Mining Residue Contacts in Proteins Using Local Structure Predictions

Mohammed J. Zaki[†], Shan Jin[†], Chris Bystroff[‡],
[†]Computer Science Department
[‡]Biology Department
Rensselaer Polytechnic Institute, Troy, NY 12180
zaki@cs.rpi.edu, bystrc@rpi.edu, jins@cs.rpi.edu

1 Introduction

Today we are witnessing a paradigm shift in predicting protein structure from its known amino acid sequence $(a_1, a_2, \cdots, a_n)$. The traditional or Ab initio folding method employed first principles to derive the 3D structure of proteins. However, even though considerable progress has been made in understanding the chemistry and biology of folding, the success of ab initio folding has been quite limited.

Instead of simulation studies, an alternative approach is to employ learning from examples using a database of known protein structures. For example, the Brookhaven Protein Database (PDB) records the 3D coordinates of the atoms of thousands of protein structures. Most of these proteins cluster into around 700 fold-families based on their similarity. It is conjectured that there will be on the order of 1000 fold-families for the natural proteins [17]. The PDB thus offers a new paradigm to protein structure prediction by employing data mining methods like clustering, classification, association rules, hidden Markov models, etc.

A fascinating property of protein chains is that they spontaneously and reproducibly fold themselves into complex three-dimensional globules when placed in an aqueous solution. The sequence of amino acids making up the polypeptide chain contains, encoded within it, the complete building instructions. This self-organization cannot occur by a random conformational search for the lowest energy state [10], since such a search would take millions of years, while proteins fold in milliseconds. In recent years, a combination of molecular biological and biophysical techniques have dissected the folding process into fast and slow components which localize to certain parts of the amino acid sequence [14].

Some small, fast-folding regions of the molecule may be identified by their sequence alone. A library of short sequence patterns that fold fast has been compiled by cluster analysis of the database of known protein structures (the I-sites Library, [3]). In this work, similar short sequences that mapped to the same local structure in different proteins were deemed to be autonomous folding units, and the short sequences were compiled into patterns or “profiles” which could then be used to predict whether or not a segment of the protein would tend to fold independently of the rest of the molecule. Cross-validation showed a strong statistical significance to the predictions made by the profiles, and later NMR studies showed that some peptides predicted to fold in isolation actually did so [19]. Peptides with a strong tendency to fold independently constitute about 30% of the amino acid residues in protein sequences. The formation of independent folding units (I-sites motifs) is the first level of self-organization in the folding process: the "initiation."

These short motifs occur in proteins of widely differing topology, and so cannot contain sufficient information to define the overall, global fold of the protein molecule. Moreover, they are too short to be the fast-folding regions found by experimental dissection. There must be a higher level of self-organization which dictates how the short pieces come together to form larger, longer globular domains. The rules defining the propagation of structure along the chain, starting from the sites of initiation, have been extracted from the database of known protein structures and formalized as a hidden Markov model (HMM), called HMMSTR [4] (or "hamster"), discussed further below. HMMSTR models the interactions between adjacent short regions of the sequence, and so attempts to model the second level of self-organization: "propagation" of structure along the sequence.

The I-sites Library models the initiation sites of folding, and the new HMM models interactions between those sites. But HMMSTR [4] is a network of connections between I-sites motifs, and thus simultaneously models both folding initiation and propagation. The two levels of complexity, not discretely defined but smoothly intermingled, are represented in the HMM as variable degrees of branching. Unbranched segments are initiations sites, whose probabilities depend simultaneously on short contiguous segments of the sequence, while branching and cycles represent multiple sequence-dependent ways of extending and linking the initiation sites. Arbitrary levels of complexity may be modeled by including HMMs recursively within overarching HMMs, the latter representing the ways of connecting the output of the HMMs it contains. Hidden Markov models are limited to data that can be expressed as one-dimensional sequences of discrete symbols, but there are techniques for overcoming both the discreteness and the one-dimensionality [13].

The next level of complexity in protein folding is called "condensation". In the first few microseconds after introducing the polypeptide chain into an aqueous solution, initiation sites form transient, rapidly-interchanging structures, favoring one or more conformations to varying degrees. These structures propagate along the chain by promoting compatible upstream and downstream conformations, and the resulting transiently-formed substructures encounter each other by through-space diffusion, condensing into larger, ordered globules, as energy dictates. The ordering of these three processes is not discrete but overlapping, and they should therefore be integrated into a single computational model. Modeling of the condensation step given predictions based on the modeling of initiation/propagation is the subject of the present work. A single Markov state prediction implies a local substructure and a single amino acid position within it. Thus, a contact between two Markov states implies a specific mode of condensation between two local substructures to form tertiary structure.

The contact map of a protein (see Figure 1) is a particularly useful representation of protein tertiary structure. Two amino acids in a protein that come into contact with each other form a non-covalent interaction (hydrogen-bonds, hydrophobic effect, etc.). More formally, we say that two residues (or amino acids) a_i and a_j in a protein are in *contact* if the 3D distance $\delta(a_i, a_j)$ is less than some threshold value t (in this paper we use $t = 7\mathring{A}$ as the threshold distance), where $\delta(a_i, a_j) = |\mathbf{r_i} - \mathbf{r_j}|$, and $\mathbf{r_i}$ and $\mathbf{r_j}$ are the coordinates of the α-Carbon atoms of amino acids a_i and

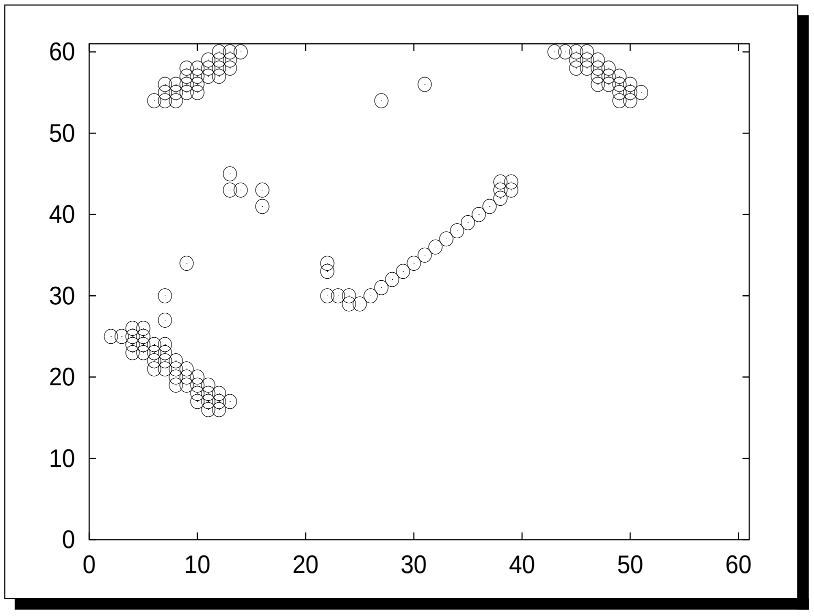

Figure 1: Contact Map (PDB file 2igd, $N = 61$)

a_j. We define *sequence separation* as the distance between two amino acids a_i and a_j in the amino acid sequence, given as $|i - j|$. A contact map for a protein with N residues is an $N \times N$ binary matrix C whose element $C(i, j) = 1$ if residues i and j are in contact, and $C(i, j) = 0$ otherwise. The contact map provides a host of useful information. For example, secondary structure can easily be discerned from it. α-Helices appear as thick bands along the main diagonal since they involve contacts between one amino acid and its four successors, while β-Sheets are thin bands parallel or anti-parallel to the main diagonal, etc. However, tertiary structure is not easily found from the contact map. For predicting the elusive global fold of a protein we are usually interested in only those contacts that are far from the main diagonal. In this paper we thus ignore any pair of residues whose sequence separation $|i - j| < 4$.

