual of the minimization problem.

We can therefore use SVMs to determine our function $G_{\mathbf{w}}$; however, this only solves half of our problem. Given a candidate $G_{\mathbf{w}}$ we must then determine if equation (3) has been violated and add more constraints to it if necessary. To accomplish this task, we build off of work done by Tsochantaridis et al. [38] which tightly couples this constraint verification problem with the SVM $\mathbf{w}$ minimization problem.

First a loss function $\Delta(\mathbf{y}_i, \mathbf{y})$ is defined that weighs the goodness of the structures $\hat{\mathbf{y}}_i$. Smaller values of $\Delta(\mathbf{y}_i, \mathbf{y})$ indicate that structures $\mathbf{y}_i$ and $\mathbf{y}$ are more similar. Adding this to the SVM constraints in equation (10b) gives

$$\forall i, \forall \mathbf{y} \in S_i : \xi_i \geq \Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \Delta\Psi_i(\mathbf{y}) \rangle. \tag{11}$$

Using this we can decide when to add constraints to our reduced problem and which constraints to add. Since at every iteration of the algorithm we determine some $\mathbf{w}$ for the current S_i, we can then find the value $\hat{\xi}_i$ assigned to variable ξ_i as a result of the optimization. $\hat{\xi}_i$ corresponds to the "worst" prediction by $\mathbf{w}$ across the structures $\mathbf{y} \in S_i$:

$$\hat{\xi}_i = \max(0, \max_{\mathbf{y} \in S_i} \Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \Delta\Psi_i(\mathbf{y}) \rangle). \tag{12}$$

This resulting $\hat{\xi}_i$, which was determined using S_i, can be compared to a similar $\hat{\xi}_i'$ that is obtained by instead maximizing over $\mathcal{Y} \setminus \{\mathbf{y}_i\}$ in equation (12). This will tell us how much the constraints we are ignoring from $\mathcal{Y} \setminus \{\mathbf{y}_i\}$ will change the solution. The constraint that is most likely to change the solution is that which would have caused the greatest change to the slack variables. Therefore we would add the constraint to S_i that corresponds to

$$\hat{\mathbf{y}}' = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_i\}} \Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \Delta\Psi_i(\mathbf{y}) \rangle. \tag{13}$$

Tsochantaridis et al. [38] show that by only adding constraints when $\hat{\mathbf{y}}'$ could change $\hat{\xi}_i$ by more than ε, one can attain a provable termination condition for the problem. The summary of this overall process appears in Algorithm 1.

Defining the Set of Valid Structures

One final issue remains to be solved to complete our algorithm. We need to specify what $\mathcal{Y}$ and $\Psi(\mathbf{x}, \mathbf{y})$ are, and how to optimize $G(\mathbf{x}, \mathbf{y})$ over $\mathcal{Y}$. In general, $\mathcal{Y}$ can be exponentially large with respect to the sequence length, making brute-force optimization impractical. Our general approach is to structure $\mathcal{Y}$ and $\Psi(\mathbf{x}, \mathbf{y})$ in a way that allows optimization of $G(\mathbf{x}, \mathbf{y})$ through dynamic programming.

Most secondary-structure prediction tools use local features to predict which regions of a protein will be helical [2]. Individual residues can have propensities for being in a helix, they can act as helix nucleation sites, or they can interact with other nearby residues. This type of information can be well captured by Hidden Markov Models (HMMs). Equivalently, we choose to capture them using Finite State Machines (FSMs). The only difference between the FSMs we use and a non-stationary HMM is that the HMM deals with probabilities, which are multiplicative, while our FSMs deal with pseudo-energies, which are additive. To a logarithm, they are the same.

We define $\mathcal{Y}$ to be the language that is recognized by some FSM. Thus a structure $\mathbf{y} \in \mathcal{Y}$ will be a string over the input alphabet of the FSM. For example, that alphabet could be $\{h, c\}$, where h indicates that the

residue at that position in the string is in a helix, and c indicates that it is in a coil region. A string $\mathbf{y}$ is read by an FSM one character at a time, inducing a specific set of transitions between internal states. Note that the FSMs we are considering do not need to be deterministic. However, they do need to satisfy the property that, for a given input string, there is at most one set of transitions leading from the initial state to a final state. We denote this sequence of transitions by $\sigma(\mathbf{y})$ and note that $\sigma(\mathbf{y})$ need not be defined for all $\mathbf{y}$.

