Secondary Structure Prediction of All-Helical Proteins Using Hidden Markov Support Vector Machines

B. Gassend, C. W. O'Donnell, W. Thies, A. Lee, M. van Dijk and S. Devadas
Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology
Cambridge, MA 02139
Contact email: gassend@mit.edu

Abstract

Our goal is to develop a state-of-the-art predictor with an intuitive and biophysically-motivated energy model through the use of Hidden Markov Support Vector Machines (HM-SVMs), a recent innovation in the field of machine learning. We focus on the prediction of alpha helices in proteins and show that using HM-SVMs, a simple 7-state HMM with 302 parameters can achieve a Q_α value of 77.6% and a SOV_α value of 73.4%. We briefly describe how our method can be generalized to predicting beta strands and sheets.

1 Introduction

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 [6, 17, 22]. To date, there has been a broad array of approaches to secondary structure prediction, including statistical techniques [10, 11, 16], neural networks [2, 15, 20, 25, 27, 28, 29, 32], Hidden Markov Models [3, 7, 18, 21, 23, 35, 36, 37, 40, 41], Support Vector Machines [4, 5, 13, 12, 24, 39], nearest neighbor methods [34] and energy minimization [17]. In terms of prediction accuracy, neural networks are among the most popular methods in use today [9, 31], delivering a pointwise prediction accuracy (Q_3) of about 77% and a segment overlap measure (SOV) of about 74% [15].

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 upwards of 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, over the past 15 years, the largest improvements in neural network prediction accuracy have been due to the integration of homologous sequence alignments [32, 15] rather than specific changes to the underlying cost model.

Of the approaches developed to date, Hidden Markov Models (HMMs) offer perhaps the most natural representation of protein secondary structure. An HMM consists of a finite set of states with learned transition probabilities between states. In biological terms, each transition corresponds to a local folding event, with the most likely sequence of states corresponding to the lowest-energy protein structure. HMMs generally contain hundreds of parameters, two orders of magnitude less than that of neural networks. In addition to providing a tractable model that can be reasoned about, the reduction in parameters lessens the risk of overlearning. However, the leading HMM methods to date [3, 40] have not exceeded a Q_3 value of 75%, and SOV scores are often unreported.

In this paper, we focus on improving the prediction accuracy of HMM-based methods, thereby advancing the goal of achieving a state-of-the-art predictor while maintaining an intuitive and biophysically-motivated cost model. Our technique relies on Hidden Markov SVMs (HM-SVMs), a recent innovation in the field of machine learning [1]. While HM-SVMs share the prediction structure of HMMs, the learning algorithm is more powerful. Unlike the expectation-maximization algorithms typically used to train HMMs, training with an SVM allows for a discriminative learning function, a soft margin criterion, and bi-directional influence of features on parameters [1].

Using the HM-SVM approach, we develop a simple 7-state HMM for predicting alpha helices and coils. The HMM contains 302 parameters, representing the energetic benefit for each residue being in the middle of a helix or being in a specific position relative to the N- or C-cap. Our technique does not depend on any homologous sequence alignments. Applied to a database of all-alpha proteins, our predictor achieves a Q_α value of 77.6% and an SOV_α score of 73.4%. Among other HMMs that do not utilize alignment information, it appears that our Q_α represents a 3.5% improvement over the previous best [23], 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.

2 Related Work

King and Sternberg share our goal of identifying a small and intuitive set of parameters in the design of the DSC predictor [16]. DSC is largely based on the classic GOR technique [11], 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 [19, 32, 30]. 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 [1] 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 [37]. The technique may be similar to ours, as we are reusing 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 [23]. 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 [32]) 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 [3]. 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) [21]. 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 [40, 41]. 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 [35, 36]. 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

Category	Predictor	Number of Parameters
Neural Net	PHD [32]	$\geq$ 10,000
Neural Net	SSPro [2]	1400-2900
Neural Net	Riis & Krogh [30]	311-600
GOR + Linear Discrimination	DSC [16]	1000
HMM	Martin et al. [21]	471
HM-SVM	this paper (alpha only)	302

over Schmidler et al. [7].

