Modeling protein families using probabilistic suffix trees

Gill Bejerano and Golan Yona*

Institute of Computer Science, Hebrew University, Jerusalem 91904, Israel.

Abstract

We present a method for modeling protein families by means of probabilistic suffix trees (PSTs). The method is based on identifying significant patterns in a set of related protein sequences. The input sequences do not need to be aligned, nor is delineation of domain boundaries required. The method is automatic, and can be applied, without assuming any preliminary biological information, with surprising success. Incorporating basic biological considerations such as amino acid background probabilities, and amino acids substitution probabilities can improve the performance in some cases.

The PST can serve as a predictive tool for protein sequence classification, and for detecting conserved patterns (possibly functionally or structurally important) within protein sequences. The method was tested on one of the state of the art databases of protein families, namely, the Pfam database of HMMs, with satisfactory performance.

1 Introduction

In the last few years there is growing effort to organize the known protein sequences into families and establish databases of protein motifs and domains. These databases have become an important tool in the analysis of newly discovered protein sequences. The biological knowledge accumulated in these databases in the form of consensus patterns, profiles and HMMs helps to detect patterns of biological significance in new protein sequences. In many cases, such discoveries can lead to the assignment of the new sequences to the existing protein families. These attempts are extremely important in view of the large number of newly sequenced proteins which await analysis.

Generally, the existing approaches can be divided into those based on short conserved motifs (e.g. [Bairoch et al. 1997, Attwood et al. 1998, Henikoff & Henikoff 1991]) and those which are based on whole domains (e.g. [Sonnhammer & Kahn 1994, Sonnhammer et al. 1997]). The manually defined patterns in PROSITE have served as an excellent seed for several such works. The methods used to represent these motifs and domains vary, and among the most popular forms are the consensus patterns [Bairoch et al. 1997, Attwood et al. 1998], the position-specific scoring matrices (profiles) [Gribskov et al. 1987, Henikoff & Henikoff 1991] and the HMMs [Krogh et al. 1996]. These forms differ in their mathematical complexity, as well as in their sensitivity/selectivity.

To model a motif, a domain or a protein family, many approaches start by building a multiple alignment. The prerequisite of a valid alignment of the input sequences is a major drawback of this strategy, since building a multiple alignment of many sequences is not an easy task. Current methods for multiple alignment apply heuristic algorithms which are not guaranteed to find the optimal alignment. Moreover, the optimal alignment itself is not guaranteed to be biologically accurate. When the related sequences are diverged and the common pattern cannot be easily distinguished

*corresponding author. email: golany@cs.huji.ac.il

from noise, or when the number of samples used to build the alignment is small then the validity of the automatically generated alignment is questionable, and the resulting model (consensus pattern, profile, HMM) may be of poor quality. Delineation of domain boundaries makes the problem even more difficult. Therefore, fully automatic methods, based on multiple alignments are not guaranteed to be of high quality for classification and pattern detection. Obviously, manually constructed alignments are preferred (PROSITE [Bairoch et al. 1997] serves as an excellent example. Another example is the Pfam database [Sonnhammer et al. 1997] that uses a semi-automatic procedure, which combines manual analysis with automatic tools). However, the sheer size of current databases calls for the development of high performance automated methods.

Other approaches which do not require prealigned sequences were tested as well [Hanke et al. 1996, Neuwald et al. 1994, Jonassen et al. 1995, Suyama et al. 1995], but most of them are limited to short patterns with limited flexibility (i.e. some constraints are imposed on the pattern). Such techniques were used to construct the blocks database [Henikoff & Henikoff 1991], where motifs are defined based on blocks (ungapped short regions of aligned amino acids) extracted from groups of related proteins, and each family is associated with a set of blocks.

Here we present an alternative approach for detecting significant patterns, based on probabilistic suffix trees (PSTs), originally introduced in [Ron et al. 1994]. The model draws on identifying significant appearances of short segments among many of the input protein sequences, regardless of the relative positions of these segments within the different proteins. These significant segments reflect some statistical properties of the corresponding protein family. Practically, they induce a probability distribution on the *next* symbol to appear right after the segment. Although probably no simple underlying statistical model exists for protein families, this property implies a certain feature which is common to a large subset of them. That feature, termed **short memory**, which is common to many natural sources, indicates that for a certain sequence the (empirical) probability distribution of the next symbol given the preceding subsequence can be quite accurately approximated by observing no more than the last L symbols in that subsequence.

This observation has led before to the suggestion of modeling many natural sources by use of order L Markov chains (L being the memory length of the model), or by the more extensive family of Hidden Markov Models (HMMs) which is more complex and allows to capture subtler features of the original source. These two families, although being able to model rich sources in an efficient manner, and lending themselves to extensive mathematical treatment, have critical drawbacks for practical use. The Markov chain model suffers from exponential growth in the number of states for a non trivial memory length, and poor source approximation at low order. The HMMs family suffers from known learnability hardness results [Abe & Warmuth 1992, Gillman & Sipser 1994], and in practise, a high quality multiple alignment of the input sequences is required to obtain a reliable model. The probabilistic suffix trees are inspired by the same reasoning. However, this family can model rich sources with high efficiency using a reasonable amount of memory [Ron et al. 1994]. The strength of the model stems from its memory length which is context dependent as observed for many natural sources.

The method is simple to apply. It does not require the input sequences to be aligned first, nor is it limited to very short patterns. Despite its simplicity, the approach is surprisingly powerful. In the next sections we first describe the model, and then demonstrate its power by applying it to known protein families.