To define $\Psi(\mathbf{x}, \mathbf{y})$, we create a helper function $\psi(\mathbf{x}, t, i)$ which assigns a vector of feature values whenever transition t is taken at position i in the sequence $\mathbf{x}$. For example, if a transition is taken to start a helix at position i, then $\psi(\mathbf{x}, t, i)$ might return features indicating that residues at position $i-3$ to $i+3$ are associated with an N-terminal helix cap. The overall feature vector is the sum of these features across all positions in the sequence: $\Psi(\mathbf{x}, \mathbf{y}) = \sum_i \psi(\mathbf{x}, \sigma(\mathbf{y})_i, i)$.

The total cost $G(\mathbf{x}, \mathbf{y})$ follows the form of equation (7). We also specify an infinite cost for structures that are the wrong length or are rejected by the FSM:

$$G(\mathbf{x}, \mathbf{y}) = \begin{cases} +\infty & \text{if } |\mathbf{x}| \neq |\mathbf{y}| \text{ or } \sigma(\mathbf{y}) \text{ is undefined} \\ \langle w, \Psi(\mathbf{x}, \mathbf{y}) \rangle & \text{otherwise} \end{cases} \tag{14}$$

This cost is easy to optimize over $\mathcal{Y}$ by using the Viterbi algorithm. This algorithm proceeds in $|\mathbf{x}|$ rounds. In round i, the best path of length s starting from an initial state is calculated for each FSM state. These paths are computed by extending the best paths from the previous round by one transition, and picking the best resulting path for each state. The algorithmic complexity is $O(|FSM| \cdot |\mathbf{x}|)$, where $|FSM|$ is the number of states and transitions in the FSM.

Implementation of the Predictor

In our experiments, we have used an extremely simple finite state machine that is presented in Figure 1. Each state corresponds to being in a helix or coil region, and indicates how far into the region we are. States `H4` and `C3` correspond to helices and coils more than 4 and 3 residues long, respectively. Short coils are permitted, but helices shorter than 4 residues are not allowed, as even 3_{10} helices need at least 4 residues to complete one turn and form the first hydrogen bond.

Table 3 lists the basic features that were used in our experiments. These features can also be considered to be the parameters of our system, as our learning algorithm assigns an appropriate weight to each one. Our choice of features is motivated by observations that amino acids have varying propensities for appearing within an alpha helix as well as for appearing at the ends of a helix, an area termed the helix cap [39]. We

introduce a single feature per residue to account for helix propensity, for a total of 20 parameters. For helix capping, we use a separate feature for each residue that appears at a given offset (-3 to $+3$) from a given end of the helix (N-terminal or C-terminal). This accounts for $20 * 7 * 2 = 280$ parameters. Finally, we also introduce a feature for very short (2-residue) and short (3-residue) coils. Thus, there are a total of 302 parameters.

Table 4 illustrates how features are associated with the transitions of the FSM. This table corresponds to the ψ function described earlier; given an FSM transition and a position in the input sequence, it outputs a set of representative features. Most of this mapping is straightforward. In the case of helix caps (labels #1 and #2), features are emitted across a 7-residue window that is centered at position $n - 1$ (the previously processed residue).

None of the features we have used involve more than one residue in the sequence. We have experimented with more complicated cost functions that model pairwise interactions between nearby residues in a helix, namely between n and $n + 3$ or n and $n + 4$. So far we have not managed to improve our prediction accuracy using these interactions, possibly because each pairwise interaction adds 400 features to the cost function, leaving much room for over-learning. Indeed, with the expanded cost functions we observed improved predictions on the training proteins, but decreased performance on the test proteins.

We have also experimented with various loss functions Δ. We have tried a 0-1 loss function (0 unless both structures are identical), hamming distance (number of incorrectly predicted residues), and a modified hamming distance (residues are given more weight when they are farther from the helix-coil transitions). Each one gives results slightly better than the previous one.

Competing Interests

The authors declare that they have no competing interests.