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 [18].

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 [29]. Rost and Sander pioneered the automatic use of multiple sequence alignments to improve the accuracy as part of their PHD predictor [32], which was the top performer at CASP2. More recently, Jones employed the PSI-BLAST tool to efficiently perform the alignments, boosting his PSIPred predictor [15] to the top of CASP3. Baldi and colleagues employ bidirectional recurrent networks in SSPro [2], a system that provided the foundation for Pollastri and McLysaght's Porter server [28]. 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 [27]. A neural network has been followed by an HMM, resulting in a simple and fast system [20]; neural networks have also been used as a post-processing step for GOR predictors [25].

The PSIPred predictor [15] 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 [9], 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 2). 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 [24, 39, 5, 4, 13, 12]. 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 [39]. 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 [24].

3 Algorithm

3.1 Formulation as an Optimization Problem

According to thermodynamics, a folded protein is in a state of minimum free-energy (except when kinetic reasons get the protein stuck in a local minimum). We therefore approach the protein structure problem as an optimization problem. We want to find a free-energy function $G(\mathbf{x}, \mathbf{y})$, which is a function of $\mathbf{x}$, the protein's amino-acid sequence and $\mathbf{y}$, the protein's secondary structure. To predict a protein's structure $\hat{\mathbf{y}}$, we perform the following minimization:

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

To go from this general statement to a working algorithm, we need to 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, 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 the 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 [26] 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:

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

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 3.5 we will be more specific about what $\mathcal{G}$ and $\mathcal{Y}$ are.

3.2 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 $G' \in \mathcal{G}$. Notice that, we start with no constraints, therefore, any function $G' \in \mathcal{G}$ satisfies equation (4). We need to check whether G' approximates the solution G to the set of (2). In particular, we verify whether G' can be used to approximate $\mathbf{y}_1$ as the solution $\hat{\mathbf{y}}_1$ of the optimization problem

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

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,

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

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; we replace 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 1 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 shows into what extend the function G' satisfies the full set of constraints in equation (3).

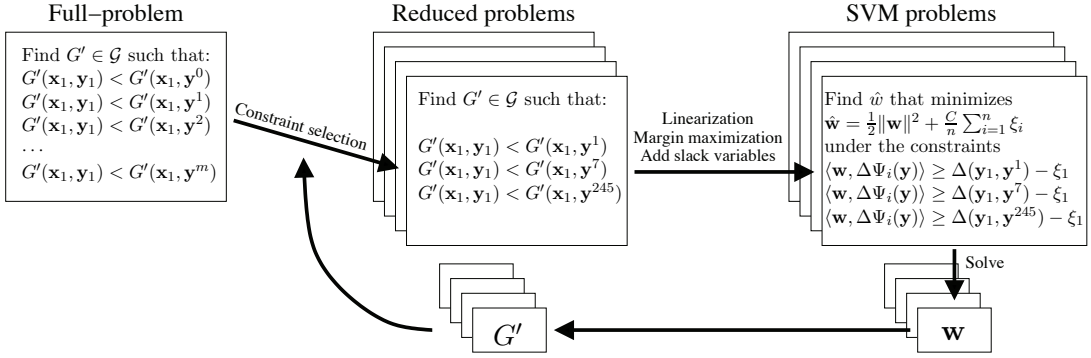


Figure 1: Summary of the learning method. In this figure each large frame represents a problem that needs to be solved. On the left, we start with an intractably large problem. At each iteration, we pick a subset of the large problem to work on, solve it approximately using an SVM formulation, and use the resulting solution to expand the subset of constraints we are working with.

3.3 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 all are 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 some specific energy characteristic that we are interested in. 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. In our method we use the powerful technique of support vector machines to quickly determine the function $G_{\mathbf{w}}$, although many other techniques are possible.