2 Theory

The definitions and the subsequently implemented algorithm are a variant on the learning algorithm originally presented in [Ron et al. 1994].
A PST over an alphabet is a non empty tree, whose nodes vary in degree between zero (for leaves) and the size of the alphabet. Each edge in the tree is labeled by a single symbol of the alphabet, such that no symbol is represented by more than one edge branching out of any single node (hence the degree of each node is bounded by the size of the alphabet). Nodes of the tree are labeled by a string, which is the one generated by walking **up** the tree from that node to the root. Each node is assigned a probability distribution vector over the alphabet. When the PST is used to predict significant patterns within a query string, this probability distribution vector comes into play. It corresponds to the probabilities the tree assigns to the next query symbol, given that the longest suffix of preceding query symbols found in the tree matches that particular node's label. An example of a PST is given in Fig. 1.

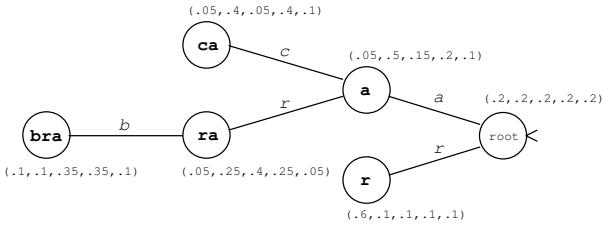

Figure 1: **An example of a PST over the alphabet** $\Sigma = a, b, c, d, r$. The tree is shown in landscape mode, which makes the prediction step more intuitive. The root is the rightmost node. The vector that appears near each node is the probability distribution over the next symbol. For example, the probability distribution associated with the substring ra is 0.05, 0.25, 0.4, 0.25 and 0.05 for the symbol a,b,c,d and r respectively (e.g. the probability to observe c after a substring, whose largest suffix in the tree is ra, is 0.4).

2.1 Definitions

Let Σ be the alphabet (e.g. the alphabet of 20 amino acids for protein sequences, or 4 nucleic acids for DNA sequences), and let $r^1, r^2, ..., r^m$ be the sample set of m strings over the alphabet Σ, where the length of the ith ($i = 1..m$) string is l_i (i.e. $r^i = r^i_1 r^i_2 ... r^i_{l_i} \quad r^i_j \in \Sigma$).

First we define the empirical probability of a substring over the given sample set, which is the number of times that substring was observed in the sample set divided by the maximal number of (possibly overlapping) occurrences it could have had, considering the sample size. Formally speaking, given a string s of length l ($s = s_1 s_2 ... s_l$) we define a set of variables

$$\chi^{i,j}_s = \begin{cases} 1 & \text{if } s_1 s_2 ... s_l = r^i_j r^i_{j+1} ... r^i_{j+(l-1)} \\ 0 & \text{otherwise} \end{cases}$$

for each $i = 1..m$ and $j = 1..l_i - (l-1)$. These indicator variables have a value of one if and only if the string s is a substring of r^i starting at position j (see Fig. 2).

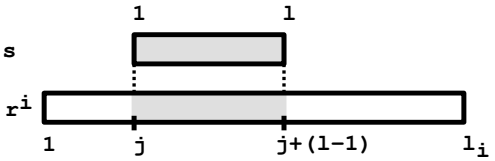

Figure 2: **A match of s and r^i at position j.** The indicator variable is set to one: $\chi^{i,j}_s = 1$.

The number of (possibly overlapping) occurrences of the string s in the string set $\{r^i\}$ is given by

$$\chi_s = \sum_{i,j} \chi_s^{i,j}$$

The total number of possible occurrences (with overlaps) of s (or any other string with the same length as s) is

$$N_s = \sum_{i \ \ s.t. \ \ l_i \geq l} (l_i - (l-1))$$

Therefore, the empirical probability of the string s is simply given by

$$\tilde{P}(s) = \frac{\chi_s}{N_s}$$

We go on to define the conditional empirical probability of observing a symbol right after a given substring as the number of times that symbol has shown up right after the given substring divided by the total number of times that substring has shown up at all, followed by any symbol. Specifically, let χ_{s*} be the number of non-suffix occurrences of the string s in the string set $\{r^i\}$, i.e.

$$\chi_{s*} = \sum_{\sigma' \in \Sigma} \chi_{s\sigma'}$$

Then the conditional empirical probability of observing the symbol σ right after the string s is defined by

$$\tilde{P}(\sigma|s) = \frac{\chi_{s\sigma}}{\chi_{s*}}$$

Finally, we define $suf(s) = s_2 s_3 ... s_l$.

2.2 Building the PST

First, we define L to be the memory length of the PST (i.e. the maximal length of a possible string in the tree).

We work our way gradually through the space of all possible substrings of lengths 1 through L, starting at single letter substrings, and abstaining from further extending a substring whenever its empirical probability has gone below a certain threshold (P_{min}), or on having reached the maximal L length boundary. The P_{min} cutoff avoids an exponentially large search space.

At the beginning of the search we hold a PST consisting of a single root node. Then, for each substring we decide to examine, we check whether there is some symbol in the alphabet for which the empirical probability of observing that symbol right after the given substring is non negligible, and is also significantly different from the empirical probability of observing that same symbol right after the string obtained from deleting the leftmost letter from our substring[1]. Whenever these two conditions hold, the substring, and all necessary nodes on its path, are added to our PST.

The reason for the two step pruning (first defining all nodes to be examined, then going over each and every one of them) stems from the nature of PSTs. A leaf in a PST is deemed useless if its prediction function is identical (or almost identical) to its parent node. However, this in itself is no reason not to examine its sons further while searching for significant patterns. Therefore, it may, and does happen that consecutive inner PST nodes are almost identical.

[1] This string corresponds to the label of the direct father of the node we are currently examining (note that the father node has **not** necessarily been added itself to the PST at this time)

Finally the resulting PST skeleton is first added the node prediction functions, using the appropriate conditional empirical probability, and then these probabilities are smoothed using a standard technique so that no single symbol is absolutely impossible right after any given substring (even though the empirical counts may attest differently).

