An Iterative Method for Improved Protein Structural Motif Recognition

Bonnie Berger* Mona Singh†

Abstract

We present an iterative algorithm that uses randomness and statistical techniques to improve existing methods for recognizing protein structural motifs. Our algorithm is particularly effective in situations where large numbers of sufficiently diverse examples of the motif are not known. These are precisely the situations that pose significant difficulties for previously known methods.

We have implemented our algorithm and we demonstrate its performance on the coiled coil motif. We test our program LearnCoil on the domain of 3-stranded coiled coils and subclasses of 2-stranded coiled coils. We show empirically that for these motifs, our method overcomes the problem of limited data.

1 Introduction

One of the most important problems in computational biology is that of predicting how a protein will fold in three dimensions when we only have access to its one-dimensional amino acid sequence. Biologists are interested in this problem since the fold or structure of a protein provides the key to understanding its biological function. Unfortunately, determining the three dimensional fold of a protein is very difficult. Experimental approaches such as NMR and X-ray crystallography are expensive and time-consuming (they can take more than a year), and often do not work at all. Therefore, computational techniques that predict protein structure based on already available one-dimensional sequence data can help speed up the understanding of protein functions.

*Math Dept. and Lab. for Computer Science (LCS), MIT, bab@theory.lcs.mit.edu. Supported in part by NSF Career Award CCR-9501997.

†DIMACS and Princeton University, mona@cs.princeton.edu. Part of this work was done at Lab. for Computer Science, MIT. Supported in part by contracts from ARPA/ONR N00014-92-J-1310, ARPA N00014-92-J-1799, NSF CCR-9310888, and NSF CCR-9501997.

An important first step in tackling the protein folding problem is a solution to the *structural motif recognition problem*: given a known commonly occurring three-dimensional substructure, or *motif*, determine whether this motif occurs in a given amino acid sequence, and if so, in what positions. In this paper, we focus on a special type of α-helical motif, known as the coiled coil motif (see section 2), although the techniques presented can be applied to other motifs as well.

Most approaches to the motif recognition problem work only for motifs that are already well-studied — that is, they are known to occur in many sufficiently diverse proteins. This knowledge usually comes from biologists who have studied many examples of the motif. However, there are many motifs for which only a small subset of examples are known, and this subset is often not rich enough to be representative of the motif. Thus, for lack of data, current prediction methods ranging from straightforward sequence alignments to more complicated methods such as those based on profiles of the motifs often fail to successfully identify such motifs.

For example, in the case of the coiled coil motif, most known instances are 2-stranded coiled coils (i.e, coiled coils consisting of 2 α-helices). As a result, known prediction algorithms work well for predicting 2-stranded coiled coils [7, 6, 3, 19, 26, 28], but do not work as well for the related 3-stranded coiled coil motif (i.e., coiled coils consisting of 3 α-helices), due to the lack of known 3-stranded coiled coil sequences. That is, for 3-stranded coiled coils, these algorithms have a large amount of overlap between the scores for sequences that do not contain coiled coils and sequences that do.

In this paper, we give an iterative method to improve existing statistical methods for structural motif recognition, particularly in the case where there are not sufficiently diverse or enough examples of the motif. Our main result is a linear-time algorithm that uses information obtained from a database of sequences of a specific motif, which we refer to as the *base motif*, to make predictions about a more general motif, which we refer to as the *target motif*.

The basic theoretical framework for our approach to structural motif recognition is due to Berger [3]. We build on this framework and introduce a heuristic that is able to improve protein structural motif recognition substantially for our test set of coiled coils. Our method has the following key features:

- The algorithm iteratively scans a large database of test sequences to find sequences that are presumed to fold into the target motif. The selected sequences are then used to update the parameters of the algorithm; these updates affect the performance of the algorithm in the next iteration.
- In each iteration, the algorithm scores all the sequences based on its current estimates of the parameters and the theoretical framework developed in [3].
- In each iteration, the algorithm uses randomness to select which sequences are presumed to fold into the target motif.
- The selected sequences are used in the beginning of the next iteration to update the parameters of the algorithm in a Bayesian-like weighting scheme.

There are several ways in which our iterative algorithm is kept running in a "safe" fashion, without increasing the false positive rate by incorporating sequences into the final database that do not fold into the motif. First, we begin with a mathematically sound scoring subroutine, that experimentally has a low false positive rate. Second, our method of computing likelihoods ensures that only a certain fraction of all residues are scored as positive examples of the motif (see section 3). Finally, while evaluating our program, we run the program with test sequences that are known not to contain coiled coils, and this has helped us determine when the algorithm is performing well.

Implementation results. In order to demonstrate the efficacy of our methods, we test them on the domain of 2- and 3-stranded coiled coils (see section 4).

First, we show how to use our methods to recognize 3-stranded coiled coils given examples of 2-stranded coiled coils. In other words, starting with a base motif of 2-stranded coiled coils, we "learn" the target motif comprising of 2- and 3-stranded coiled coils. The initial predictor already has good performance on 2-stranded coiled coils, so we test our algorithm by its performance on 3-stranded coiled coils.

We evaluate our algorithm on 3-stranded coiled coils with respect to two statistical cross validation tests: the "leave one out" test and the "leave half out" test. In the first scenario, the algorithm starts with data from the 2-stranded coiled coil database, and iterates on a test set that contains sequences which are known to form 3-stranded coiled coils, sequences which are thought to form 3-stranded coiled coils, sequences for which no structural information is available, and sequences which are known *not* to contain coiled coils. The category of each sequence in this test set is *not* known to the algorithm, and the sequences which do not contain coiled coils are given to the algorithm in order to test its robustness. At the end of the procedure, the algorithm is evaluated by the number of the 3-stranded coiled coil sequences which it recognizes. Each time a sequence that is present in the database the algorithm is building is scored, it is removed from that database to avoid the possibility of unfairly biasing the test. In this scenario, we find that our algorithm greatly enhances the recognition of 3-stranded coiled coils, without affecting its performance on sequences that are known not to contain coiled coils. In particular, we are able to select 93% of the sequences that are conjectured by biologists to contain coiled coils, with no false positives out of the 286 sequences known not to contain coiled coils. Previously, the best performance without false positives is 67%.