3.4 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 $\mathbf{y}_i$. They do this by solving the dual of the minimization problem

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

under the constraints

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

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 $\hat{\mathbf{y}}_i$. 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 smallest possible SVM "slack variable" values for ξ_i in equation (10a). This minimum $\hat{\xi}_i$ will be

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

This minimum $\hat{\xi}_i$, 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

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

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

```
Input: (x_1, y_1), ..., (x_n, y_n), C, ε
S_i ← ∅ for all 1 ≤ i ≤ n
repeat (
 for i = 1, ..., n do (
  Set up the cost function H(y) = Δ(y_i, y) − ⟨w, ΔΨ_i(y)⟩
  Compute ŷ = argmax_{y∈Y} H(y)
  Compute ξ_i = max{0, max_{y∈S_i} H(y)}
  if H(ŷ) > ξ_i + ε then (
   S_i ← s_i ∪ {ŷ}
   w ← optimize over S = ∪_i S_i
))) until no S_i has changed during iteration
```

Algorithm 1: Algorithm for iterative constraint based optimization.

3.5 Defining the Set of Valid Structures

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

Most secondary-structure prediction tools use local features to predict which regions of a protein will be helical [31]. 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, 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 $G(\mathbf{x}, \mathbf{y})$, we create the cost function $\psi(\mathbf{x}, t, i)$ which assigns a vector of feature values whenever a transition t is taken at position i in the sequence $\mathbf{x}$. These feature values determine the total cost $G(\mathbf{x}, \mathbf{y})$ by

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

This cost is easy to optimize over $\mathcal{Y}$ by using the Viterbi algorithm [33]. 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 FSM state. The complexity of the algorithm is $O(|FSM| \cdot |\mathbf{x}|)$, where $|FSM|$ is the number of states and transitions in the FSM.

4 Results

We now present results from our implementation of our algorithm. It was written in Objective Caml, and uses SVM^{struct}/SVM^{light} [14] by Thorsten Joachims.

4.1 Finite State Machine Definition

In our experimentation, we have used an extremely simple finite state machine that is presented in Figure 2. 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 the dataset we used did not contain any helices less than 4 residues long.

The features that were used in our experiments are presented in Table 1. The exact way in which they are associated with transitions in the FSM is indicated in Table 2.

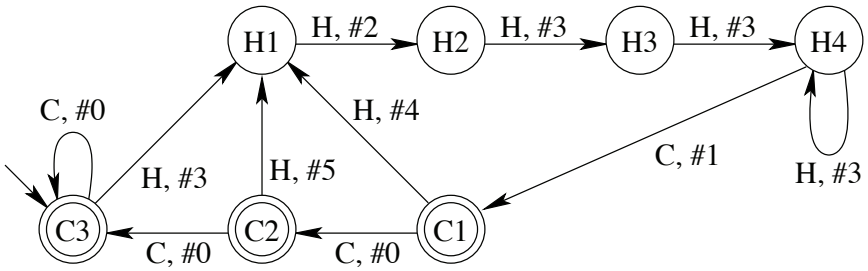


Figure 2: The finite state machine we used. Double circles represent accept states. The arrow leading into state `C3` indicates that it is an initial state. Each transition is labeled with the type of structure it corresponds to: helix (`H`) or coil (`C`), and a feature label (`#i`) indicating which features correspond to this transition in Table 2.

Name	Number of features	Comment
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 1: Summary of features that are considered.

Label	Features	Comment
#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 2: Features that arise from each transition in the FSM. R_i denotes the residue at position i in the protein, and n is the position at which we are in the protein.

We have experimented with various loss functions Δ (see Section 3.4). We have tried a 0-1 loss functions (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 tranitions). Each one gives results slightly better than the previous one.

None of the features we have used involve more than one residue in the sequence. We have done some experimentation with more complicated cost functions in which pairwise interactions betweens 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.