We now present the procedure for building a PST out of a sample set, based on [Ron et al. 1994]. The procedure uses 5 external parameters: L the memory length, P_{min} the minimal probability with which strings are required to occur, r which is a simple measure of the difference between the prediction of the candidate at hand and its direct father node, γ_{min} the smoothing factor and α.

We use $\bar{T}$ to denote the tree, $\bar{S}$ to denote the set of (unique) strings which we need to check, and $\bar{\gamma}_s$ to denote the probability distribution (over the next symbol) associated with the node s.

The algorithm: **Build-PST** (P_{min}, α, γ_{min}, r, L)

1. **Initialization:** let $\bar{T}$ consist of a single root node (with an empty label), and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma \quad and \quad \tilde{P}(\sigma) \geq P_{min}\}$.
2. **Building the PST skeleton:** While $\bar{S} \neq \emptyset$, pick any $s \in \bar{S}$ and do:
   (a) Remove s from $\bar{S}$
   (b) If there exists a symbol $\sigma \in \Sigma$ such that

$$\tilde{P}(\sigma|s) \geq (1+\alpha)\gamma_{min} \quad and \quad \frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suf(s))} \begin{cases} \geq r \\ or \\ \leq 1/r \end{cases}$$

   then add to $\bar{T}$ the node corresponding to s and all the nodes on the path to s from the deepest node in $\bar{T}$ that is a suffix of s.
   (c) If $|s| < L$ then for every $\sigma' \in \Sigma$, if $\tilde{P}(\sigma' s) \geq P_{min}$, then add $\sigma' s$ to $\bar{S}$.
3. **Smoothing the prediction probabilities:** For each s labeling a node in $\bar{T}$, let

$$\bar{\gamma}_s(\sigma) \equiv (1 - |\Sigma|\gamma_{min})\tilde{P}(\sigma|s) + \gamma_{min}$$

The final step (step 3) of the learning algorithm is the smoothing process, which assures that no symbol is predicted to have a zero probability, no matter what suffix is observed before it. The value of γ_{min} defines the minimum probability for a symbol, and the empirical probabilities should be adjusted to satisfy this requirement. This is done by decreasing the empirical probabilities, such that a total of $|\Sigma|\gamma_{min}$ is "collected", to be later shared by all symbols. The decrement of each empirical probability is done in proportion to its value.

2.3 Prediction using a PST

Given a string s its prediction by a certain PST is done letter by letter, where the probability of each letter is calculated by scanning the tree in search of the longest suffix that appears in the tree and ends just before that letter. The conditional probability of this letter given this suffix is given by the probability distribution associated with the corresponding node in the PST.

For example, to predict the string $s = abracadabra$ with the PST given in Fig. 1 the following procedure is carried out:

$$
\begin{aligned}
P^T(s) &= P^T(abracadabra) \\
&= P^T(a)P^T(b|a)P^T(r|ab)P^T(a|abr)P^T(c|abra)P^T(a|abrac)\cdots P^T(a|abracadabr) \\
&= \bar{\gamma}_{root}(a)\ \bar{\gamma}_a(b)\ \bar{\gamma}_{root}(r)\ \bar{\gamma}_r(a)\ \bar{\gamma}_{bra}(c)\ \bar{\gamma}_{root}(a)\cdots\bar{\gamma}_r(a) \\
&= 0.2\cdot 0.5\cdot 0.2\cdot 0.6\cdot 0.35\cdot 0.2\cdot 0.4\cdot 0.2\cdot 0.5\cdot 0.2\cdot 0.6 = 4.032\cdot 10^{-6}
\end{aligned}
$$

Note that $-\log_2 P^T(s) \simeq 17.92$ bit (1.63 per symbol) while $-\log_2(0.2^{11}) \simeq 25.54$ bit (2.32 per symbol).

3 Results

To test our approach, a PST was created for each family in the Pfam database [Sonnhammer et al. 1997] release 1.0. This database contains 175 protein families[2] derived from the SWISSPROT 33 database [Bairoch & Boeckman 1992]. Each family was divided into a training set and a test set, in ratio 4:1, such that 4/5 of the family members are in the training set, and the PST was trained on the training set. Then, for each sequence in the database we calculated its probability as predicted by the PST. To avoid bias due to length differences, the probability is normalized by the length, and each sequence is reported along with its probability per letter. Hits are sorted by decreasing probability.

The quality of the PST model is estimated by applying the "equivalence number" criterion [Pearson 1995]. The equivalence number criterion set a threshold at the point where the number of false positives equals the number of false negatives, i.e. it is the point of balance between selectivity and sensitivity (we use the term *iso-point* to denote this point). The quality of the model is measured by the number of hits above this threshold which belong to the family (i.e. hits which are considered successfully detected) relative to the total number of proteins in the family. The results for the 170 protein families in the Pfam database release 1.0, with more than 10 members, are given in table 1 (see appendix A).

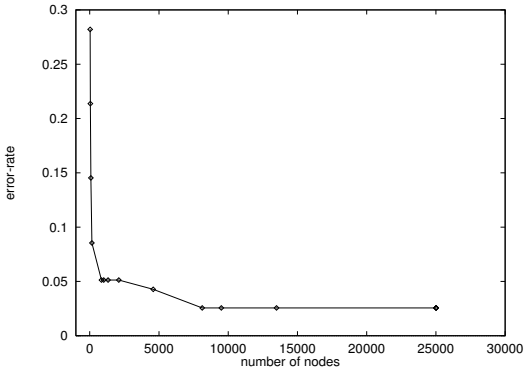

Figure 3: **Improving prediction by increasing the number of nodes.** A PST was built for the glyceraldehyde 3-phosphate dehydrogenases family, for different values of P_{min}, and the quality of the PST was estimated by applying the equivalence number criterion (see text). The graph plots the error rate (the number of family members which were not identified, i.e. their score was below the threshold set by the iso-point) vs. the number of nodes in the PST (which increases as P_{min} decreases). Note that the error rate decreases as the number of nodes increases. At some value of P_{min} the quality does not improve much, while the tree keeps growing. If the results are satisfactory then it is suggested to stop the incremental process at this point.