We also test our algorithm on 3-stranded coiled coils in a much more difficult scenario. In particular, instead of cross validating our procedure by leaving out just one sequence at a time when testing, the algorithm iterates on test sequences that contain only half of the sequences known to form 3-stranded coiled coils. It is then evaluated by its performance on the 3-stranded coiled coil sequences that are not iterated upon. In this scenario, we also find improved performance. The 3-stranded coiled coil sequences are split in half 3 times, and on average, the algorithm is able to select 85% of the left out 3-stranded coiled coil sequences, with likelihood scores higher than that of the highest scoring negative sequence. On average, the previous best performance without false positives is 67%.

Finally, we test our program on subfamilies of 2-stranded coiled coils using the leave one out criterion. For 2-stranded coiled coils, we have a good data set consisting of a diverse set of sequences. However, to test our program, we simulate a limited data problem by testing our program `LearnCoil` on subfamilies of 2-stranded coiled coils. That is, one subfamily of 2-stranded coiled coils is chosen to make up the base motif, and the class of all 2-stranded coiled coils is the target motif. Here we find that we have excellent performance; i.e., we are able to completely learn the coiled coil regions in our entire 2-stranded coiled coil database starting from a database consisting of coiled coils from any one subfamily. Based on our experiments, such performance does not appear to be possible without the use of our iterative algorithm. In particular, the best performance for the non-iterative approach ranges between 70 and 88%.

Biological significance. As a consequence of this work, we have identified many new sequences that we believe contain coiled coils or coiled-coil-like structures. One of our more striking findings is the existence of one and occasionally two coiled-coil-like regions in the envelope glycoproteins of many retroviruses, including Human Immunodeficiency Virus (HIV), Simian Immunodeficiency Virus (SIV), and Human T-cell Lymphotropic Virus (HTLV). Independent experimental investigations have also predicted these coiled-coil-like regions in HIV and SIV [9, 25]. Our computational analysis indicates that the envelope glycoproteins of retroviruses can be classified into two structural groups, and a companion paper detailing our findings is forthcoming [5].

2 Further background

The coiled coil motif is found in fibrous proteins, DNA binding proteins, and in tRNA-synthetase proteins. Recently it has been proposed that the 3-stranded coiled coil motif acts as the cell fusion mechanism for many viruses, and algorithms for predicting these structures could aid in the study of how viruses invade cells. Computational methods [7, 26] have already identified such coiled coil regions in influenza virus hemagluttinin and

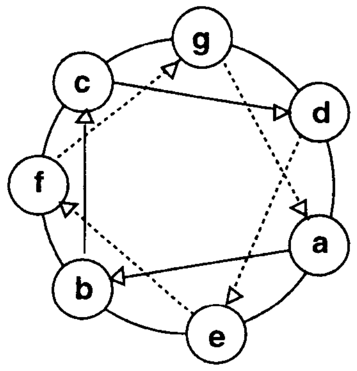

Figure 1: Top view of a single strand of a coiled coil. Each of the seven positions $\{a, b, c, d, e, f, g\}$ corresponds to the location of an amino acid residue which makes up the coiled coil. The arrows between the seven positions indicate the relative locations of adjacent residues in an amino acid subsequence. The solid arrows are between positions in the top turn of the helix, and the dashed arrows are between positions in the next turn of the helix.

Moloney murine leukemia virus envelope protein; both of these predictions have been confirmed by X-ray crystallography [13, 17].

Coiled coils are a particular type of α-helix, consisting of two or more α-helices wrapped around each other with a slight left-handed superhelical twist. Coiled coils have a cyclic repeat of seven positions, a, b, c, d, e, f, and g (see Figure 1). The seven positions are spread out along two turns of the helix. Coiled coils show a characteristic heptad repeat with hydrophobic residues found in positions a and d, and this repeat makes coiled coils particularly amenable to recognition by computational techniques.

Previous approaches to predicting coiled coils. Computational methods have been quite successful for predicting coiled coils [28, 26, 19, 3, 6, 7]. Standard approaches [28, 26] look at the frequencies of each amino acid residue in each of the seven repeated positions. Overall this *singles method* does pretty well. When the `NewCoils` program of Lupas et al. [26] is tested on the PDB (the database of all solved protein structures), it finds all sequences that contain coiled coils. On the other hand, 2/3 of the sequences it predicts to contain coiled coils do not. That is, the false positive rate for the singles method is quite high.

Recently researchers have given a linear-time algorithm for predicting coiled coils by approximating dependencies between positions in the coiled coil using pairwise frequencies [3, 6, 7]. This method uses estimates of probabilities for singles and pair positions (i.e., the probability that a particular residue is in a given heptad repeat position, and the probability that a given residue pair exists in a given pair of heptad repeat positions). For a given residue's contribution to the score, the algorithm considers residues at the structurally relevant distances $i = 1$, $i = 2$ and $i = 4$, calculating the geometric mean of the three quantities $P(k, k+i)/P(k+i)$, where $P(k, k+i)$ is the probability of finding residues k and $k+i$ distance i apart in a coiled coil, and $P(k+i)$ is the probability of finding residue $k+i$ in a coiled coil.

This method of predicting coiled coils has been very effective for predicting 2-stranded coiled coils. When tested on the PDB (the set of solved structures), the `PairCoil` algorithm based on this method selects out all sequences that contain coiled coils, and rejects all the sequences that do not contain coiled coils. Furthermore, when tested on a database of 2-stranded coiled coils (with a sequence removed from the database at the time it is scored), each amino acid residue in a coiled coil region is correctly labeled as being part of a coiled coil. As mentioned before, however, `PairCoil` does not have as good performance on 3-stranded coiled coils.