4.2 Results

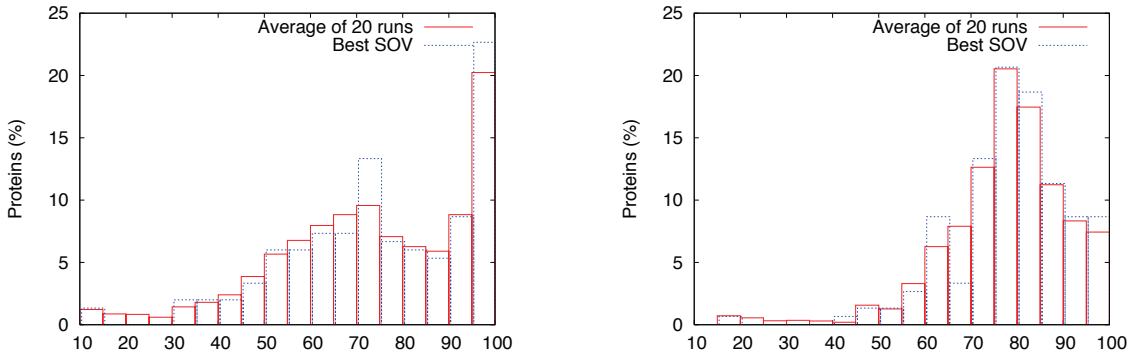
We have been working with a set of 300 non-homologous all-alpha proteins taken from EVA's largest sequence-unique subset of the PDB [8] at the end of July 2005. The sequences and structures have been extracted from PDB data processed by DSSP. Only alpha helices have been considered (`H` residues in DSSP files); everything else has been lumped as *coil* regions.

In our experimentation, we have been splitting our 300 proteins into two 150 protein subsets. The first set is used to train the cost function; the second set is used to evaluate the cost function once it has been learned. Since the results vary a bit depending on how the proteins are split in two sets, we have trained the cost function on 20 random partitions into training and test sets, and taken averages.

We present results using both the Q_α and SOV_α metrics. The Q_α metric is simply the number of incorrectly 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 [42].

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
Stardard deviation of 20 runs	1.0	1.4	0.6	0.9	30

Table 3: Results of our predictor. We have provided an average case.



(a) SOV_α (b) Q_α

Figure 3: Histograms showing the distribution of Q_α and SOV_α across proteins in the test set. We have shown the average case, and the best case which has the highest SOV_α.

Our results have been obtained for a slack variable weighting factor $C = 0.08$ in equation (10a). The algorithm termination criterion was for $\epsilon = 0.1$. Both of these parameters have a big impact on prediction accuracy and training time.

5 Conclusion

In this paper we have present a method to predict alpha helices in all-alpha proteins. The HMM is trained using a support vector machine method which iteratively picks a cost function based on a set of constraints, and uses the predictions resulting from this cost function to generate new constraints for the next iteration.

On average, our method is able to predict all-alpha helices with an accuracy of 73.4% (SOV_α) or 77.6% (Q_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 [32]. In addition, recent techniques such as PSIPred [15] consider 3-10 helices (the DSSP state 'G') to be part of a helix rather than loop, and report gains of about 2% in overall Q_3 if helices are restricted to 4-helices (as in most HMM techniques, including ours).

The real power of the machine learning method we use is its applicability beyond HMM models. Indeed, instead of describing a protein structure as a sequence of HMM states, we could equally describe it as a parse tree of a context-free grammar or multi-tape grammar. 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 they are forming hydrogen bonds with in adjacent sheets.

A Example learned weight vector

Tables 4 and 5 show the w vector that led to the best test SOV.

A	-86
B	-43

Table 4: Residue independent pseudo-energies.