[2]The Pfam database uses a semi-automatic procedure, which combines manual analysis and automatic tools to build multiple alignments for groups of related proteins. Each multiple alignment is closely monitored to avoid misalignments, and HMMs are derived from these alignments. The HMMs can be used for searching close relatives in the database, and the most significant hits are included in the corresponding family. The process iterates until it converges, while being traced manually.

It should be noted that table 1 only demonstrates the potential of the PST model, but must not be taken as an upper bound on its performance. To obtain these results we simply ran the PST learning procedure with a fixed set of parameters for all families which we found to result in good performance in reasonable running time. However, the performance of a PST can be improved by simply tuning the values of the parameters, either globally or per each family. One can either decide to examine more nodes (lower P_{min}), or lower the criteria of acceptance of a candidate (lower α or lower r) or even deepen the tree (increase L). This can be done in an efficient, incremental manner. This is demonstrated in Fig. 3 for the glyceraldehyde 3-phosphate dehydrogenases family. Adding more nodes would tend to increase sensitivity **without** decreasing its selectivity, simply because more (longer) substrings that are observed in the training set are "recorded" in the PST. This means that only leaves are further expanded, while the probability distributions of internal nodes are not affected. This way, the predictions over the test set are refined. However, since long substrings observed in the training set are not expected to occur in unrelated sequences, the prediction of unrelated sequences is based on short substrings corresponding to internal nodes close to the root, and therefore is not expected to change.

The only limitation one should keep in mind is that the size of the tree (the number of nodes), as well as the running time, may increase drastically as the parameters are refined, while the improvement in the quality may not be significant. However, if computation time is of no concern then the PST can run as long as the quality improves (i.e. more family members are detected above the iso-point). In two cases no further improvement is expected: 1) when all sequences of the family are detected 2) when all strings in the training set are exhaustively recorded, and predicted with very high probability.

An additional improvement is expected if a larger sample set is used to train the PST. Currently, the PST is built from the training set **alone**. Obviously, training the PST on **all** strings of a family (as Pfam database is created) would improve its prediction as well.

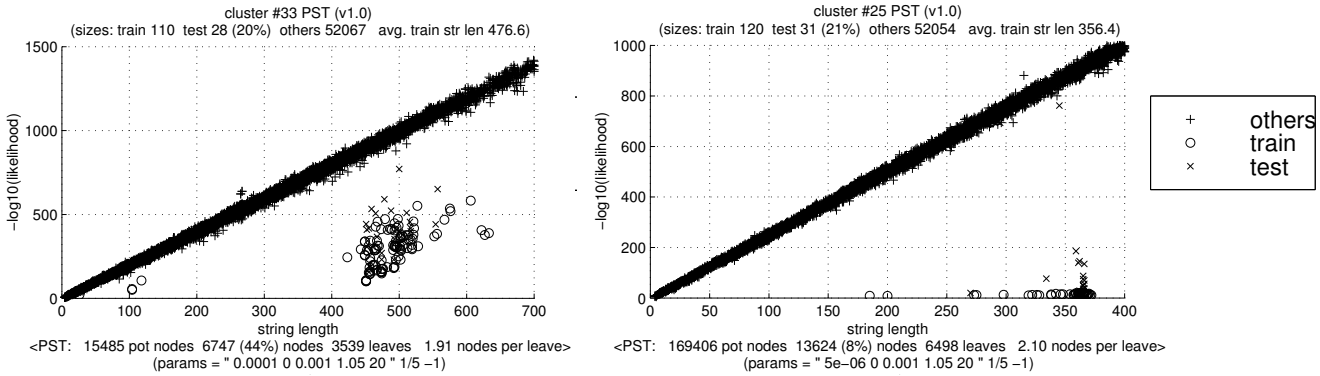

Figure 4: **Left: Performance of a PST of the Neurotransmitter-gated ion-channels** ($P_{min} = 0.0001$)**. Right: Performance of a PST of the MHC I family** ($P_{min} = 0.000005$)**.** The latter, being trained with a lower value of P_{min}, is an example of an extreme fit of the PST to the training set. Note that the prediction of the test set improves as well, while the unrelated sequences are left unaffected. When the PST was trained with the same set of parameters as the PST on the left, its performance graph resembled that of the graph on the left.

To demonstrate the performance of this model, two examples are given in Fig. 4. The probabilities (as predicted by the PST) of the training set sequences, the test set sequences and all the other sequences in the database are plotted vs. the sequences length. Note that the unrelated sequences show a clear linear relation in log scale. The training set and the test set samples are located far

below this line.

The PST can also be used to predict which segments of a given query sequence are suspected to be functionally or structurally important. These segments correspond to regions of high probability. This is demonstrated in Fig. 5. Note that along these regions not all letters are essentially of high probability, which may suggest a substitution or a gap in the query sequence. Recall that the PST does not require the input sequences to be aligned nor does it make use of such information during the learning process. The input sequences were not fragmented according to domain boundaries before the learning phase, and therefore this information was solely self attained. In this view, the PST model can also help to guide a multiple alignment of a set of related sequences, by suggesting an initial seed, which is based on the regions of high probability.