Since `PairCoil` has better performance than the singles method algorithm, particularly with respect to the false-positive rate, this is the scoring method we build on, as well as the scoring method to which we compare our results.

Related computational methods. Other types of iterative approaches have been applied to sequence alignment and protein structure prediction by researchers [30, 1, 21, 15]. Algorithmically, our approach differs from these approaches in two major ways. The first is our use of randomness to incorporate sequences into our database, and the second is our use of weighting to update the database (see section 3). In addition, several of these papers are directed toward sequence alignment, and sequence alignment is not so effective a tool for predicting coiled coils, as the various subfamilies of coiled coils do not align well to each other. Also, since the goal of these other methods is often to output potential matching alignments, the testing of these algorithms is quite different. In particular, although some of these approaches use the "leave one out" criterion, to the best of our knowledge, none of them test performance with the "leave half out" criterion.

Various machine learning techniques have been applied to the protein structure prediction problem. The two main approaches are neural nets (e.g., [22, 29, 27]) and hidden Markov models (e.g., [23, 2]). Both of these approaches require adequate data on the target motif, since there is a "training session" on sequences that are *known* to contain the target motif. Our approach differs from these methods since it does not require well analyzed data on the target motif per se. Instead it uses already available data on a base motif and generalizes it to recognize the target motif, by running on a large number of sequences, some of which are suspected to fold into the target motif.

3 The algorithm

Our algorithm is initially given a database of residues which are known to fold into the base motif. From this database, it calculates parameters which are estimates of the singles and pair probabilities for the base motif's positions. It is also given a set of sequences upon which it iterates. Our algorithm takes advantage of the fact that the target motif is a generalization of the base motif. In particular, once the algorithm has identified some of the sequences that are presumed to contain the target motif, it can modify its database (and thus its estimates of the probabilities) by "adding" these newly found sequences. (See Figure 2.) The algorithm iterates four basic steps:

1. The algorithm uses its estimates of the pair and singles probabilities to determine a likelihood function, which maps residue scores to a likelihood of the residue belonging to the target motif.

2. The algorithm scores each of the test sequences using the estimated probabilities, and calculates the likelihoods for each of these sequences.

3. The algorithm flips coins with probability proportional to the likelihood of each score to determine which parts (if any) of each sequence are presumed to be part of the target motif. The residues which are thus determined to be presumed examples of the target motif make up the new database for the next iteration.

4. The algorithm uses the base motif database and the new database just determined in this iteration to update its estimates of the singles and pair probabilities for the target motif by weighting the two databases.

The algorithm continues iterating until the new database stabilizes. Ultimately, we use the final database and the parameters estimated from this database to make better predictions about the target motif.

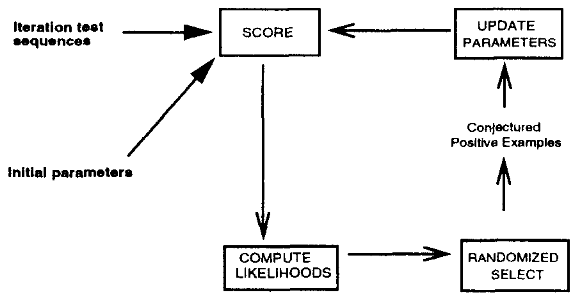

Figure 2: Our basic algorithm. Initially, the algorithm starts off with a set of initial parameters and a set of test sequences upon which it iterates. In each iteration, the algorithm selects a new database, and re-estimates its parameters.

We now describe each of the components of the algorithm in more detail, using coiled coils as an example, although the algorithm can be applied to other protein motifs.

Scoring. In our implementation, we use the PairCoil program described by Berger et al. [7] as our scoring procedure (see also section 2), although any good prediction algorithm with a low false positive rate can be used for scoring. The scoring procedure gives a *residue score* for each amino acid in a given sequence, as well as a *sequence score*, which is the maximum residue score in the sequence.

We use this scoring procedure with probabilities for the singles and pair positions for the target motif estimated using the original database of the base motif, and the database built by the algorithm in the previous iteration. In each iteration after the first, when we score a sequence we check to see if it was identified in the previous iteration. If it was, we remove this sequence from the database and adjust the probabilities before scoring.

Given good estimates for the probabilities for the singles and pair positions for the motif, and reasonable assumptions about dependencies in the motif, the PairCoil scoring method which we use as a subroutine is mathematically justified [3].

Computing likelihoods. Once we have a sequence score, we assess it by converting it into a likelihood that the sequence contains the target motif. In each iteration of the algorithm, we compute a function that takes a residue score and computes the likelihood that the residue is part of the target motif.

We compute this likelihood function in a manner described in [7]. In particular, every sequence in a large sequence database is scored. (Ideally, this large sequence database is the PIR. However, in practice, to save time, we use a sampled version of the PIR, which is 1/25-th the size; the likelihood function calculated using this sampled PIR is a good approximation to the likelihood function calculated using the entire PIR.) The sampled PIR residue score histograms are nearly Gaussian distributed with some extra probability mass added on the right-hand tail. This extra mass is attributed to residues in the target motif, since they are expected to score higher. In the case of the coiled coil motif, given the biological data currently available, it is estimated that between 1/50 and 1/30 of residues in the PIR are in a coiled coil. To fit a Gaussian to the histogram data, we calculate the mean so that the extra probability mass on the right side of the mean corresponds to between 1/50 and 1/30 of the total mass of the PIR. We then compute the standard deviation using only scores below that mean, where a Gaussian better fits the histogram data. The likelihood that a residue with a given score is a coiled coil is estimated as the ratio of the extra histogram mass above the Gaussian at that score (corresponding to data assumed to be coiled) to the total histogram mass at that score. A least square fit line is then used to approximate the likelihood function in the linear region from 10 to 90 percent. This line then gives an approximation for the likelihoods corresponding to all scores.

One feature of this method of computing likelihoods is that it does not allow too many residues to be considered as part of coiled coils. This helps keep the false positive rate of the algorithm low.