Authors contributions

BG participated in the design of the method, implemented the bulk of the software, and drafted much of the manuscript. CWO participated in the design of the method, helped to implement the software, and drafted much of the manuscript. WT participated in the design of the method, helped to implement the software, and drafted much of the manuscript. AL refined the HMM cost models used for prediction. MvD participated in the design of the method and helped to draft the manuscript. SD initiated the project, participated in the design of the method, coordinated, and helped to draft the manuscript. All authors

read and approved the final manuscript.

Acknowledgements

We thank Chris Batten, Edward Suh and Rodric Rabbah for their early contributions to this work. WT also thanks Saman Amarasinghe for supporting his part in this research.

References

1. Eyrich et al V: **EVA: Continuous automatic evaluation of protein structure prediction servers**. *Bioinformatics* 2001, **17**(12):1242–1243.
2. Rost B: **Review: Protein Secondary Structure Prediction Continues to Rise**. *Journal of Structural Biology* 2001, **134**(2):204–218.
3. Zemla A, Česlovas Venclovas, Fidelis K, Rost B: **A Modified Definition of Sov, a Segment-Based Measure for Protein Secondary Structure Prediction Assessment**. *Proteins* 1999, **34**(2):220–223.
4. Jones DT: **Protein Secondary Structure Prediction Based on Position-specific Scoring Matrices**. *Journal of Molecular Biology* 1999, **292**:195–202.
5. Nguyen MN, Rajapakse JC: **Prediction of protein secondary structure using bayesian method and support vector machines**. In *ICONIP* 2002.
6. King RD, Sternberg MJ: **Identification and application of the concepts important for accurate and reliable protein secondary structure prediction**. *Protein science* 1996, **5**:2298–2310.
7. Garnier J, Osguthorpe D, Robson B: **Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins**. *Journal of Molecular Biology* 1978, **120**:97–120.
8. Levin J, Pascarella S, Argos P, Garnier J: **Quantification of secondary structure prediction improvement using multiple alignments**. *Protein Engineering* 1993, **6**:849–854.
9. Rost B, Sander C: **Prediction of protein secondary structure at better than 70% accuracy**. *Journal of Molecular Biology* 1993, **232**:584–599.
10. Riis S, Krogh A: **Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments**. *Journal of Computational Biology* 1996, **3**:163–183.
11. Altun Y, Tsochantaridis I, Hofmann T: **Hidden Markov Support Vector Machines**. In *ICML '03: Proceedings of the* 20^{th} *International Conference on Machine Learning* 2003.
12. Tsochantaridis I, Altun Y, Hoffman T: **A crossover between SVMs and HMMs for protein structure prediction**. In *NIPS Workshop on Machine Learning Techniques for Bioinformatics* 2002.
13. Nguyen MN, Rajapakse JC: **Prediction of protein secondary structure using bayesian method and support vector machines**. In *9th International Conference on Neural Information Processing* 2002.
14. Bystroff C, Thorsson V, Baker D: **HMMSTR: a Hidden Markov Model for Local Sequence-Structure Correlations in Proteins**. *Journal of Molecular Biology* 2000, **301**.
15. Martin J, Gibrat JF, Rodolphe F: **Hidden Markov Model for protein secondary structure**. In *International Symposium on Applied Stochastic Models and Data Analysis* 2005.
16. Won K, Hamelryck T, Prügel-Bennett A, Krogh A: **Evolving Hidden Markov Models for Protein Secondary Structure Prediction**. In *Proceedings of IEEE Congress on Evolutionary Computation* 2005:33–40.
17. Won KJ, Prügel-Bennett A, Krogh A: **Training HMM Structure with Genetic Algorithm for Biological Sequence Analysis**. *Bioinformatics* 2004, **20**(18):3613–3627.
18. Schmidler SC, Liu JS, Brutlag DL: **Bayesian Segmentation of Protein Secondary Structure**. *Journal of Computational Biology* 2000, **7**(1/2):233–248.

19. Schmidler SC, Liu JS, Brutlag DL: **Bayesian Protein Structure Prediction**. *Case Studies in Bayesian Statistics* 2001, **5**:363–378.