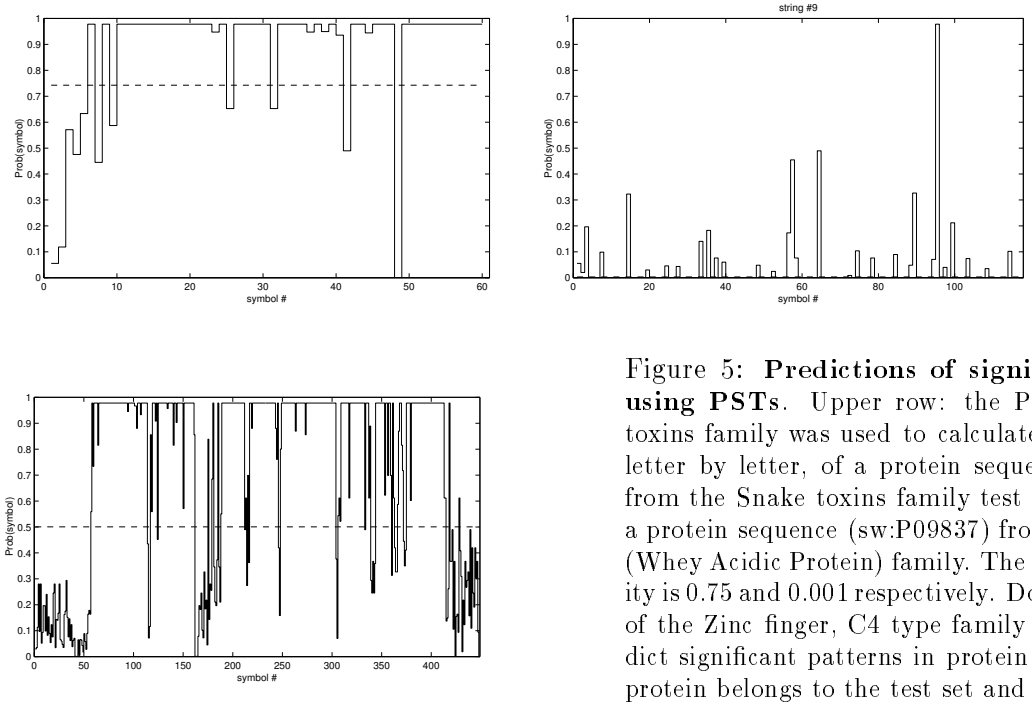

Figure 5: **Predictions of significant patterns using PSTs**. Upper row: the PST of the Snake toxins family was used to calculate the probability, letter by letter, of a protein sequence (sw:P01445) from the Snake toxins family test set (left), and of a protein sequence (sw:P09837) from the WAP-type (Whey Acidic Protein) family. The average probability is 0.75 and 0.001 respectively. Down left: the PST of the Zinc finger, C4 type family was used to predict significant patterns in protein sw:P10826. The protein belongs to the test set and its average probability per symbol is 0.5.

The performance of the PSTs (i.e. predicting the correct assignments) is currently compared to the performance of the corresponding HMMs of the Pfam database[3] using the hmmsearch program which is a part of the HMMER package[4]. However, while the prediction of PST takes a few minutes for all sequences in the database, searching the SWISSPROT database with a typical HMM may take about two hours on a SGI workstation R4000. The results of these calculations will be made available soon.

4 Adding biological considerations to the PST model

The basic PST model does not require any assumption on the input data, nor does it utilize any a-priori information we may have about the data. Incorporating such information may improve the

[3]The HMMs and the (manually adjusted) multiple alignments which were used to generate them are generously available through the Pfam homepage http://www.sanger.ac.uk/Pfam/

[4]This extremely useful software is also generously available through the web, by Sean Eddy's lab, at http://hmmer.wustl.edu/

performance of this model, since it adjusts the general purpose model to the specific problem of identifying significant patterns in macromolecules, where some characteristic features and processes are observed. Specifically, the information we would like to consider is the amino acids background probabilities, and the amino acids substitution probabilities. These distributions are integrated into the PST model and some changes are made to account for this a-priori information.

Few additional definitions are needed here: Let P_0 be the a-priori probability distribution over the alphabet Σ and let q_{ab} be the probability that amino acid a is replaced by amino acid b ($a, b \in \Sigma$). In the new procedure, only strings that their prediction of the next symbol differs from what expected simply by chance (the background probabilities) are included in the tree.

Also, define $\bar{\chi}_s$ as the number of *distinct* sequences which include the substring s. This number is required to exceed a certain threshold N_{min}, which is defined in proportion to the total number m of strings in the sample set (i.e. $N_{min} = c \cdot m$ where c is chosen to be, say, 0.2 so that the substring s is required to exist in at least 20% of the member proteins)

Finally, the last step (step 3) of the learning algorithm (the smoothing step), is modified, and it is now based on a position-based pseudo-counts method (similarly to the method suggested by [Henikoff & Henikoff 1996]). This method adds hypothetical sequences to the sample set in order to avoid zero probabilities which are due to undersampling. The number and the distribution of pseudo-counts is "position-based" (i.e. different for each node) and takes into account the *diversity*, i.e. the number of different amino acid types observed after the corresponding substring, as well as the counts of actually observed amino acids.

For a node represented by the substring s, denote by R_s the diversity at s, i.e. the number of different amino acids observed after s.

$$R_s = |\{\sigma \mid \chi_{s\sigma} > 0\}|$$

Denote by B_s the total number of pseudo-counts added for the node s. As suggested in [Henikoff & Henikoff 1996] B_s is set to $B_s = \mu R_s$ where μ can be optimized for best performance[5]. Then, the number of pseudo-counts of amino acid a at this node is given by

$$\begin{aligned} b_a &= B_s \sum_{i=1}^{20} Prob(i|s) \cdot Prob(a|i) \\ &= B_s \sum_{i=1}^{20} \frac{\chi_{si}}{\chi_{s*}} \cdot \frac{q_{ia}}{Q_i} \end{aligned}$$

Where $Q_i = \sum_{k=1}^{20} q_{ik}$. The probability of observing a after the string s is defined as the weighted average of the empirical probability $\tilde{P}(a|s)$ and the a-priori probability as defined by the pseudo-counts, $P_{pseudo}(a|s) = b_a/B_s$.