Randomized selection of the new database. Once we have obtained the likelihood function for an iteration, we wish to use the likelihoods to build a new database of sequences presumed to fold into the target motif. At the beginning of each iteration, our new database contains no sequences. Then for each sequence in the set of test sequences, we do the following. First, we score each sequence and then convert its sequence score to a likelihood. Next, we draw a number uniformly at random from the interval [0, 1]. If the number drawn is less than or equal to the likelihood of the sequence, then the sequence is added to the new database. All residues in this sequence that have scores equal to the sequence score or greater than the 50% likelihood score (which is the algorithm's cutoff for a residue being in a coiled coil) are added to the database. Once we have processed every sequence in our test set, then we have our new database of sequences presumed to fold into the target motif.

In practice, we find that adding randomness substantially improves the performance of our algorithm. In fact, if the procedure is written just to accept sequences that have greater than 50% likelihood, then the algorithm fails to recognize many sequences which are known to contain 3-stranded coiled coils. On the other hand, if the procedure lowers the threshold value for acceptance, then its false positive rate increases.

Updating parameters. Once we have a new database of sequences that are thought to contain the target motif, we need to update the parameters used by the algorithm for scoring. In our case, in each iteration of the algorithm, the scoring procedure requires updates of the estimates of probabilities for singles and pair positions. The most straightforward way to update the probabilities is to use a maximum likelihood estimate from frequency counts from the new database. However, this does not work that well in practice. Instead, we update each probability by taking a weighted average of the probability given by the base motif database and the probability given by the new database.

Our way of updating probabilities can be described in a mathematical framework. The approach we give is motivated by a Bayesian viewpoint [20]. In particular, we model the probabilities as the parameters of a Multinomial distribution, and we use the conjugate Dirichlet density to model the prior information we have about these probabilities [8]. In fact, the approach we give is not completely Bayesian, as we will use the seen data to pick the parameters of the prior distribution; this is sometimes called a Bayes/Non-Bayes compromise [20].

Suppose we want to estimate the probabilities of various amino acids in position a of a coiled coil. (We estimate the probabilities for one position of a coiled coil independent of probabilities for other positions. Here, we focus on the case of updating singles probabilities; updating pair probabilities is analogous.) There are two types of data available to us. First, we have a frequency count vector $\vec{x} = (x_1, x_2, \ldots, x_{20})$ where x_i is the number of times amino acid i appears in position a the base motif database. In addition, in each iteration of the algorithm we have a new count vector $\vec{y} = (y_1, y_2, \ldots, y_{20})$ where y_i is the number of times amino acid i appears in position a in the new database (i.e., the database consisting of the sequences we have selected in this iteration of the algorithm). It is assumed that the count vector $\vec{y}$ is generated at random according to the Multinomial distribution with parameter $\vec{p} = (p_1, p_2, \ldots, p_{20})$. These parameters are the probabilities we wish to estimate.

Our prior beliefs are modeled by the Dirichlet density. The Dirichlet density is parameterized by $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_k)$, where $\alpha_i > 0$ and $\alpha_0 = \sum \alpha_i$. It has mean $(\alpha_1/\alpha_0, \alpha_2/\alpha_0, \ldots, \alpha_k/\alpha_0)$. The larger α_0 is, the smaller the variance is. The Bayesian estimate for the probabilities $p_1, p_2, \ldots, p_{20}$ can be found by looking at the posterior distribution. It turns out that this posterior distribution is the Dirichlet distribution $\mathcal{D}(\vec{\alpha} + \vec{y})$ [8, 20]. That is, the new parameter of the distribution is the vector sum of the original parameters and the observed data. Thus, a Bayesian estimate for probability p_i after seeing the data $\vec{y}$ is:

$$\frac{\alpha_i + y_i}{\alpha_0 + y_0}, \text{ where } y_0 = \sum_{i=1}^{20} y_i.$$

In picking the parameters of the prior distribution, we depart from the traditional Bayesian approach, and choose the parameters of the prior distribution after seeing the data. In particular, since the base motif and the target motif are related, we want the base motif database to have a strong effect on the estimates for our probabilities, and thus we choose the variance of the prior distribution accordingly.

The mean of the Dirichlet density is specified by the estimated probabilities of the base motif. The variance of the density is picked as follows. If $0 < \lambda < 1$ is the effect, or weight, that we want the base motif database to have, we let $\alpha_i = x_i \cdot \frac{\lambda}{1-\lambda} \frac{y_0}{x_0}$, where $x_0 = \sum_{i=1}^{20} x_i$ and $y_0 = \sum_{i=1}^{20} y_i$. (Actually, we have to be careful in the case where $x_i = 0$.) It is easy to verify that our estimate for the probability p_i is given by $\lambda \frac{x_i}{x_0} + (1-\lambda)\frac{y_i}{y_0}$. Namely, our updated probability is a weighted average of the probability given by the base motif database and the probability given by the new database.

In practice, we have found that our method of updating probabilities has worked well. It is superior to a maximum likelihood approach which uses just the current iteration's frequency counts. These estimates of the probabilities are especially problematic in the zero frequency case. Our method also performs better than an unweighted approach using both the initial frequency counts and the current iteration's frequency counts. These estimates of the probabilities are largely dependent on the size of the original database, and the number of residues that are presumed at each iteration to be part of the target motif. In our test domain of coiled coils, we found that this method of updating probabilities missed more sequences that contain coiled coils than did our method for updating probabilities.

Using Dirichlet mixture densities as priors to estimate amino acid probabilities has been studied by Brown et al. [12]. Their approach uses as a prior the maximum likelihood estimate of a mixture Dirichlet density, based on data previously obtained from multiple alignments of various sets of sequences. Their approach is a pure Bayesian approach, and their prior distribution has a smaller effect on the final probability estimates.

Algorithm termination. The iteration process terminates when it stabilizes; that is, when the number of residues added from the previous iteration changes by less than 5%. Usually the procedure converges in around six iterations; otherwise, we terminate it after 15 iterations. In practice, we found that the algorithm rarely had to to be terminated due to lack of convergence.