Previous work on contact prediction has employed Neural Networks [6], and statistical techniques based on correlated mutations [12, 15]. Recent work by Vendruscolo et al [16] has also shown that it is possible to recover the 3D structure from even corrupted contact maps. In this paper we present a new hybrid technique for contact map prediction. We first predict local structural elements using an HMM. The HMM simultaneously represents the initiation and propagation steps of protein folding. We then apply association mining technique on top of the HMM states to predict the states that frequently co-occur with contacts. These sets are then used for predicting contacts in unseen proteins. Our model obtains 19% accuracy and coverage over the set of all proteins; the model is also 5.2 times better than a random predictor. We can significantly enhance coverage to over 40% if we sacrifice accuracy (13%). For short proteins (length< 100) we get 30% accuracy and coverage (4.5 times better than random); if we lower accuracy to 26% we can get coverage upto 63%. We believe that these results are better than (or equal to) those reported previously.

2 Hybrid Mining Approach

Here we describe the hybrid technique used in conjunction, to predict residue contacts. We first use an HMM to predict local substructures within the protein. We then use meta-level mining on the output of the HMM using association rule mining. The following sections provide a brief introduction to these two methods (readers familiar with them can safely skip ahead).

2.1 Hidden Markov Models

The description of HMM below is based on the excellent tutorial by Rabiner [13]. An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of observed symbols.

An HMM is made up of a finite number N of states. At each time step t a new state is entered based on a transition probability distribution which depends on the previous state (the Markovian property). After each transition is made, an observation output is produced according to a fixed probability distribution which depends on the the current state. Thus there a N such observation probability distributions.

As an example of modeling proteins via HMMs, let consider an “urn and residue” model. There are N urns (or states) each filled with a large number of the 20 possible amino acids. The observation sequence (a protein?) is generated by initially choosing one of the N urns (according to an initial probability distribution), selecting a residue from the initial urn, recording which amino acid it is, replacing it, and then choosing a new urn (state) according to a transition probability distribution associated with the current urn. A step corresponds to a residue position. Thus a typical observation sequence might be:

step or position	1 2 3 4 $\cdots$ T
urn (hidden) state	$q_3 q_1 q_1 q_2 \cdots q_{N-2}$
amino acid (observation)	G L A K $\cdots$ S

An HMM is made up of the following components: T is the length of the observation sequence; N the number of states in the model; M the number of observation symbols (for simplicity we assume here that the output is a discrete symbol, e.g. an amino acid. However we actually use a continuous vector output as we shall see later); $Q = \{q_1, q_2, \cdots q_N\}$ is set of HMM states; $V = \{v_1, v_2, \cdots, v_M\}$ is the set of output symbols; $A = \{a_{ij}\}$ gives the set of state transition probabilities, i.e., $a_{ij} = P(q_j \ at \ t+1 | q_i \ at \ t)$; $B = \{b_j(k)\}$ is the output symbol probability distribution in state q_j, i.e., $b_j(k) = P(v_k \ at \ t | q_j \ at \ t)$; and finally $\pi = \{\pi_j\}$ gives the initial state distribution, i.e., $\pi_j = P(q_j \ at \ t = 1)$.

Using the model, an observation sequence $O = O_1 O_2 \cdots O_T$ is generated as follows: 1) choose an initial state i_1 based on π, 2) set position $t = 1$, 3) choose O_t according to $b_{i_t}(k)$, 4) choose i_{t+1} according to $\{a_{i_t i_{t+1}}\}$, $i_{t+1} = 1, 2, \cdots N$, and 5) set $t = t + 1$; return to step 3 if $t < T$; otherwise terminate the procedure.

An HMM can be compactly represented using the notation $\lambda = (A, B, \pi)$. There are three key problems that have to be solved to build a useful model: 1) *Evaluation Problem:* Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model $\lambda = (A, B, \pi)$, how to compute the probability of the observed sequence $P(O|\lambda)$. This can be solved using the Forward-Backward algorithm [13]. 2) *Estimation Problem:* Given the observation sequence $O = O_1O_2 \cdots O_T$, how to choose a state sequence $I = i_1i_2 \cdots i_T$, which is optimal in some meaningful sense. This can be solved using the Viterbi algorithm [13]. 3) *Maximization Problem:* How to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$. This can be solved using the Baum-Welch reestimation method [13]. We will discuss in the next section the exact details of how the HMM is built to model proteins.

2.2 Association Rules

Since its introduction, Association Rule Mining (ARM) [1], has become one of the core data mining tasks, and has attracted tremendous interest among data mining researchers and practitioners. ARM is an undirected or unsupervised data mining technique, which works on variable length data, and it produces clear and understandable results. It has an elegantly simple problem statement, that is, to find the set of all subsets of items or attributes that frequently occur in many database records or examples, and additionally, to extract the rules telling us how a subset of items influences the presence of another subset.

The association mining task can be stated as follows: Let $\mathcal{I}$ be a set of items, and $\mathcal{D}$ a database of examples, where each example has a unique identifier (*tid*) and contains a set of items. A set of items is also called an *itemset*. An itemset with k items is called a k-itemset. The *support* of an itemset X, denoted $\sigma(X, \mathcal{D})$, is the number of examples in $\mathcal{D}$ where it occurs as a subset. An itemset is *frequent* or *large* if its support is more than a user-specified *minimum support (min_sup)* value.

An *association rule* is an expression $A \Rightarrow B$, where A and B are itemsets. The support of the rule is the joint probability of a example containing both A and B, and is given as $\sigma(A \cup B)$. The *confidence* of the rule is the conditional probability that an example contains B, given that it contains A, and is given as $\sigma(A \cup B)/\sigma(A)$. A rule is *frequent* if its support is greater than *min_sup*, and it is *strong* if its confidence is more than a user-specified *minimum confidence (min_conf)*.

The data mining task is to generate all association rules in the database, which have a support greater than *min_sup*, i.e., the rules are frequent, and which also have confidence greater than *min_conf*, i.e., the rules are strong. In this paper we are interested in rules with a specific item, called the *class*, as a consequent, i.e., we mine rules of the form $A \Rightarrow \mathbf{c_i}$ where $\mathbf{c_i}$ is a class attribute ($1 \leq i \leq k$).

This task can be broken into two steps:

1. Find all frequent itemsets having minimum support for at least one class $\mathbf{c_i}$. The search space for enumeration of all frequent itemsets is 2^m, which is exponential in m, the number of items. However, if we assume that there is a bound on the example length, we can show that ARM is essentially linear in the database size [21].

2. Generate strong rules having minimum confidence, from the frequent itemsets. We generate and test the confidence of all rules of the form $X \Rightarrow \mathbf{c_i}$, where X is frequent.

2.2.1 Mining Frequent Closed Itemsets

We mine the frequent sets based on the Formal Concept Analysis approach [7], which is a very elegant mathematical framework for extracting "concepts" from databases.