The new procedure is described next:

The algorithm: **New-Build-PST** (N_{min}, α, r, L)

1. **Initialization:** let $\bar{T}$ consist of a single root node (with an empty label), and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma \quad and \quad \bar{\chi}_\sigma \geq N_{min}\}$.

2. **Building the PST skeleton:** While $\bar{S} \neq \emptyset$, pick any $s \in \bar{S}$ and do:

[5] We haven't optimized this parameter yet. Ad hoc, we used the same value as in [Henikoff & Henikoff 1996]

(a) Remove s from $\bar{S}$

(b) If there exists a symbol $\sigma \in \Sigma$ such that

$$\tilde{P}(\sigma|s) \geq (1+\alpha)P_0(\sigma) \quad and \quad \frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suf(s))} \begin{cases} \geq r \\ or \\ \leq 1/r \end{cases}$$

then add to $\bar{T}$ the node corresponding to s and all the nodes on the path to s from the deepest node in $\bar{T}$ that is a suffix of s.

(c) If $|s| < L$ then for every $\sigma' \in \Sigma$, if $\bar{\chi}_{\sigma' s} \geq N_{min}$ then add $\sigma' s$ to $\bar{S}$.

3. **Smoothing the prediction probabilities:** For each s labeling a node in $\bar{T}$, let

$$\bar{\gamma}_s(\sigma) \equiv \frac{\chi_{s*}}{\chi_{s*}+B_s}\tilde{P}(a|s) + \frac{B_s}{\chi_{s*}+B_s}P_{pseudo}(a|s) = \frac{\chi_{s*}}{\chi_{s*}+B_s}\frac{\chi_{sa}}{\chi_{s*}} + \frac{B_s}{\chi_{s*}+B_s}\frac{b_a}{B_s}$$

When this procedure was applied[6] to 40 Pfam families on which the original PST model performed worst, we found that for 22 families same or better separation was obtained using much smaller trees (for example, the error rate for the Sigma-54 transcription factors improved from 16.1% with 5040 nodes, to 8.9% with 2182 nodes). This is not true for all families, probably since our pruning criteria still need to be refined. The results are yet preliminary. However, this direction seems promising and it is still under study.

5 Discussion

In this paper we presented a new approach for modeling a group of related proteins, without incorporating any prior information about the input sequences. The method applies probabilistic suffix trees to capture some statistical properties of the input family, by recording significant occurrences of substrings of variable lengths. The method induces probability distributions over the next symbols from the empirically observed distributions. Thus, a variable length memory feature is essentially recorded in the tree structure. The PST model is well adapted to the problem at hand in terms of magnitude. Short motifs on the order of 20-30 are well integrated into a simply calibrated learning algorithm whose output is reasonably small (a tree of a few thousands nodes is typically enough for very good discriminative power, while several hundreds already do rather well).

Any new sequence is then compared to the PST model. By accumulating a prediction over the whole length of the query sequence, less conserved regions come into play. Only the balance between possibly several recognizable motifs and unknown regions determines the degree of similarity a query sequence has to the given model. The resulting tree is, as stated above, efficient in prediction, making the query stage after the initial tree building almost immediate, even in terms of comparison with the whole database (of some tens of thousands of sequences). A further speed up in prediction can always be achieved by a one time investment in the straightforward conversion of the tree into a not much larger Probabilistic Finite Automata (PFA), of identical predictions (see [Ron et al. 1994]). The tree structure itself, when examined, can tell by its very nature, what are the different patterns significantly found in the group of sequences from which it was constructed.

[6] The substitution probabilities we used are those from which the blosum62 scoring matrix [Henikoff & Henikoff 1992] was derived

As was demonstrated in section 3 the performance of the model can be improved by simply relaxing the parameters of the learning procedure. This process will eventually result in fitting the model to the training set. However, this does not affect the generalization power of the model in our case (see section 3). Yet, from the perspective of computational learning theory, it may be interesting to define a total cost function which accounts for the quality as well as for the model's complexity, and to grow the PST only until the cost is optimized. Another approach is to build a PST for both the training set and the test set independently, while comparing the induced probability distributions. When either model is (over)fitted to the noise and bias in the corresponding sample set, the distance (e.g. KL-divergence) between these distributions is expected to increase and the generalization power of the model is not expected to improve further (work in progress).

The model was applied to all protein families in the Pfam database, and the results show that the model can predict and identify the other members of the protein family with surprising success when compared with the state of the art in family modeling. The method does not assume any further biological information, but such information can be easily incorporated to improve its sensitivity. Indeed, when biological knowledge is taken into account (such as the amino acid background probabilities and the substitution probabilities) similar or better separation is achieved in some cases, using smaller trees. Further improvements should take into account the possibility of gaps, and generalization of nodes to account for more complex patterns (e.g. regular expressions), and are currently under consideration.

6 Acknowledgments

We thank Dana Ron for valuable discussions and Naftali Tishby for helpful suggestions. We also thank Amnon Barak for making the MOSIX parallel system available for us.

References

[Abe & Warmuth 1992] Abe, N. & Warmuth, M. (1992). On the computational complexity of approximating distributions by probability automata. *Machine Learning* **9,** 205-260.

[Attwood et al. 1998] Attwood, T. K., Beck, M. E., Flower, D. R., Scordis, P. & Selley, J. (1998). The PRINTS protein fingerprint database in its fifth year. *Nucl. Acids Res.* **26,** 304-308.

[Bairoch et al. 1997] Bairoch A., Bucher P., & Hofmann K. (1997). The PROSITE database, its status in 1997. *Nucl. Acids Res.* **25,** 217-221.