20. Chu W, Ghahramani Z: **Protein Secondary Structure Prediction Using Sigmoid Belief Networks to Parameterize Segmental Semi-Markov Models**. In *European Symposium on Artificial Neural Networks Bruges (Belgium)* 2004:81–86.

21. Krogh A, Larsson B, von Heijne G, Sonnhammer E: **Predicting transmembrane protein topology with a hidden markov model: application to complete genomes**. *Journal of Molecular Biology* 2001, **305**:567–580.

22. Qian N, Sejnowski T: **Predicting the secondary structure of globular proteins using neural network models**. *Journal of Molecular Biology* 1988, **202**(4):865–884.

23. Baldi P, Brunak S, Frasconi P, Soda G, Pollastri G: **Exploiting the past and the future in protein secondary structure prediction**. *Bioinformatics* 1999, **15**.

24. Pollastri G, McLysaght A: **Porter: a new, accurate server for protein secondary structure prediction**. *Bioinformatics* 2005, **21**(8):1719–1720.

25. Petersen TN, Lundegaard C, Nielsen M, Bohr H, Bohr J, Brunak S, Gippert GP, Lund O: **Prediction of Protein Secondary Structure at 80% Accuracy**. *PROTEINS: Structure, Function, and Genetics* 2000, **14**:17–20.

26. Lin K, Simossis VA, Taylor WR, Heringa J: **A simple and fast secondary structure prediction method using hidden neural networks**. *Bioinformatics* 2005, **21**(2):152–159.

27. Ouali M, King RD: **Cascaded multiple classifiers for secondary structure prediction**. *Protein Science* 2000, **9**:1162–1176.

28. Eyrich V, Marti-Renom M, Przybylski D, Madhusudhan M, Fiser A, Pazos F, Valencia A, Sali A, Rost B: **EVA: Continuous automatic evaluation of protein structure prediction servers**. *Bioinformatics* 2001, **17**(12):1242–1243.

29. Nguyen MN, Rajapakse JC: **Multi-Class Support Vector Machines for Protein Secondary Structure Prediction**. *Genome Informatics* 2003, **14**:218–227.

30. Ward J, McGuffin L, Buxton B, Jones D: **Secondary structure prediction with support vector machines**. *Bioinformatics* 2003, **19**(13):1650–1655.

31. Ceroni A, Frasconi P, Passerini A, Vullo A: **A Combination of Support Vector Machines and Bidirectional Recurrent Neural Networks for Protein Secondary Structure Predict**. In *Advances in Artificial Intelligence, 8th Congress of the Italian Association for Artificial Intelligence, Volume 2829* 2003:142–153.

32. Casborn J: **Protein Secondary Structure Class Prediction with Support Vector Machines**. *MSc Dissertation, University of Sussex* 2002.

33. Hua S, Sun Z: **A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approach**. *Journal of Molecular Biology* 2001, **308**:397–407.

34. Hu HJ, Pan Y, Harrison R, Tai PC: **Improved Protein Secondary Structure Prediction Using Support Vector Machine With a New Encoding Scheme and an Advanced Tertiary Classifier**. *IEEE Transactions on Nanobioscience* 2004, **3**(4):265–271.

35. **EVA Largest sequence of unique subset of PDB**. *http://salilab.org/~eva/res/weeks.html#unique*.

36. Kabsch W, Sander C: **Dictionary of protein secondary structure**. *Biopolymers* 1983, **22**.

37. Berman H, Westbrook J, Feng Z, Gilliland G, Bhat T, Weissig H, Shindyalov I, Bourne P: **The Protein Data Bank**. *Nucleic Acids Research* 2000, **28**.

38. Tsochantaridis I, Hofmann T, Joachims T, Altun Y: **Support Vector Machine Learning for Interdependent and Structured Output Spaces**. In *ICML* 2004.

39. Aurora R, Rose G: **Helix capping**. *Protein Science* 1998, **7**.

Figures

Figure 1 - Predictor finite state machine

The finite state machine we used. Double circles represent accept states. The arrow leading into state `C3` indicates that it is an initial state. Each transition is labeled with the type of structure it corresponds to: helix (`H`) or coil (`C`), and a label (`#i`) indicating which features correspond to this transition in Table 4.