Consider an itemset X. Let $Y = \{E \in \mathcal{D} | X \subseteq E\}$ be the set of all examples E in the database $\mathcal{D}$ where X occurs. Further let $X' = \{i \in \mathcal{I} | i \in \cap_{E \in Y} E\}$ be the set of all items that are common to all examples in the set Y. Then we say that X is *closed* if $X = X'$. In other words X is the maximal set of items that is common to all examples in Y. A closed itemset is also called a *concept*.

The set of all closed frequent itemsets can be orders of magnitude smaller than the set of all frequent itemsets, especially for real (dense) datasets [20]. At the same time, we don't loose any information; the closed itemsets uniquely determine the set of all frequent itemsets and their *exact* frequency. Thus instead of mining all the frequent itemsets we only mine the frequent closed itemsets using the CHARM algorithm [22] we recently developed. A detailed description of the algorithm is beyond the scope of this paper. Suffice it to say that CHARM can handle very large disk-resident or external memory databases; it has been tested on databases with millions of examples, and it scales linearly in the database size. We refer the reader to [22] for the algorithm description and its efficiency.

3 HMMSTR: An HMM for local structure in proteins

We describe here the hidden Markov model, HMMSTR [4], for general protein sequences based on the I-sites library of sequence-structure motifs [3]. In the next section we will show how we apply association mining on the output of HMMSTR to predict residue contacts.

Unlike the linear HMMs used to model individual protein families [5], HMMSTR has a highly branched topology and captures recurrent local features of protein sequences and structures that transcend protein family boundaries. The model extends the I-sites library by describing the adjacencies of different sequence-structure motifs as observed in the protein database, and achieves a great reduction in parameters by representing overlapping motifs in a much more compact form.

The I-sites (Invariant or Initiation sites) library consists of an extensive set of short sequence motifs, length 3 to 19, obtained by exhaustive clustering of sequence segments from a non-redundant database of known structures [3, 8]. Each sequence pattern correlates strongly with a recurrent local structural motif in proteins. Approximately one third of all residues in the database are found in an I-sites motif that can be predicted with a high degree of confidence ($>$ 70%). The library is non-redundant in that no motif is completely contained within another, longer motif. However, many of the motifs overlap. Furthermore, the isolated motif model does not capture higher order relationships such as the distinctly non-random transition frequencies between the different motifs. The redundancy inherent in the I-sites model suggests a better representation that would model the diversity of the motifs and their higher order relationships while condensing features they have in common. A hidden Markov model is well suited to this purpose.

3.1 Description of HMMSTR

Each of the 262 I-sites motif was represented as a chain of Markov states, each of which contains information about the sequence and structure attributes of a single position in the motif. Adjacent positions were represented by transitions from one state to the next. Hierarchical merging of these linear chains of states, based on sequence and structure similarity, resulted in a graph containing almost all the motifs. The merged graph of I-sites motifs comprises a network of states connected by probabilistic transitions, or more specifically, an HMM as shown in Figure 2.

Each state in HMMSTR can produce, or ”emit”, amino acids and structure symbols according to a probability distribution specific to that state. There are four probability distributions defined for the states in HMMSTR, b, d, r, and c, which describe the probability of observing a particular amino acid, secondary structure, backbone angle region, or structural context descriptor, respectively. A context descriptor represents the classification of a secondary structure type according to its context. For example, a hairpin turn is distinguished from a diverging turn, and a beta-strand in the middle of a sheet is distinguished from one at the end of a sheet. More formally, for a given state q_i, there are a set of emission probabilities, collectively called B_i. Here, we use four in this collection, denoted b, d, r, and c. The values $b_i(m)$ ($1 \le m \le 20$) are associated with probabilities for the emission of amino acids. The values $d_i(m)$ ($1 \le m \le 3$), are the probabilities of emitting helix(H), strand(S) or loop(T), respectively. The values $r_i(m)$ ($1 \le m \le 11$) are the probabilities of emitting one of the 11 dihedral angle symbols. Finally, $c_i(m)$ ($1 \le m \le 10$) are probabilities of emitting one of ten structural context symbols.

The database is encoded as a linear sequence of amino acids and structural observables. The amino acid sequence data consists of a “parent” amino acid sequence of known three-dimensional structure, and an amino acid profile obtained by alignments to the parent sequence [3]. The amino acid of the parent sequence is denoted by O_t, and the profile by $\{O_t^m\}(1 \le m \le 20)$. For the structural identifiers at each position t, the following nomenclature is used: 3-state secondary structure D_t, discrete backbone angle region R_t, and the context symbol C_t. A sequence s of length T is given by the values of the attributes at all positions $s_t = \{O_t, \{O_t^m\}, D_t, R_t, C_t\}$ ($1 \le t \le T$). The utility of the HMM to model database sequences is based on the notion of a path. A path is a sequence of states through the HMM, denoted $Q = q_1 q_2 \cdots q_T$. Thus, the probability of a sequence s given the model λ, $P(s|\lambda)$, is obtained by summing the relevant contributions from all possible paths Q:

$$P(s|\lambda) = \sum_{all\ I} \pi_{i_1} B_{i_1}(s_1) a_{i_1 i_2} b_{i_2}(s_2) \cdots a_{i_{T-1} i_T} b_{i_T}(s_T)$$

where $I = i_1 i_2 \cdots i_T$ is a fixed sequence of states and $B_i(s_t)$ is the probability of observing s_t at state q_i, which for observation of a single sequence is given by

$$B_i(s_t) = \begin{pmatrix} d_i(D_t) \\ r_i(R_t) \\ c_i(C_t) \end{pmatrix} b_i(O_t)$$

Usually, only one of the structural emission symbols d, r, or c is included in B_i in any given training run. However, in principle, any combination could be used. Our HMMs showed significant improvements in performance when we used amino acid profiles instead of single amino acid sequences for training and for subsequent predictions. For the probability of observing a given