	H_R	N_R^{-3}	N_R^{-2}	N_R^{-1}	N_R^0	N_R^1	N_R^2	N_R^3	C_R^{-3}	C_R^{-2}	C_R^{-1}	C_R^0	C_R^1	C_R^2	C_R^3
G	-1731	443	26	-73	250	150	-179	-319	-277	1	123	369	-833	215	187
A	764	534	484	800	-745	-628	-471	-528	-357	-386	-452	-499	41	580	336
V	997	512	603	727	-824	-794	-583	-311	-340	-588	-667	-879	-68	706	501
I	1683	611	540	858	-1364	-1202	-1001	-425	-388	-591	-815	-990	380	822	381
L	1440	756	879	989	-1143	-1057	-743	-394	-392	-447	-614	-826	450	948	669
F	734	653	559	686	-750	-592	-551	-332	-283	-478	-718	-601	30	581	433
P	-4024	376	-110	-232	2325	1479	601	-178	-132	169	283	0	-2343	-607	-327
M	645	623	554	736	-930	-750	-300	-309	-349	-340	-450	-511	141	778	615
W	769	550	558	864	-551	-435	-356	-184	-255	-488	-647	-762	-265	-29	236
C	-1507	56	-253	-262	50	-204	-276	-292	21	308	296	482	-844	113	-195
S	-769	575	383	547	85	55	-125	-314	-451	-281	-167	35	-573	448	304
T	-14	706	689	968	-235	-56	-23	-205	-522	-679	-489	-434	-592	425	248
N	-917	498	235	463	-140	-194	-461	-454	-242	-114	65	231	-438	308	153
Q	556	656	445	849	-512	-533	-378	-372	-373	-399	-464	-706	-50	742	450
Y	495	435	335	457	-771	-581	-579	-448	-249	-462	-385	-433	-94	569	517
H	-664	322	106	324	-26	-68	-158	-324	-291	14	146	327	-269	473	270
D	-559	886	614	890	299	230	207	16	-499	-510	-498	-214	-725	183	208
E	296	637	522	747	-352	-186	-183	-292	-416	-379	-261	-344	-88	487	414
K	11	567	373	476	-522	-446	-414	-327	-226	-203	-164	-165	-91	548	441
R	329	323	367	642	-429	-476	-317	-297	-269	-435	-412	-369	-58	583	374

Table 5: Residue dependent pseudo-energies

References

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov Support Vector Machines. In *ICML '03: Proceedings of the* 20^{th} *International Conference on Machine Learning*, 2004.

[2] P. Baldi, S. Brunak, P. Frasconi, G. Soda, and G. Pollastri. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15:937–946, 1999.

[3] C. Bystroff, V. Thorsson, and D. Baker. HMMSTR: a hidden markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, 301:173–190, 2000.

[4] J. Casborn. Protein Secondary Structure Class Prediction with Support Vector Machines. MSc Dissertation, University of Sussex, 2002.

[5] A. Ceroni, P. Frasconi, A. Passerini, and A. Vullo. 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, pages 142–153, 2003.

[6] C. C. Chen, J. P. Singh, and R. B. Altman. Using imperfect secondary structure predictions to improve molecular structure computations. *Bioinformatics*, 15(1):53–65, 1999.

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

[8] EVA Largest sequence of unique subset of PDB. http://salilab.org/ eva/res/weeks.html#uniquej.

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

[10] G. Fasman and P. Chou. Prediction of the secondary structure of proteins from their amino acid sequence. *Adv. Enzymol.*, 47:45–148, 1978.

[11] J. Garnier, D. Osguthorpe, and B. Robson. Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of Molecular Biology*, 120(1):97–120, 1978.

[12] H.-J. Hu, Y. Pan, R. Harrison, and P. C. Tai. Improved Protein Secondary Structure Prediction Using Support Vector Machine With a New Encoding Scheme and an Advanced Tertiary Classifier. *IEEE Transactions on Nanobioscience*, 3(4):265–, 2004.

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

[14] T. Joachims. Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*, pages 169–185. MIT Press, 1998.

[15] D. T. Jones. Protein Secondary Structure Prediction Based on Position-specific Scoring Matrices. *Journal of Molecular Biology*, 292:195–202, 1999.

[16] R. D. King and M. J. Sternberg. Identification and application of the concepts important for accurate and reliable protein secondary structure prediction. *Protein science*, 5:2298–2310, 1996.