In our implementation, the running time of the entire algorithm is linear in the total number of residues in all sequences which are given as input. The basic operation in each iteration is scoring every sequence using the PairCoil algorithm. For each sequence, the PairCoil scoring program takes time linear in the number of residues. Since we have at most a fixed number of iterations, the entire algorithm is linear-time.

Distinguishing the base and target motifs. After running LearnCoil, the "learned" target concept contains both 2- and 3-stranded coiled coils. The problem of distinguishing one set from the other remains. The MultiCoil program of Wolf, Kim, and Berger [unpublished results, 1996] is being developed for this purpose and in initial experiments performs well.

4 Results

We have implemented our algorithm in a C program called LearnCoil. We test our program on the domain of 3-stranded coiled coils and subclasses of 2-stranded coiled coils. First we describe the databases we use to test the program, and then we follow by describing the program's performance.

4.1 The databases and test sequences

Our original database of 2-stranded coiled coils consists of 58,217 amino acid residues which were gathered from sequences of myosin, tropomyosin, and intermediate filament proteins [7]. We also have separate databases containing sequences from each of these protein subclasses individually. A synthetic peptide of tropomyosin is the only solved structure among these.

We test LearnCoil on the 3-stranded coiled coils by starting the algorithm with the base database of all 2-stranded coiled coils. We test LearnCoil on the 2-stranded coiled coils by starting the algorithm with a base database of one of the subfamilies of the 2-stranded coiled coils.

The set of iteration test sequences for testing performance on 3-stranded coiled coils consists of the following 5516 sequences: 286 known non-coiled coils from the non-redundant version of the PDB created in [7] (the PDB is the database of solved protein structures); 2% of the sequences in OWL (a large non-redundant composite database, where no two sequences in the database are exactly the same and no two sequences show only "trivial" differences [10]), with any obvious members of the PDB removed (2815 total); sequences in OWL whose names contain the strings actinin, alpha spectrin, dystrophin, tail fiber, laminin, fibrinogen, env, spike, glycoprotein, bacteriophage T4 wac, bacteriophage K3 fibritin, heat shock transcription, or macrophage scavenger receptor, as well as the 3-stranded coiled coil mutant for GCN4 (2415 total, of which many are thought to contain 3-stranded coiled coils, and the 46 sequences given below are known to contain them).

The 3-stranded coiled coil set is comprised primarily of laminin and fibrinogen sequences, as well as influenza virus hemagluttinin, Moloney murine leukemia envelope protein, 2 heat shock transcription factors, bacteriophage T4 and K3 wac proteins, the trimeric GCN4 mutant, 2 macrophage scavenger receptors, and bacteriophage T3 and T7 tail fibers.

Our set of iteration test sequences for 2-stranded coiled coils includes: 1/23 of the PIR (1553 total); the 286 known non-coiled coils; and the two of the subfamilies out of myosins, tropomyosins, and intermediate filaments. (For example, when we start with a database of intermediate filaments, our iteration test sequences include myosins and tropomyosins.)

Note that most of the sequences in our 2- and 3-stranded coiled coil data sets do not have solved structures. However, there is strong experimental support that they contain coiled coils, although often the boundaries of the coiled coil regions are difficult to specify exactly. We do not know the three dimensional structure for most of the protein sequences in our iteration test sets (except for the sequences from the PDB and portions of the sequences making up the 2- and 3-stranded coiled coil data sets).

4.2 Testing on 3-stranded coiled coils

We test the algorithm on 3-stranded coiled coils in two ways: the "leave one out" test and the "leave half out" test. In both cases, LearnCoil improves recognition of 3-stranded coiled coils starting with an initial database of 2-stranded coiled coils. We measure LearnCoil's performance on the 286 non-coiled coil proteins, and an evaluation set consisting of 3-stranded coiled coil sequences. We assume that a false negative prediction has occurred when a sequence in the 3-stranded coiled coil evaluation set receives a score with a corresponding likelihood less than 50%. We assume a false positive has occurred when a non-coiled coil protein scores at least 50% likelihood. Since our algorithm is randomized, the final likelihoods are found by averaging LearnCoil outputs over five runs.

In the first "leave one out" scenario, the algorithm is run with all the 5516 iteration test sequences described in section 4.1. Once the algorithm terminates, each of the 46 sequences in the 3-stranded coiled coil set is scored with respect to parameters calculated from the new database in the final iteration minus the effects of this sequence. That is, since the 46 3-stranded coiled coil sequences are included in the iteration test set, if a sequence appears in the final database, before scoring this sequence, the sequence is removed to avoid the possibility of unfairly biasing the test.

The weight of the original database (i.e., relative to the new database) was chosen empirically to be $\lambda = 0.1$. This makes sense because 2- and 3-stranded coiled coils are sufficiently different; thus, it may require much more weight for the newly identified sequences to effectively broaden the new database to contain 3-stranded coiled coils. We also experimented with weights in the range $0 \leq \lambda \leq 0.5$ but $\lambda = 0.1$ gave the best results.

Our algorithm LearnCoil positively identifies 43 out of 46 (93%) of the 3-stranded coiled coil sequences and makes no false positive predictions. In contrast, PairCoil positively identifies 31 out of 46 (67%) of the 3-stranded coiled coils and also makes no false positive predictions (see Table 1). Moreover, using the final databases that LearnCoil produced, we are able to recognize all the sequences in the 2-stranded coiled coil database. Thus the final databases produced by the LearnCoil algorithm performs well on both 2- and 3-stranded coiled coils.

In the second "leave half out" scenario, we split the 3-stranded coiled coil sequence set in half in the fol-

Base Set	Evaluation Set	Performance without LearnCoil		Performance with LearnCoil	
		% of seqs	# of false positive seqs	% of seqs	# of false positive seqs
2-str CCs	46 3-str CCs	67%	0/286	93%	0/286

Table 1: Learning 3-stranded coiled coils from 2-stranded coiled coils using the leave one out criterion.

lowing manner. First, the 46 3-stranded coiled coil sequences are divided into the following subgroups: α-fibrinogens, β-fibrinogens, γ-fibrinogens, laminins, tail fibers, heat shocks, and all remaining protein sequences. Next, each of these subgroups is randomly divided into two parts, one for each half; this ensures that in the final split, each half is fairly representative of examples of the 3-stranded coiled coil motif.