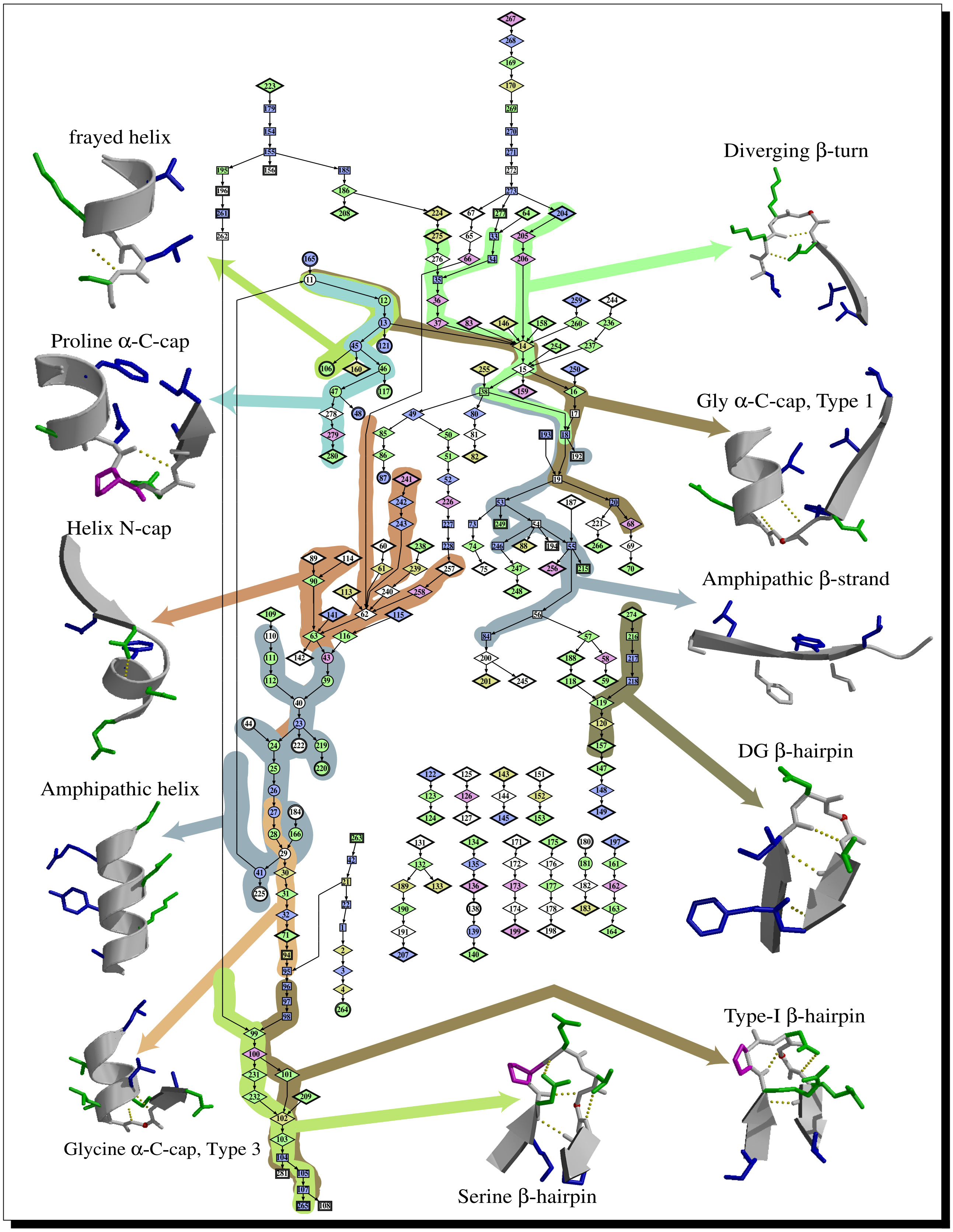

Figure 2: HMMSTR model built from I-sites Library. For a detailed description, see [4]

profile $\{O_t^m\}$ position t in a sequence, we use the multinomial distribution, and the expression for B becomes

$$B_i(s_t) = \begin{pmatrix} d_i(D_t) \\ r_i(R_t) \\ c_i(C_t) \end{pmatrix} \sum_{m=1}^{20} b_i(m)^{Ncount \times O_t^m}$$

To give equal weight to the information in sequence families of different depths, $Ncount$ was taken to be a global parameter.

3.2 Training HMMSTR

For training, evaluation and testing of the HMMSTR we used a non-redundant database of proteins of known structure, PDBselect:December 1998 [9] containing 691 proteins and their sequence families. The proteins in the set have $< 25\%$ sequence similarity. Entries in the database were selectively removed if the structure was solved by NMR, had a large number of disulfide bridges or cis-peptide bonds, or if it was a membrane-associated protein according to the header records. Disordered or missing coordinates in the middle of a sequence were addressed by dividing the sequence at that point. Contiguous segments of length less than 20 were ignored. Multiple sequence alignments were generated from each sequence using PSI-BLAST [2] after filtering the query sequences for low-complexity regions [18]. Data for training the HMM included the sequence profile, computed from the multiple sequence alignment as described before [3], the DSSP secondary structure assignments [11], the backbone angles, and a structural "context" symbol.

Backbone angles were measured from the coordinates and assigned, using a Voronoi method, to 11 regions of phi/psi space. The centroids of 10 regions were chosen by K-means clustering of a large subset of trans phi/psi pairs from the database. The 11th region is all cis peptides.

A randomly selected set of 73 of the 691 proteins (19,000 positions) was then set aside and not used for training, but only for the final cross-validation. Before cross-validation, a test for true independence was applied to each member of the test set, and 12 members were removed. The final test set thus contained 61 proteins and 16,000 positions.

The remaining set of 618 parent sequences (145,000 positions) was used for training, and divided into a large set of 564 sequences (133,000 positions), used for optimization via the Expectation-Maximization algorithm, and a small set of 54 sequences (12,000 positions) used to evaluate the predictive ability of the model during training. Note that the small set of 54 sequences is used only for evaluation of the performance of a model and may thus appear to be a test set. However, decisions regarding the modification of the model are based on results of those evaluations. The set of 54 sequences is therefore not a test set, but a training set. For the final round of training we re-combined the large and small training sets, to a total of 618 sequence families. After the final round of training, the models were frozen.

4 Data Format and Preparation

After HMMSTR is built we again took the 691 proteins from PDBSelect and computed for each protein the optimal HMMSTR states that agree with the observed amino acids in the protein. In other words for each protein sequence-structure we solve the estimation problem, i.e., given the observation sequence $O = O_1 O_2 \cdots O_T$, how to choose a state sequence $I = i_1 i_2 \cdots i_T$, which is optimal. The output probability distributions of all the states thus chosen for a protein sequence is used as input for the association mining algorithm. In fact, rather than a single state associated with a given residue, we have available the probability that the residue at the given position is associated with all the states of HMMSTR, i.e., we have available $P(q_i|a_j)$ for all the 282 HMMSTR states ($1 \leq i \leq 282$) for all the residues in a given protein ($1 \leq j \leq n$, where n is the length of the protein). For each residue we also know the amino acid at that position; the b, d, r, and c outputs, which describe the probability of observing a particular amino acid, secondary structure, backbone angle region, or structural context descriptor, respectively; the spatial coordinates of the α-Carbon atom $\langle x, y, z \rangle$; a distance vector of length n giving the distance of this residue from all other residues in the protein; and the 20 amino acid profiles for that position. A protein data file may look like this:

```
PDB Name: 153l_
Sequence Length: 185

Position: 1
Residue: R
Coordinates: 0.0  -73.2  177.6
Profile: 0.0 ... 1.0 ... 0.0 #20 Values
HMMSTR State Probabilities:
        0.0 ... 0.7 .... 0.3 ... 0.0 #282 Values
Distance Vector: 0 3 5 ... 18 15 13 #185 Values, i.e., Seq Length

Position: 2
Residue: T
Coordinates: -124.4  0.2  -177.1
Profile: 0.0 ... 1.0 ... 0.0 #20 Values
HMMSTR State Probabilities:
        0.0 ... 0.9 ... 0.1 ... 0.0 #282 Values
Distance Vector: 3 0 3 ... 15 13 10 #185 Values

...

Position: 185
Residue: Y
Coordinates: -88.7  0.0  0.0
Profile: 0.0 ... 0.4 ... 0.6 ... 0.0 #20 Values
HMMSTR State Probabilities:
        0.0 ... 0.2 ... 0.5 ... 0.3 ... 0.0 #282 Values
Distance Vector: 15 13 10 ... 5 3 0 #185 Values
```

We have a file like the one shown above for all of the 691 non-redundant set of proteins from PDBSelect. Disordered or missing coordinates in the middle of a protein sequence were addressed by dividing the sequence at that point. This produces a set of 794 files, most of them containing an entire protein sequence, but some of these correspond to proteins that were split.