[17] J. L. Klepeis and C. A. Floudas. ASTRO-FOLD: A Combinatorial and Global Optimization Framework for Ab Initio Prediction of Three-Dimensional Structures of Proteins from the Amino Acid Sequence. *Biophysical Journal*, 85:2119–2146, 2003.

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

[19] J. Levin, S. Pascarella, P. Argos, and J. Garnier. Quantification of secondary structure prediction improvement using multiple alignments. *Protein Engineering*, 6:849–854, 1993.

[20] K. Lin, V. A. Simossis, W. R. Taylor, and J. Heringa. A simple and fast secondary structure prediction method using hidden neural networks. *Bioinformatics*, 21(2):152–159, 2005.

[21] J. Martin, J.-F. Gibrat, and F. Rodolphe. Hidden Markov Model for protein secondary structure. In *International Symposium on Applied Stochastic Models and Data Analysis*, 2005.

[22] L. J. McGuffin and D. T. Jones. Benchmarking secondary structure prediction for fold recognition. *Proteins: Structure, Function, and Genetics*, 52(2):166–175, 2003.

[23] M. N. Nguyen and J. C. Rajapakse. Prediction of protein secondary structure using bayesian method and support vector machines. In *9th International Conference on Neural Information Processing*, volume 2, pages 616–620, 2002.

[24] M. N. Nguyen and J. C. Rajapakse. Multi-Class Support Vector Machines for Protein Secondary Structure Prediction. *Genome Informatics*, 14:218–227, 2003.

[25] M. Ouali and R. D. King. Cascaded multiple classifiers for secondary structure prediction. *Protein Science*, 9:1162–1176, 2000.

[26] The Research Collaboratory for Structural Bioinformatics PDB. http://www.rcsb.org/pdb/.

[27] T. N. Petersen, C. Lundegaard, M. Nielsen, H. Bohr, J. Bohr, S. Brunak, G. P. Gippert, and O. Lund. Prediction of Protein Secondary Structure at 80% Accuracy. *PROTEINS: Structure, Function, and Genetics*, 14:17–20, 2000.

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

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

[30] S. Riis and A. Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of Computational Biology*, 3:163–183, 1996.

[31] B. Rost. Review: Protein Secondary Structure Prediction Continues to Rise. *Journal of Structural Biology*, 134(2):204–218, 2001.

[32] B. Rost and C. Sander. Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, 232:584–599, 1993.

[33] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[34] A. A. Salamov and V. V. Solovyev. Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. *Journal of Molecular Biology*, 247:11–15, 1995.

[35] S. C. Schmidler, J. S. Liu, and D. L. Brutlag. Bayesian Segmentation of Protein Secondary Structure. *Journal of Computational Biology*, 7(1/2):233–248, 2000.

[36] S. C. Schmidler, J. S. Liu, and D. L. Brutlag. Bayesian Protein Structure Prediction. *Case Studies in Bayesian Statistics*, 5:363–378, 2001.

[37] I. Tsochantaridis, Y. Altun, and T. Hoffman. A crossover between SVMs and HMMs for protein structure prediction. In *NIPS Workshop on Machine Learning Techniques for Bioinformatics*, 2002.

[38] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *ICML '04: Proceedings of the* 21^{st} *International Conference on Machine Learning*, 2004.

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

[40] K. Won, T. Hamelryck, A. Prügel-Bennett, and A. Krogh. Evolving Hidden Markov Models for Protein Secondary Structure Prediction. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 33–40, 2005.

[41] K.-J. Won, A. Prügel-Bennett, and A. Krogh. Training HMM Structure with Genetic Algorithm for Biological Sequence Analysis. *Bioinformatics*, 20(18):3613–3627, 2004.

[42] A. Zemla, Česlovas Venclovas, K. Fidelis, and B. Rost. A Modified Definition of Sov, a Segment-Based Measure for Protein Secondary Structure Prediction Assessment. *Proteins*, 34(2):220–223, 1999.