We split the 3-stranded coiled coil sequences 3 times in the above manner. This then gives us six different iteration and evaluation sets. Each evaluation set consists of 23 3-stranded coiled coil sequences, and the corresponding iteration test set consists of 5493 sequences (the original 5516 sequences, minus the 23 sequences in the evaluation set). We run **LearnCoil** on each of the six iteration test sets, and evaluate the algorithm by its performance on the corresponding evaluation sets (namely, those 3-stranded coiled coil sequences which are not included in the iteration test set). Note that the set of sequences with solved structures that do not contain coiled coils are included in all iteration test sets, and are scored using the leave one out criterion.

For each iteration test set, our algorithm is again run 5 times with $\lambda = .1$, and with final likelihoods averaged over the runs. Table 2 gives the performance of our algorithm on the different evaluation sets. On average, **LearnCoil** selects out 85% of the 3-stranded coiled coil sequences not originally in the set of sequences upon which it iterates. In contrast, PairCoil on average selects out 67% on the same sets of sequences. In all but one of the six experiments, the algorithm does not get any false positives from the set of solved structures. In the one scenario when it does get a false positive, the likelihood of all sequences in the corresponding evaluation set (B1) that score above 50% also score higher than this false positive.

The average performance of **LearnCoil** on the 3-stranded coiled coil sequences included in the iteration test set is 88%. (Individual performance data for each of the six experiments is not shown.) This average does not seem to be significantly higher than the algorithm's average performance on the sequences in the evaluation set. Thus in comparing the results in Table 2 with the results in Table 1, it appears that the decreased performance on these runs with the splits is the result of fewer available 3-stranded coiled coil sequences to the algorithm, and not upon whether the evaluation criterion is the leave one out criterion or the leave half out criterion.

4.3 Testing on subclasses of 2-stranded coiled coils

Our results on subclasses of the 2-stranded coiled coil motif indicate that we are able to "learn" coiled coil regions in one family of proteins starting with an initial database consisting of coiled coils from another family of proteins. Our techniques improve non-iterative approaches, such as the PairCoil program, which fail to identify many conjectured coiled coil residue positions when restricted to a database of only one protein family.

We test **LearnCoil** on three different domains (Table 3): tropomyosins (TROPs) as a base set and myosins (MYOs) and intermediate filaments (IFs) as a target set; myosins as a base set and tropomyosins and intermediate filaments as a target set; intermediate filaments as a base set and myosins and tropomyosins as a target set. A different set of iteration test sequences is used for each of these tests; that is, the set that includes sequences of the two proteins in the target set. For these experiments, we have residue data, and thus our performance measure is with respect to these. False negatives are coiled coil residues of sequences in the target set which do not have at least a 50% likelihood. False positives are defined as in section 4.2.

Here the weight of the original database was empirically chosen to be $\lambda = 0.3$. One possible explanation for this is since the subclasses of 2-stranded coiled coils have more similarities than differences, the program does not have to be so aggressive in picking up the target set. Moreover, the goal is a target set of 2-stranded coiled coils, and this is best achieved by weighting each of the 3 types of proteins equally. We also experimented with weights of $\lambda = 0.1$ and $\lambda = 0.5$, and while their overall performance was similar, they produced more false positives.

First, we consider experiments with tropomyosins in the base set and myosins and intermediate filaments in the target set. **LearnCoil** positively identifies 99% of the myosin and intermediate filament residues in the 2-stranded database and makes one false positive prediction. This is in contrast to PairCoil, which obtains a performance of 70.9%, with four false positive and two false negative predictions.

Next we consider experiments with a base set of myosins and a target set of tropomyosins and intermediate filaments. **LearnCoil** positively identifies 99% of the tropomyosin and intermediate filament residues and makes one false positive prediction. This is in contrast to PairCoil, which obtains a performance of 88.8%, with two false positive and one false negative predictions.

Lastly, we consider experiments with a base set of intermediate filaments and a target set of tropomyosins and myosins. **LearnCoil** positively identifies 99.4% of the tropomyosin and myosin residues and makes two false positive predictions. In contrast, PairCoil obtains a performance of 83.3%, with four false positive predic-

Base Set	Evaluation Set	Performance without LearnCoil		Performance with LearnCoil	
		% of seqs	# of false positive seqs	% of seqs	# of false positive seqs
2-str CCs	Set A1, 23 3-str CCs	65%	0/286	87%	0/286
2-str CCs	Set A2, 23 3-str CCs	70%	0/286	83%	0/286
2-str CCs	Set B1, 23 3-str CCs	74%	0/286	87%	1/286
2-str CCs	Set B2, 23 3-str CCs	61%	0/286	78%	0/286
2-str CCs	Set C1, 23 3-str CCs	70%	0/286	96%	0/286
2-str CCs	Set C2, 23 3-str CCs	65%	0/286	78%	0/286

Table 2: Learning 3-stranded coiled coils from 2-stranded coiled coils using the leave half out criterion. The 3-stranded coiled coil sequences are split 3 times, giving us six different iteration and evaluation sets. The evaluation sets are A1, A2, B1, B2, C1 and C2 (A1 and A2 are a result of one split, etc.).

tions. One possible explanation for more false positives here is that the intermediate filaments have a less obvious coiled-coil structure and there very well may be non-coiled coil residues in the database; consequently, starting with a table of solely intermediate filaments may select out non-coiled coils for the target database. In addition, it is speculated that some of the intermediate filaments may be heterodimers, and thus quite different from the myosins and tropomyosins (which are thought to be homodimers).

For all the three above experiments, LearnCoil improves performance of PairCoil in identifying coiled coil residues, while also improving its false positive rate.