Given a protein file, we now have to transform the data into a format that can be easily mined for frequent closed itemsets, i.e., we need to prepare the data in the relational or tabular format where we have multiple attributes (columns) for each example (rows) or record. Since we are interested in predicting the contact between a pair of amino acids, we use each pair as an example in the training set, associated with a special *class* attribute indicating whether it is a contact (C) or non-contact (NC); amino acids a_i and a_j are in contact if $\delta(a_i, a_j) < 7\mathring{A}$, i.e., the distance between α-carbons of amino acids a_i and a_j is less then $7\mathring{A}$. Our new database has an entry showing the two amino acids and their class for each pair of amino acids for each protein. In order to avoid predicting purely local contacts we ignore all pairs whose sequence separation $|i - j| < 4$. Note also that the number of contacts N_C is a lot smaller than the number of non-contacts N_{NC} for any protein.

We found that the percentage of contacts (or number of database entries with class 1) over all pairs is less than 1.7%. Across the 794 files, the longest sequence had length 907, while the smallest had length 35. There were 17,618,115 pairs over all proteins, while only 292,126 pairs were in contact. This database thus corresponds to a highly biased binary classification problem. That is, we have to build a mining model that can discriminate between contacts and non-contacts between amino acids pairs, where the examples are overwhelmingly biased towards the non-contacts.

Our database so far doesn't have enough information for good discrimination. All we have is the amino acids making up the pair and whether they are in contact or not. We need to add more "context" information to facilitate the classification. It is easy to incorporate, for each amino acid in the pair, the 3 secondary structure symbols (d_i, d_j), the 11 backbone angle regions (r_i, r_j), and the 10 structural context descriptors (c_i, c_j). For each pair we would also like to add the HMMSTR state probabilities. Since association rules only work for categorical attributes, we need to convert the continuous state probabilities into discrete values. To do this we take the ratio of each of the 282 HMMSTR state probabilities for a_i against the background or prior probability of an amino acid being in that state; if the ratio is more than some threshold we include the state in the context of a_i, else we ignore it. We repeat the same process for a_j. Using a similar thresholding method one can incorporate the amino acid profiles for positions i and j. With all this context information for both a_i and a_j we obtain a new database to be used to find the frequent itemsets characterizing the contacts and non-contacts. In summary the database might have the following columns for pairs of amino acids over all proteins:

```
Protein and Position Information: ProteinID  PairID i j |i-j|
Amino Acids and Context: ai aj di dj ri rj ci cj
Profile: pi1 pi2 ... pj1 pj2 ...
HMMSTR: qi1 qi2 ... qj1 qj2 ...
Class: C or NC
```

Note that the number of columns can be variable for different pairs depending on the profile and HMMSTR state probabilities. p_{i_1}, p_{i_2}, etc. show the other amino acids that can appear in position i (provided the probability is more than some threshold), and finally q_{i_1}, q_{i_2}, etc. show HMMSTR states with probabilities more than some factor of the prior probability of those states.

5 Association Mining on the Pairs Database

We are now in a position to cast the above database in the association framework. Each attribute-value pair is an item, and is represented with a fixed, unique integer. For example $a_i = G$ is one item and $a_i = L$ is another item. By the same token each value of c_i, d_i, r_i, L_i, and R_i is a different item. Each of the HMMSTR states becomes a distinct item, as do the profile values. The items for the context attributes of a_i and a_j are also kept distinct. Finally we separate the examples that are contacts from those that are non-contacts to get two databases, denoted as $\mathcal{D}_C$ and $\mathcal{D}_{NC}$, respectively.

Given these databases our goal is to find high support and high confidence rules of the form $A \Rightarrow C$ and $A \Rightarrow NC$, that discriminate between the contact pairs and the non-contact pairs, respectively. Below we describe the mining/training and testing phases, where we learn from examples using the frequent closed itemsets, and then classify unseen examples as being contacts or non-contacts, respectively.

5.1 Mining on Known Examples

The goal of the mining phase is to learn from known contact and non-contact examples and build a model or rule set that discriminates between the two classes. We selected a random 90% of the files for training, out of a total of 794 files. The remaining 10% of the files were kept aside for testing the mined rule set.

Since we are primarily interested in predicting the contacts rather than the non-contacts, we mine only on the contacts database $\mathcal{D}_C$. However, we do use the non-contacts database $\mathcal{D}_{NC}$ to prune out those itemsets that are frequent in both sets. Building a discriminative rule set consists of the following steps, in order:

1. *Mining*: We use CHARM [22] to mine all the frequent closed itemsets in $\mathcal{D}_C$ based on a suitably chosen *min_sup* value. Let's denote the set of frequent closed itemsets as $\mathcal{F}$.

2. *Counting*: We compute the support of all itemsets in $\mathcal{F}$ in the non-contacts database $\mathcal{D}_{NC}$.

3. *Pruning*: We compute the probability of occurrence of each itemset in $\mathcal{F}$ in both the contact and non-contact databases. The probability of occurrence is simply the support of the itemset divided by the number of examples in the given dataset. For example, if itemset $X \in \mathcal{F}$, then the probability of its occurrence in $\mathcal{D}_C$ is given as $P(X, \mathcal{D}_C) = \sigma(X, \mathcal{D}_C)/|\mathcal{D}_C|$.

   As a first step in pruning we can remove all itemsets $X \in \mathcal{F}$ which have a greater probability of occurrence in the non-contact database than in the contact database, i.e., if $P(X, \mathcal{D}_{NC}) > P(X, \mathcal{D}_C)$. Actually, we compute the ratio of the contact probability versus the non-contact probability for X, and prune it if this ratio is less than some suitably chosen threshold ρ, i.e., we prune X if $P(X, \mathcal{D}_C)/P(X, \mathcal{D}_{NC}) < \rho$. In other words we want to retain only those itemset that have a much greater chance of predicting a contact rather than a non-contact.

5.2 Testing on Unknown Examples

The goal of the testing phase is to find how accurately the mined set of rules predict the contacts versus the non-contacts in new examples not used for training. We used a random 10% of the files in the database for testing. The test set had a total of 2,336,548 pairs, out of which 35,987 or 1.54% were contacts. Since we do know the true class of each example it is easy for us to find out how well our rules are for prediction.

For testing we generate a combined database $\mathcal{D}_t$ containing all pairs of amino acids in contact or otherwise. For each example we know the true class. We assign each example a predicted class using the following steps:

1. *Evidence Calculation* For each example E in the test dataset $\mathcal{D}_t$, we compute which itemsets in the set of mined and pruned closed frequent itemsets $\mathcal{F}$ are subsets of E. Let's denote the set of these itemsets as S. We next calculate the cumulative contact and non-contact support for example E, i.e., the sum of the supports of all itemsets in S in the contact and non-contact database. Finally, we compute the evidence for E being a contact, i.e., we take the ratio of the cumulative contact support over cumulative non-contact support, denoted as ρ_E. Any example E with zero contact support is taken to be a non-contact and discarded, and only the examples or test pairs with positive contact support are retained for the next step.