[Bairoch & Boeckman 1992] Bairoch, A. & Boeckman, B. (1992). The SWISS-PROT protein sequence data bank. *Nucl. Acids Res.* **20,** 2019-2022.

[Gillman & Sipser 1994] Gillman, D. & Sipser, M. (1994). Inference and minimization of hidden Markov chains. In *Proc. of the Seventh Annual Workshop on Computational Learning Theory* pp 147-158.

[Gribskov et al. 1987] Gribskov, M., Mclachlen, A. D. & Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA* **84,** 4355-4358.

[Hanke et al. 1996] Hanke, J., Beckmann, G., Bork, P. & Reich, J. G. (1996). Self-organizing hierarchic networks for pattern recognition in protein sequence. *Protein Sci.* **5,** 72-82.

[Henikoff & Henikoff 1991] Henikoff, S. & Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. *Nucl. Acids Res.* **19,** 6565-6572.

[Henikoff & Henikoff 1992] Henikoff, S. & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA* **89,** 10915-10919.

[Henikoff & Henikoff 1996] Henikoff, J. G. & Henikoff, S. (1996). Using substitution probabilities to improve position-specific scoring matrices. *Comp. App. Biosci.* **12:2,** 135-143.

[Jonassen et al. 1995] Jonassen, I., Collins, J. F. & Higgins, D. G. (1995). Finding flexible patterns in unaligned protein sequences. *Protein Sci.* **4,** 1587-1595.

[Krogh et al. 1996] Krogh, A., Brown, M., Mian, I. S., Sjölander, K. & Haussler, D. (1996). Hidden Markov models in computational biology: Application to protein modeling. *J. Mol. Biol.* **235,** 1501-1531.

[Neuwald et al. 1994] Neuwald, A. F. & Green, P. (1994). Detecting patterns in protein sequences. *J. Mol. Biol.* **239,** 698-712.

[Pearson 1995] Pearson, W. R. (1995). Comparison of methods for searching protein sequence databases. *Protein Sci.* **4,** 1145-1160.

[Ron et al. 1994] Ron, D., Singer, Y. & Tishby, N. (1996). The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning* **25** 117-150.

[Sonnhammer & Kahn 1994] Sonnhammer, E. L. L. & Kahn, D. (1994). Modular arrangement of proteins as inferred from analysis of homology. *Protein Sci.* **3,** 482-492.

[Sonnhammer et al. 1997] Sonnhammer, E. L., Eddy, S. R., Durbin, R. (1997). Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins* **28,** 405-420.

[Suyama et al. 1995] Suyama, M., Nishioka, T. & Oda, J. (1995). Searching for common sequence patterns among distantly related proteins. *Protein Eng.* **8:11,** 1075-1080.

Appendix A - PST performance for all Pfam families

(See next pages)

Family	Size	Coverage	No. sequences missed	error-rate
IG superfamily	884	0.414	51	0.06
Protein kinase	725	0.523	108	0.15
Globin	681	0.974	15	0.02
7 transmembrane receptor (Rhodopsin family)	515	0.707	36	0.07
Homeobox domain	383	0.333	27	0.07
EF hand	320	0.401	25	0.08
Ribulose bisphosphate carboxylase, large chain	311	0.995	4	0.01
Zinc finger, C2H2 type	297	0.362	23	0.08
ABC transporters	269	0.518	44	0.16
Trypsin	246	0.730	22	0.09
Ras family (contains ATP/GTP binding P-loop)	213	0.930	8	0.04
Cytochrome P450	204	0.917	17	0.08
Tubulin	196	0.943	1	0.01
Elongation factor Tu family (contains ATP/GTP binding P-loop)	184	0.802	15	0.08
ATP synthase alpha and beta subunits	180	0.694	6	0.03
Alcohol/other dehydrogenases, short chain type	180	0.661	20	0.11
Core histones H2A, H2B, H3 and H4	178	0.887	6	0.03
Cytochrome c	175	0.891	11	0.06
Snake toxins	172	0.936	4	0.02
Cytochrome b(N-terminal)/b6/petB	170	0.658	3	0.02
EGF-like domain	169	0.133	18	0.11
Heat shock hsp70 proteins	163	0.906	7	0.04
Fibronectin type III domain	161	0.242	23	0.14
4Fe-4S ferredoxins and related iron-sulfur cluster binding domains.	152	0.559	18	0.12
Class I Histocompatibility antigen, domains alpha 1 and 2	151	0.494	3	0.02
Zinc-binding metalloprotease domain	148	0.029	21	0.14
Reverse transcriptase (RNA-dependent DNA polymerase)	147	0.237	17	0.12
Glutathione S-transferases.	144	0.849	14	0.10
Actins	142	0.965	4	0.03
RNA recognition motif. (aka RRM, RBD, or RNP domain)	141	0.353	22	0.16
Intermediate filament proteins	139	0.607	5	0.04
Zinc finger, C4 type (two domains)	139	0.152	6	0.04
Neurotransmitter-gated ion-channel	138	0.882	4	0.03
Src Homology domain 3	137	0.126	16	0.12
Helix-loop-helix DNA-binding domain	133	0.194	7	0.05
Insulin/IGF/Relaxin family	132	0.715	3	0.02
Cytochrome b(C-terminal)/b6/petD	130	0.313	27	0.21
Heat shock hsp20 proteins	129	0.625	7	0.05
Zinc-binding dehydrogenases	129	0.970	6	0.05
Src Homology domain 2	128	0.157	5	0.04
Response regulator receiver domain	128	0.424	19	0.15
Ligand-binding domain domain of nuclear hormone receptors	127	0.313	7	0.06
Phospholipase A2	122	0.938	3	0.02
glyceraldehyde 3-phosphate dehydrogenases	117	0.977	3	0.03
lipocalins	115	0.858	7	0.06
Alpha amylases (family of glycosyl hydrolases)	114	0.750	14	0.12
Protein hormones (family of somatotropin, prolactin and others)	111	0.961	4	0.04
Kazal-type serine protease inhibitor domain	110	0.735	6	0.05
Cytochrome C oxidase subunit II	109	0.897	2	0.02
Sugar (and other) transporters	107	0.880	15	0.14