We also tested LearnCoil with the NewCoils program [26] as the underlying scoring algorithm. For subclasses of 2-stranded coiled coils, we find that LearnCoil enhances the performance of NewCoils as well. It obtains a performance of 96.2% when tropomyosins are used as the base set, a performance of 95.3% when myosins are used, and a performance of 98.2% when intermediate filaments are used. The program does not make any false positive predictions when run on these three test domains. In contrast, the non-iterative version of NewCoils has substantial overlap between the residue scores for coiled coils and non-coiled coils in all of the three test domains.

4.4 New coiled-coil-like candidates

The LearnCoil program has identified many new sequences that we believe contain coiled-coil-like structures. Table 4 lists some examples of "newly found" viral proteins (i.e., proteins for which PairCoil indicates that no coiled coil is present, but LearnCoil indicates a coiled-coil-like structure is present). We believe that the proteins given in Table 4 either contain coiled coils or coiled-coil-like structures. Indeed, the MultiCoil distinguisher program (see *Distinguishing the base and target motifs*) indicates that the 14-3-3 protein contains a 2-stranded coiled coil, but all others contain 3-stranded coiled coils.

In Table 4, the first six proteins for which coiled coil regions are predicted are envelope glycoproteins of retroviruses. Close analysis of these envelope glycoprotein sequences (Rous sarcoma, Avian sarcoma, HTLV, equine infectious anemia virus, HIV and SIV) as well as other retrovirus envelope sequences has suggested that these proteins can be categorized in two structural groups, based on the number of coiled-coil-like regions found.

The first group contains one coiled coil region, and includes Moloney murine leukemia virus envelope protein (whose structure has been solved [17]) as well as most of the retrovirus envelope proteins. In the second group, there are two coiled-coil-like regions. These regions are thought to take part in a new coiled-coil-like structure which has been identified in recent biological work. This structure is believed to consist of a parallel, homotrimeric coiled coil encircled by three helices with a heptad-repeat pattern packed in an antiparallel formation. It is thought to be in the envelope glycoproteins of both HIV and SIV [9, 25].

Our program seems to be able to accurately predict this new coiled-coil-like structure. For example, it identifies two coiled-coil-like regions in SIV. Independently, the biological investigation of SIV by Blacklow et al. experimentally predicts that these are the two regions that are part of the coiled-coil-like structure [9]. One of these regions (comprising the outer three helices) is predicted by the NewCoils program and is given a 26% likelihood by the PairCoil program. The other region (comprising the trimeric coiled coil) is only predicted by our LearnCoil program. This region corresponds to the N-terminal fragment in the paper of Blacklow et al. In fact, the region LearnCoil predicts and the region that Blacklow et al. find are almost identical: LearnCoil predicts a coiled-coil-like structure starting at residue 553 and ending at residue 601, whereas Blacklow et al. start the region at residue 552 and end it at residue 604.

Moreover, there is biological evidence that several other of the sequences in Table 4 contain coiled-coil-like structures. Our predictions were made independently of these results. For instance, recently, the crystal structure of two 14-3-3 proteins have been solved [24, 31]. The paper of Liu et al. studies the zeta transform of the 14-3-3 structure in E. coli, and they report a 2-stranded anti-parallel coiled coil structure. On the other hand, the paper of Xiao et al. studies the human T-cell τ dimer, and they report helical bundles. Although there is some uncertainty here, it is likely that the 14-3-3 protein we have identified contains a coiled-coil-like structure, if not a coiled coil itself.

The proteins reported in Table 4 are compared to

Base Set	Evaluation Set	Performance without LearnCoil		Performance with LearnCoil	
		% of residues	# of false positive seqs	% of residues	# of false positive seqs
TROPs	MYOs + IFs	71%	4/286	99%	1/286
MYOs	TROPs + IFs	89%	2/286	99%	1/286
IFs	MYOs + TROPs	83%	4/286	99%	2/286

Table 3: Learning 2-stranded coiled coils from a restricted set

PIR Name	LearnCoil Likelihood	PairCoil Likelihood
Rous sarcoma virus, env	>90%	<10%
Avian sarcoma virus, env	>90%	<10%
human T-cell lymphotropic virus – type I, env	>90%	<10%
equine infectious anemia virus, env	>90%	<10%
HIV, env	>90%	<10%
SIV, env	>90%	26%
fruit fly 14-3-3 protein	52%	<10%
human T-cell surface glycoprotein CD4 precursor	90%	<10%
mouse hepatitis virus E2 glycoprotein precursor	>90%	23%
human rotavirus A glycoprotein NCVP5	>90%	<10%
human respiratory syncytial virus fusion glycoprotein	>90%	<10%

Table 4: Newly discovered coiled-coil-like candidates

the PairCoil program. The NewCoils program of Lupas et al. finds some of these proteins; however, in general, this program finds a significant number of false positives. The NewCoils program identifies some of the same coiled-coil-like regions in mouse hepatitis virus glycoprotein, human rotavirus glycoprotein, human respiratory syncytial virus glycoprotein, HIV envelope protein and SIV envelope protein. The 14-3-3 protein, the human T-cell lymphotropic virus envelope protein, the human T-cell surface glycoprotein CD4 precursor, Rous sarcoma virus envelope protein, Avian sarcoma virus envelope protein, and equine infectious anemia virus envelope protein are found only using our LearnCoil program. (NewCoils finds a coiled coil region in equine infectious anemia virus envelope protein, but it is different from the one LearnCoil finds.) As mentioned above, there is biological evidence that at least some of the proteins that only LearnCoil finds (e.g., the 14-3-3 protein and the retrovirus envelope proteins) contain coiled-coil-like structures.

5 Conclusions

In this paper, we have shown that an iterative algorithm that uses randomness and statistical techniques can substantially enhance existing methods for protein motif recognition. We have designed a program LearnCoil and demonstrate its ability to "learn" the 2- and 3-stranded coiled coil motif.