2. *Prediction* To make the final prediction if a test pair of residues is in contact or not, we sort all test examples E (with positive cumulative contact support) in decreasing order of contact evidence ρ_E. Finally, the top γ fraction of examples in terms of ρ_E are predicted to be contacts and the remaining $1 - \gamma$ fraction of examples as non-contacts. How γ is chosen will be explained below.

5.3 Model Accuracy and Coverage

In predicting contacts versus non-contacts for the test examples, we have to evaluate the mined model based on two metrics: *Accuracy* and *Coverage*. Furthermore, we are only interested in the prediction of contacts; thus accuracy and coverage is only considered for contacts. Accuracy is the ratio of correct contacts to the predicted contacts, while coverage is the percentage of all contacts correctly predicted. Thus, accuracy tells us how good the model is, while coverage tells us the number of contacts predicted.

More formally, let N_{tc} denote the number of true contacts in the test examples, N_{pc} the number of predicted contacts, N_{tpc} the number of true predicted contacts, and let N_a denote the number of all possible contacts, i.e., $N_a = (N - 3) \times (N - 2)/2$ (where N is the protein length), since the contact map is symmetric and pairs with sequence separation less than 4 are ignored. The accuracy of the model is given as:

$$A = N_{tpc}/N_{pc}$$

The coverage of the model is given as:

$$C = N_{tpc}/N_{tc}$$

The number of contacts predicted N_{pc} of course depends on how we chose γ, since the top γ fraction of test examples based on evidence is predicted as contacts. Since a protein is characterized by N_{tc} true contacts, we set $\gamma = N^*_{tc}/N^*_a$ and then predict the top γ fraction of examples as contacts. Note that N^*_{tc} and N^*_a denote the actual contacts and all pairs, respectively, that have positive contact support, since we discard examples with zero contact support. By adopting the above method, the number of predicted contacts is limited to those actually present in the protein. Further, this method has been used by previous approaches to contact map prediction [6, 12], and we retain it to facilitate comparison with previous results.

We also compare our model against a random predictor. The accuracy of random prediction of contacts is defined as:

$$A_r = N_{tc}/N_a$$

6 Experimental Results

We mined the pairs database using various combinations of context information and then tested the model on the unseen proteins. The pairs databases for training and testing had the following approximate sizes: $\mathcal{D}_C = 32MB$ for the training contacts database, $\mathcal{D}_{NC} = 2GB$ for the training non-contacts database, and $\mathcal{D}_t = 340MB$ for the testing database (includes both contacts and non-contacts). For all experiments below, we used a minimum support of 0.5% in the contact database, and we pruned a pattern if the ratio of contact to non-contact frequency was less than 4 (except for the amino acids only case where we used a ratio of 1.5).

Amino Acids Only Our first goal was to test how much information is contained in the amino acids only, i.e., for both training and testing, each example consisted of only the two amino acids a_i and a_j, and nothing else. Figure 3 shows the accuracy, coverage, and improvement of the mined model over the random predictor for the test set. The accuracy and coverage is the mean value over all proteins. The figure shows that the amino acids have some information that can be used to predict contacts versus non-contacts, but this information is not too good. The figure plots the accuracy and coverage as percentages. It also plots the improvement of the model over the random predictor. The x-axis shows the *prediction factor*, which is related to the γ value (used to predict the top fraction of pairs as contacts). The prediction factor is in multiples of N^*_{tc}, the number of true contacts in the protein with positive contact evidence. For example, a value of 10 means that the top $(10 \times N^*_{tc})/N^*_a$ fraction of the examples are predicted as contacts.

The left-most graph in Figure 3 shows the accuracy and coverage of the predictor over test proteins of all lengths. The other two figures on the right show how accuracy and coverage change with protein length. We have divided the test proteins into four bins: $1 \leq N < 100$, $100 \leq N < 170$, $170 \leq N < 300$, and $300 \leq N$.

We find that over all proteins the amino acids by themselves can be used to give an 8.5% accuracy, 1.5% coverage, and an improvement over a random predictor by a factor of 2.4. Note also the interesting trend in the graph. As the prediction factor increases we get better and better coverage, but the accuracy trails off. This represents the classic accuracy versus coverage trade-off common to many prediction problems. Which value to choose for the prediction factor depends on what is more important. It has been reported in [16] that the 3D structure of proteins can be recovered quite robustly, even from corrupted contacts maps. This implies that coverage should have an higher weight than accuracy. In any case, if we had to choose a value representing the best trade-off, we can pick the point where the accuracy and coverage curves intersect. This happens