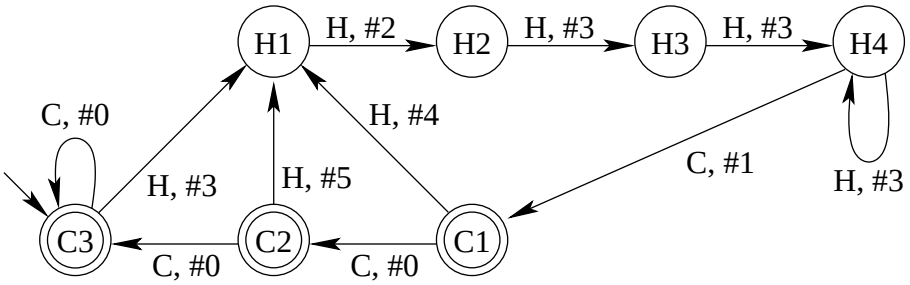

Figure 1: Predictor finite state machine

Figure 2 - Q_α accuracy histogram

Histogram showing the distribution of Q_α across proteins in the test set. We have shown the average case, and the best of the 20 runs which has the highest Q_α.

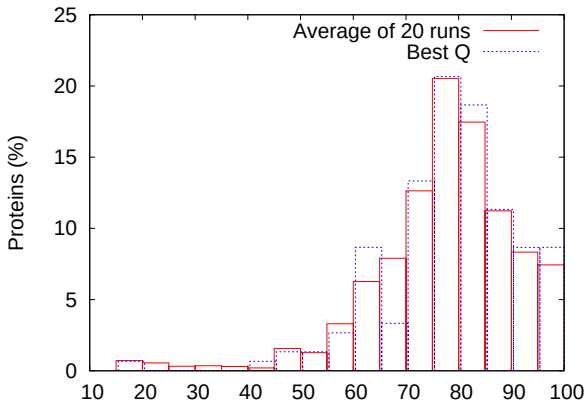

Figure 2: Q_α accuracy histogram

Figure 3 - SOV_α accuracy histogram

Histogram showing the distribution of SOV_α across proteins in the test set. We have shown the average case, and the best of the 20 runs which has the highest SOV_α.

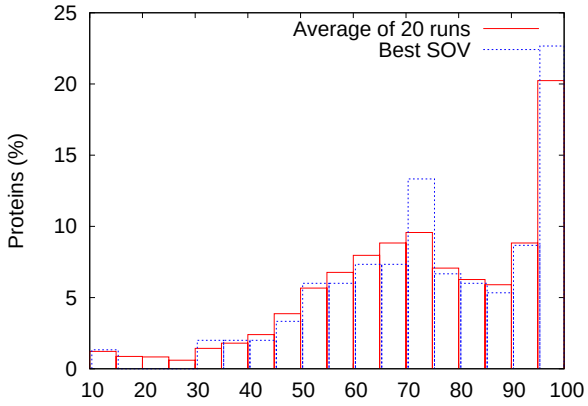

Figure 3: SOV_α accuracy histogram

Figure 4 - Summary of learning algorithm

Summary of the learning method. In this figure each large frame represents a problem that needs to be solved. On the left, we start with an intractably large problem. At each iteration, we pick a subset of the large problem to work on, solve it approximately using an SVM formulation, and use the resulting solution to expand the subset of constraints we are working with.