There is evidence that our program may have identified a new coiled-coil-like motif that occurs in the envelope glycoprotein of many retroviruses, and a companion paper on this topic is forthcoming [5]. We are also exploring an application of our algorithm to bacterial signaling proteins [4]. It is our hope that biologists will use this program to help identify other new coiled-coil-like structures.

In the future we plan to apply the LearnCoil program to motifs other than those that have coiled-coil-like properties. Limited data is a problem for many protein structure prediction problems. There are newly discovered protein motifs for which biologists cannot yet predict, and more importantly, do not yet even know the structural features that characterize the motifs. We hope to extend the techniques developed here to aid in the determination of crucial structural features that give rise to these motifs, as well as to learn how to predict which proteins exhibit this motif.

Acknowledgements

Thanks to Peter Kim, Ethan Wolf, Theodore Tonchev, Ron Rivest, Trevor Jim, and Stephen Blacklow for many helpful discussions.

References

[1] T. K. Attwood and J. B. C. Findlay. Design of a discriminating fingerprint for G-protein-coupled receptors. *Protein Engineering*, 6:167–176, 1993.

[2] P. Baldi. Hidden Markov models in molecular biology. Technical report, JPL California Institute of Technology, 1993.

[3] B. Berger. Algorithms for protein structural motif recognition. *Journal of Computational Biology*, 2:125–138, 1995.

[4] B. Berger, A. Cochran, P. S. Kim, and M. Singh, 1996. Work in progress.

[5] B. Berger, P. S. Kim, and M. Singh, 1996. In preparation.

[6] B. Berger and D. Wilson. Improved algorithms for protein motif recognition. In *Symposium on Discrete Algorithms*, pages 58–67. SIAM, January 1995.

[7] B. Berger, D. B. Wilson, E. Wolf, T. Tonchev, M. Milla, and P. S. Kim. Predicting coiled coils using pairwise residue correlations. *Proceedings of the National Academy of Sciences*, 92:8259–8263, 1995.

[8] J. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 1985.

[9] S. Blacklow, M. Lu, and P. S. Kim. A trimeric subdomain of the simian immunodeficiency virus envelope glycoprotein. *Biochemistry*, 34:14955, 1995.

[10] A. J. Bleasby and J.C. Wooton. *Protein Engineering*, 4, 1990.

[11] C. Branden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, Inc., New York and London, 1991.

[12] M. Brown, R. Hughey, A. Krogh, I. S. Mian, K. Sjölander, and D. Haussler. Using Dirichlet mixture priors to derive hidden Markov models for protein families. In *International Conference on Intelligent Systems and Molecular Biology*, pages 47–55, 1993.

[13] P. A. Bullough, F. M. Hughson, J. J. Skehel, and D. C. Wiley. Structure of influenza hemagglutinin at the pH of membrane fusion. *Nature*, 371:37–43, 1994.

[14] C. Carr and P. S. Kim. A spring-loaded mechanism for the conformational change of influenza hemagglutinin. *Cell*, 73:823–832, May 1993.

[15] I. Dodd and J. B. Egan. Improved detection of helix-turn-helix DNA-binding motifs in protein sequences. *Nucleic Acid Research*, 18:5019–5026, 1990.

[16] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc., New York, 1973.

[17] D. Fass, S. Harrison, and P. Kim. Retrovirus envelope domain at 1.7 Angstrom resolution. *Nature Structural Biology*, 3(5), May 1996.

[18] D. Fass and P. S. Kim. The envelope protein of moloney murine leukemia virus contains a three-stranded coiled coil: Structural similarity between retrovirus and influenza membrane-fusion proteins. Submitted for publication, 1995.

[19] V. Fischetti, G. Landau, J. Schmidt, and P. Sellers. Identifying periodic occurrences of a template with applications to protein structure. *Information Processing Letters*, 45(1):11–18, 1993.

[20] Irving John Good. *The Estimation of Probabilities*. The MIT Press, Cambridge, MA, 1965.

[21] M. Gribskov. Translational initiation factors IF-1 and eIF-2α share an RNA-binding motif with prokaryotic ribosomal protein S1 and polynucleotide phosphorylase. *Gene*, 119:107–111, 1992.

[22] L. Holley and M. Karplus. Protein secondary structure prediction with a neural network. *Proceedings of the National Academy of Sciences*, 86:152–156, 1989.

[23] A. Krogh, M. Brown, S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. Technical Report UCSC-CRL-93-32, University of California at Santa Cruz, 1993.

[24] D. Liu, J. Bienkowska, C. Petosa, R. J. Collier, H. Fu, and R. Liddington. Crystal structure of the zeta isoform of the 14-3-3 protein. *Nature*, 376, July 13, 1995.

[25] M. Lu, S. Blacklow, and P. S. Kim. A trimeric structural domain of the HIV-1 transmembrane glycoprotein. *Nature Structural Biology*, 2(12), December 1995.

[26] A. Lupas, M. van Dyke, and J. Stock. Predicting coiled coils from protein sequences. *Science*, 252:1162–1164, 1991.

[27] R. Maclin and J. Shavlik. Refining algorithms with knowledge-based neural networks: improving the Chou-Fasman algorithm for protein folding. In Stephen Hanson, George Drastal, and Ronald Rivest, editors, *Computational Learning Theory and Natural Learning Systems*, pages 249–286. The MIT Press, 1994.

[28] D. A. D. Parry. Coiled coils in alpha-helix-containing proteins: analysis of residue types within the heptad repeat and the use of these data in the prediction of coiled-coils in other proteins. *Bioscience Reports*, 2:1017–1024, 1982.

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

[30] R. Tatusov, S. Altschul, and E. Koonin. Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *Proceedings of the National Academy of Science*, 91:12091–12095, December 1994.

[31] B. Xiao, S. Smerdon, D. Jones, G. Dodson, Y. Soneji, A. Aitken, and S. Gamblin. Structure of a 14-3-3 protein and implications for coordination of multiple signalling pathways. *Nature*, 376, July 13, 1995.