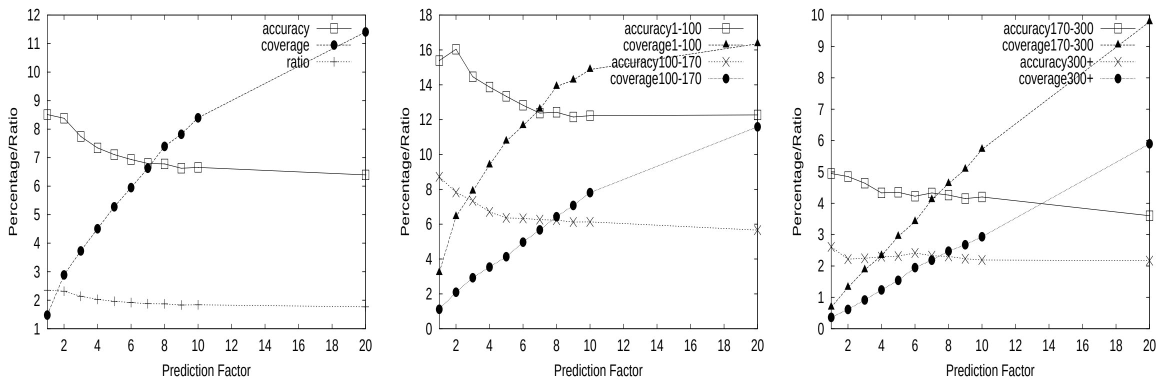

Figure 3: Amino Acids Only

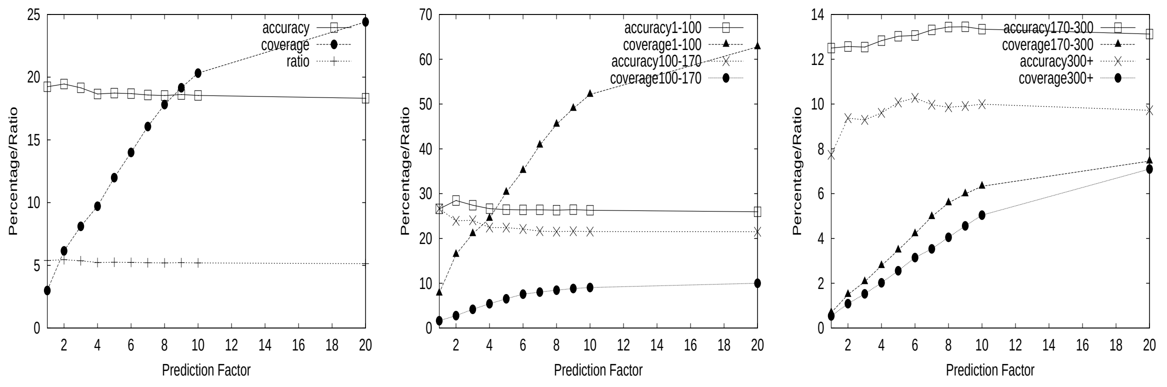

Figure 4: HMMSTR States and Amino Acids

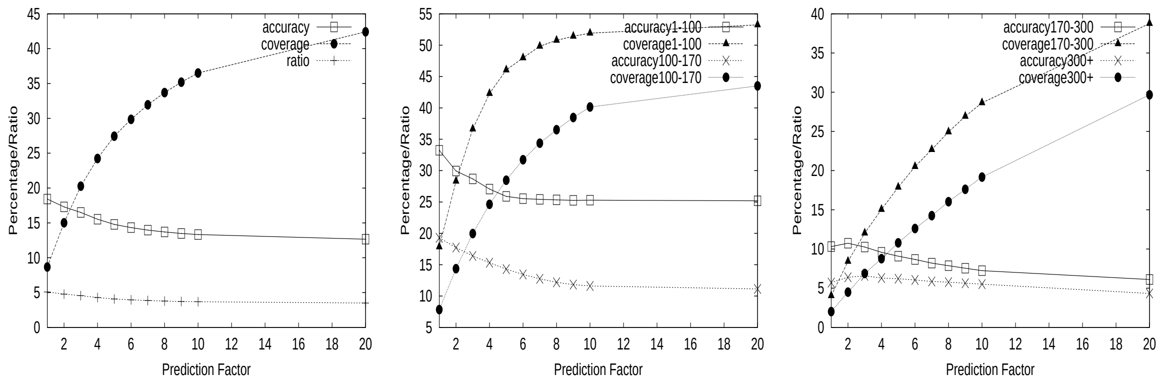

Figure 5: HMMSTR States, Amino acids, and R_t, D_t, C_t Symbols

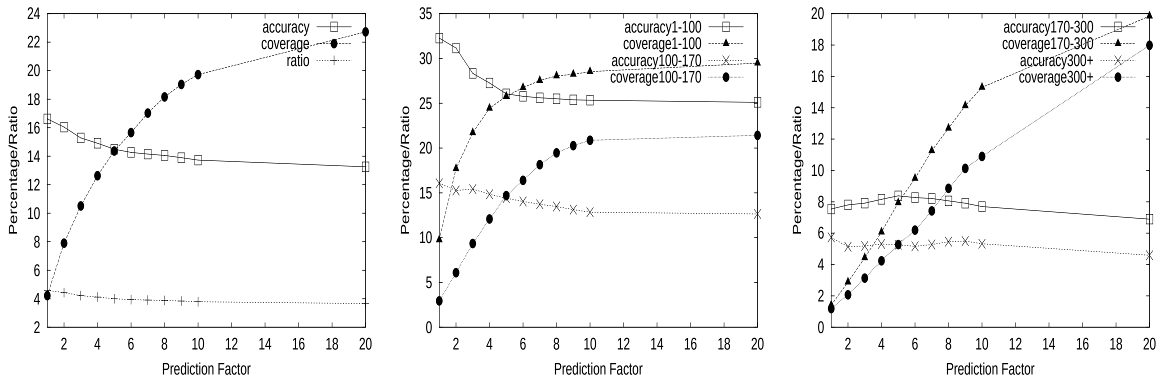

Figure 6: HMMSTR States, Amino Acids, and Amino Acid Profiles

for a prediction factor of 7, where we have roughly 7% accuracy and coverage, and which is 2 times better than random. For 6.3% accuracy we can increase coverage to 14%.

When we consider the results for proteins of different lengths, we find the same trade-off between accuracy and coverage. Looking at the crossover point, we get around 13% accuracy and coverage for short proteins with $N < 100$, 6% for $100 \leq N < 170$, 4.5% for $170 \leq N < 300$, and around 2% of longer proteins.

HMMSTR States and Amino Acids We next added the HMMSTR states corresponding to a_i and a_j, i.e., we added the columns $q_{i_1}, q_{i_2}, \cdots$ and $q_{j_1}, q_{j_2}, \cdots$ to the training and testing sets. Figure 4 shows the results. If we look at the cross-over point we get almost 19% accuracy and coverage, while the model remains 5.2 times better than random. For 18% accuracy we can get coverage of 25% (still 5.1 times better than random). Figure 7 shows the results in a slightly different format. It plots the improvement in coverage/accuracy over a random prediction. These results are comparable to or better than the results recently reported in [23], where they examined pairwise amino acid interactions in the context of secondary structural environment (helix, strand, and coil), and used the environment dependent contact energies for contact prediction experiments. For about 25% coverage our model does more than 5 times the random predictor, as compared to the 4 times improvement reported in [23]. Figure 8 shows the predicted contact map for the protein $2igd$ that

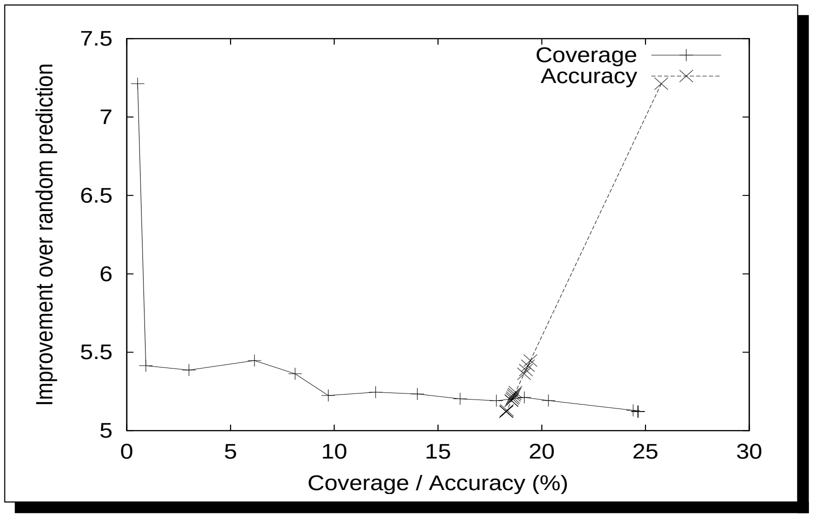

Figure 7: Improvement over Random Prediction

we used in the introduction. We got 35% accuracy and 37% coverage for this protein. The figure shows the true contacts, the contacts correctly predicted, and all the contacts predicted (correctly or incorrectly).

If we look at proteins of various lengths in Figure 4, we find that for $N < 100$, we get 26% accuracy and 63% coverage at the extreme point (4 times over random). For $100 \leq N < 170$ we get 21.5% accuracy and 10% coverage towards the end (6 times over random), for $170 \leq N < 300$ we get 13% accuracy and around 7.5% coverage (6.5 times over random), and for longer proteins we get 9.7% accuracy and 7.5% coverage (7.8 times over random). We believe these results are the best, or at least comparable to those reported so far in the literature on contact map prediction [6, 12]. For example, Fariselli and Casadio [6], used a Neural Network based approach