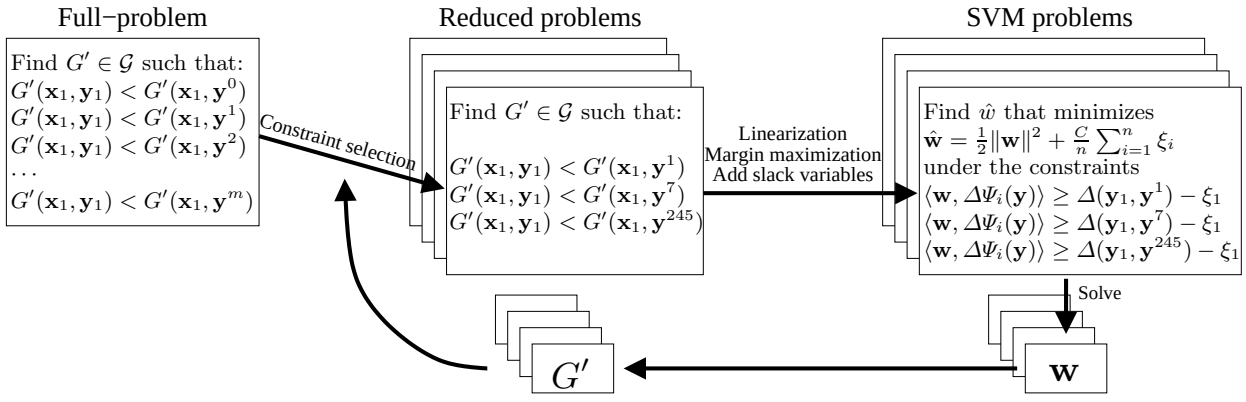

Figure 4: Summary of the learning algorithm.

Algorithms

Algorithm 1 - Algorithm for iterative constraint based optimization

Algorithm for iterative constraint based optimization.

Algorithm 1 Algorithm for iterative constraint based optimization.

Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ε
$S_i \leftarrow \emptyset$ for all $1 \leq i \leq n$
$\mathbf{w} \leftarrow$ any arbitrary value
repeat (
  for $i = 1, \ldots, n$ **do** (
    Set up the cost function:
      $H(\mathbf{y}) = \Delta(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \Delta\Psi_i(\mathbf{y}) \rangle$
    Compute $\hat{\mathbf{y}} = \mathtt{argmax}_{\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_i\}} H(\mathbf{y})$
    Compute $\hat{\xi}_i = \max\{0, \mathtt{max}_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
    if $H(\hat{y}) > \hat{\xi}_i + \varepsilon$ **then** (
      $S_i \leftarrow s_i \cup \{\hat{y}\}$
      $\mathbf{w} \leftarrow$ optimize over $S = \cup_i S_i$
))) **until** no S_i changes during iteration

Tables

Table 1 - Number of Predictor Parameters

Number of parameters used for various protein structure predictors.

Number of Predictor Parameters		
Category	**Predictor**	**Number of Parameters**
Neural Net	PHD [9]	$\geq$ 10,000
Neural Net	SSPro [23]	1400-2900
Neural Net	Riis & Krogh [10]	311-600
GOR + Linear Discrimination	DSC [6]	1000
HMM	Martin et al. [15]	471
HM-SVM	this paper (alpha only)	302

Table 1: Number of predictor parameters.

Table 2 - SOV_α and Q_α Results

Performance of our algorithm on all-alpha protein structure prediction.

SOV_α and Q_α Results					
Description	SOV_α (%) (train)	SOV_α (%) (test)	Q_α (%) (train)	Q_α (%) (test)	Training time (s)
Best run for SOV_α	76.4	75.1	79.6	78.6	123
Average of 20 runs	75.1	73.4	79.1	77.6	162
Standard deviation of 20 runs	1.0	1.4	0.6	0.9	30

Table 2: SOV_α and Q_α results.

Table 3 - Features considered by predictor

Summary of basic features that are considered. Each of these features corresponds to a parameter that is learned by our algorithm.

Features considered by predictor		
Name	Number of features	Description
A	1	Penalty for very short coil
B	1	Penalty for short coil
H_R	20	Energy of residue R in a helix
C_R^i	140	Energy of residue R at position i relative to C-cap
N_R^i	140	Energy of residue R at position i relative to N-cap
Total	302	

Table 3: Features considered by predictor.

Table 4 - Sets of features emitted by FSM transition

Sets of features that are emitted by transitions in the FSM. R_i denotes the residue at position i in the protein, and n is the current position of the FSM.

Sets of features emitted by FSM transition		
Label	Features	Description
#0	0	Coil defined as zero-energy
#1	$\sum_{i=-3}^{+3} C_{R_{n+i-1}}^{i-1}$	End of helix processing (C-cap)
#2	$H_{R_n} + \sum_{i=-3}^{+3} N_{R_{n+i-1}}^{i-1}$	Start of helix processing (N-cap)
#3	H_{R_n}	Normal helix residue
#4	$H_{R_n} + A$	Helix after very short coil
#5	$H_{R_n} + B$	Helix after short coil

Table 4: Sets of features emitted by FSM transition.