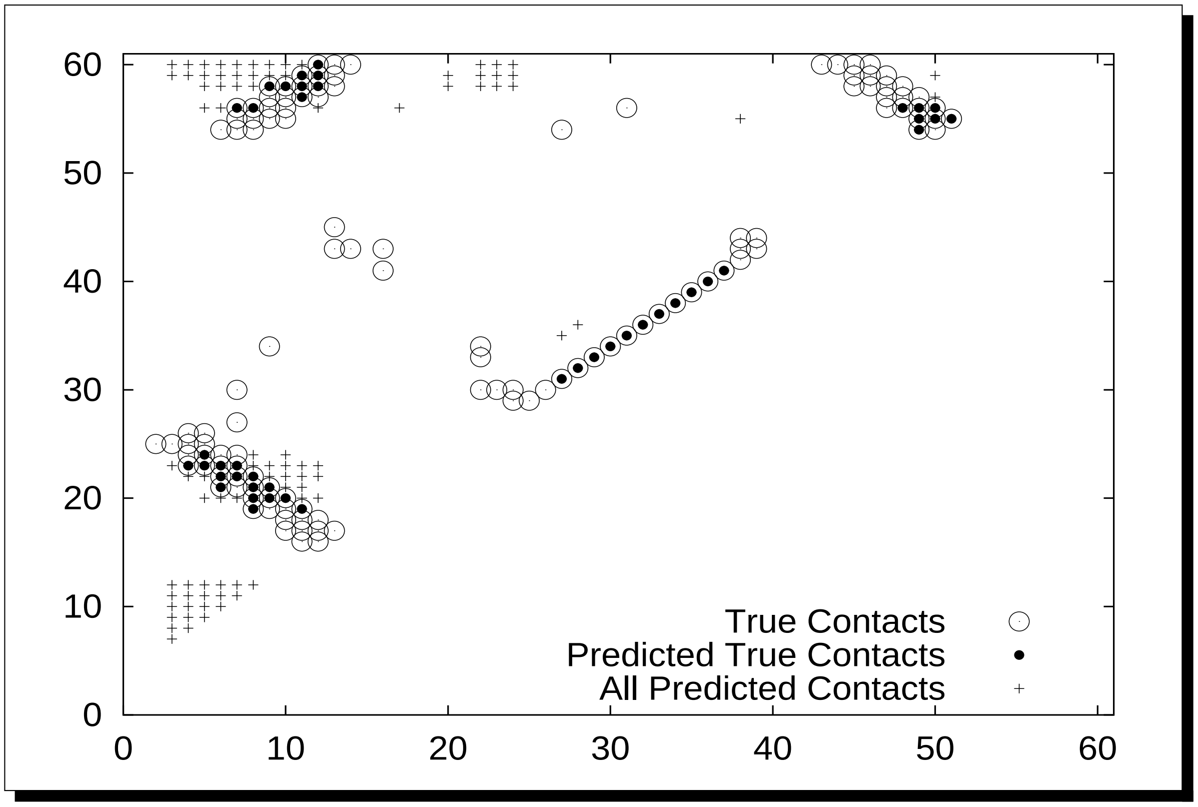

Figure 8: Predicted Contact Map (PDB file 2igd, $N = 61$)

over pairs database, with other contextual information like sequence context windows, amino acid profiles, and hydrophobicity values. They reported an 14.4% accuracy over all proteins, with an 5.4 times improvement over random. They also got 18% accuracy for short proteins with an 3.1 times improvement over random. Olmea and Valencia [12] on the other hand used correlated mutations in multiple sequence alignments for contact map prediction. They added other information like sequence conservation, alignment stability, contact occupancy, etc. to improve the accuracy. They reported 26% accuracy for short proteins, but they did not report the result for all proteins. While we believe that our hybrid approach does better, we should say that direct comparison is not possible, since previous works used a different (and smaller) PDB_select database for training and testing. One draw back of these previous approaches is that they do not report any coverage values, so it is not clear what percentage of contacts are correctly predicted. Another approach to contact map prediction was presented in [15], which was based on correlated mutations. They obtained an accuracy of 13% or 5 times better than random.

Adding Additional Information We next tried to add more columns to the training database. For example we separately added the amino acid profiles, and the structural context symbols for the 3-state secondary structure D_t, discrete backbone angle region R_t, and the context symbol C_t. The results for these cases are shown in Figure 5 and Figure 6. As we can see adding the profiles did not add any additional prediction power to our model, while adding the structural symbols had a positive (somewhat mixed) effect on accuracy and coverage. It appears that while the accuracy of the prediction drops a little there is tremendous boost in the coverage of the model. For example at around 18% accuracy we get about 25% coverage using the HMM states and amino acids (see Figure 4), but when we add the structural symbols, we get about 44% converage for an accuracy of 12.5%. What this tells us is that the structural symbols can be helpful in providing the right context for the predictions and thus help in identifying a larger portion of the contacts.

In conclusion we have presented a new hybrid HMM and association rule mining method for contact predictions. Our results are the best or comparable to those previously reported. We are currently working to further improve both accuracy and coverage by carefully selecting many of the threshold parameters used in the experiments, as well as by adding additional attributes that might help prediction.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996.

[2] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17), 3389-402, 1997.

[3] C. Bystroff and D. Baker. Prediction of local structure in proteins using a library of sequence-structure motifs. *Journal of Molecular Biology*, 281(3), 565-77, 1998.

[4] C. Bystroff, V. Thorsson, and D. Baker. HMMSTR: A hidden markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, (to appear), 2000.

[5] S. Eddy. Profile hidden markov models. *Bioinformatics*, 14(9), 755-63, 1998.

[6] P. Fariselli and R. Casadio. A neural network based predictor of residue contacts in proteins. *Protein Engineering*, 12(1), 15-21, 1999.

[7] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.

[8] K. Han and D. Baker. Global properties of the mapping between local amino acid sequence and local structure in proteins. *Proc. Natl. Acad. Sci. U S A*, 93(12), 5814-5818, 1996.

[9] U. Hobohm and C. Sander. Enlarged representative set of protein structures. *Protein Science*, 3(3), 522-524, 1994.

[10] B. Honig. Protein folding: from the levinthal paradox to structure prediction. *Journal of Molecular Biology*, 293(2), 283-93, 1999.

[11] W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22, 2577-2637, 1983.

[12] O. Olmea and A. Valencia. Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Folding & Design*, 2, S25-S32, June 1997.

[13] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257-86, 1989.

[14] L. Serrano, A. Matouschek, and A.R. Fersht. The folding of an enzyme. III. Structure of the transition state for unfolding of barnase analysed by a protein engineering procedure. *Journal of Molecular Biology*, 224(3), 805-18, 1992.

[15] D. Thomas, G. Casari, and C. Sander. The prediction of protein contacts from multiple sequence aligments. *Protein Engineering*, 9(11):941-48, 1996.

[16] M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. *Folding & Design*, 2(5), 295-306, September 1997.

[17] Y. I. Wolf, N. V. Grishin, and E. V. Koonin. Estimating the number of protein folds and families from complete genome data. *Journal of Molecular Biology*, 299(4), 897-905, 2000.

[18] J. Wootton and S. Federhen. Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.*, 266, 554-71, 1996.

[19] Q. Yi, C. Bystroff, P. Rajagopal, R. E. Klevit, and D. Baker. Prediction and structural characterization of an independently folding substructure in the src sh3 domain. *Journal of Molecular Biology*, 283(1), 293-300, 1998.

[20] M. J. Zaki. Generating non-redundant association rules. In *6th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, August 2000.

[21] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372-390, May-June 2000.

[22] M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed association rule mining. Technical Report 99-10, Computer Science Dept., Rensselaer Polytechnic Institute, October 1999.

[23] C. Zhao and S.-H. Kim. Environment-dependent residue contact energies for proteins. *Proc. Natl. Acad. Sci. USA*, 97(6), 2550-5, 2000.