Table 1: **PST performance for all Pfam families (part 1).** The number of proteins in the family is given in the second column. Coverage (third column) is the total portion of the sequences which was included in the multiple alignment used to define the domain or the family in the Pfam database. Each family was divided into a training set and a test set and the PST was build from the training set. To test the quality of this PST, we calculate the probability that the PST induces on each database sequence, and each family sequence (from the train set **and** the test set) whose probability is above the iso-point is considered successfully detected (see text for more details). The average error-rate is 0.093. The set of parameters used to train the PST is $P_{min} = 0.0001$ $\alpha = 0$ $\gamma_{min} = 0.001$ $r = 1.050$ and $L = 20$. To train a PST on a typical family with this set of parameters it takes about two hours, at the most, on a pentium II 266 Mhz. Additional 5 minutes are needed to predict all sequences in the SWISSPROT database.

Table 1...continued

Family	Size	Coverage	No. sequences missed	error-rate
Lectin C-type domain short and long forms	106	0.478	14	0.13
Zinc finger, CCHC class	105	0.072	12	0.11
E1-E2 ATPases	102	0.636	7	0.07
wnt family of developmental signaling proteins	102	0.936	6	0.06
Bacterial regulatory helix-loop-helix proteins, lysR family	101	0.476	16	0.16
FAD/NAD-binding domain in oxidoreductases	101	0.244	12	0.12
Ribulose bisphosphate carboxylase, small chain	99	0.721	3	0.03
Serpins (serine protease inhibitors)	98	0.908	9	0.09
Basic region plus leucine zipper transcription factors	95	0.217	10	0.11
Protein-tyrosine phosphatase	92	0.577	8	0.09
Cysteine proteases	91	0.682	11	0.12
L-lactate dehydrogenases	90	0.910	6	0.07
2Fe-2S iron-sulfur cluster binding domains	88	0.785	5	0.06
RNase H	87	0.186	12	0.14
Ser/Thr protein phosphatases	86	0.810	5	0.06
Chaperonins 60 Kd subunit	84	0.948	5	0.06
Ank repeat	83	0.151	10	0.12
Retroviral aspartyl proteases	82	0.156	12	0.15
Subtilase family of serine proteases	82	0.563	9	0.11
Cytochrome C oxidase subunit I	80	0.215	13	0.16
Cyclins	80	0.635	9	0.11
ATP synthase A chain	79	0.649	6	0.08
Transforming growth factor beta like domain	79	0.296	6	0.08
C2 domain	78	0.175	6	0.08
Glutamine synthetase	78	0.807	5	0.06
Thioredoxins	76	0.723	11	0.14
PH (pleckstrin homology) domain	75	0.150	5	0.07
Oxidoreductases, nitrogenase component 1 and other families	75	0.800	8	0.11
Sushi domain	75	0.454	8	0.11
Peptide hormones (family of glucagon, GIP, secretin, VIP)	73	0.613	2	0.03
Photosynthetic reaction center protein	73	0.888	1	0.01
Eukaryotic aspartyl proteases	72	0.771	12	0.17
C-type lysozymes and alpha-lactabulmin	72	0.907	1	0.01
recA bacterial DNA recombination proteins	72	0.928	3	0.04
Pancreatic ribonucleases	71	0.926	1	0.01
Glutamine amidotransferases class-I	69	0.605	8	0.12
Aldehyde dehydrogenases	69	0.907	9	0.13
Iron/manganese superoxide dismutases (SODM)	69	0.931	5	0.07
Zinc finger, C3HC4 type	69	0.093	10	0.14
Phorbol esters / diacylglycerol binding domain	68	0.112	7	0.10
Small cytokines (intecrine/chemokine), interleukin-8 like	67	0.662	4	0.06
ATPases Associated with various cellular Activities (AAA)	66	0.378	8	0.12
Copper/zinc superoxide dismutases (SODC)	66	0.912	5	0.08
Bacterial regulatory helix-loop-helix proteins, araC family	63	0.348	9	0.14
Aminotransferases class-I	63	0.942	7	0.11
ATP synthase subunit C	62	0.855	5	0.08
Cystine-knot domain	61	0.502	4	0.07
Copper binding proteins, plastocyanin/azurin family	61	0.835	3	0.05
Mitochondrial carrier proteins	61	0.874	7	0.11
Sigma-70 factors	61	0.679	5	0.08
Carboxylesterases	60	0.900	5	0.08
Ribosomal protein S12	60	0.958	2	0.03
NADH dehydrogenases	57	0.933	4	0.07
Chaperonins 10 Kd subunit	57	0.953	4	0.07
Metallothioneins	56	0.963	0	0.00
Pilins (bacterial filaments)	56	0.700	6	0.11
Sigma-54 transcription factors	56	0.663	9	0.16
Kunitz/Bovine pancreatic trypsin inhibitor domain	55	0.665	5	0.09
Heme-binding domain in cytochrome b5 and oxidoreductases	55	0.250	4	0.07
Neuraminidases	55	0.799	2	0.04

Table 1: **PST performance for all Pfam families (part 2)**

Table 1...continued

Family	Size	Coverage	No. sequences missed	error-rate
Peroxidases	55	0.745	7	0.13
Ribosomal protein S4	54	0.952	4	0.07
Fungal Zn(2)-Cys(6) binuclear cluster domain	54	0.061	10	0.19
Crystallins beta and gamma	53	0.851	1	0.02
Cystatin domain	53	0.742	4	0.08
Pancreatic hormone peptides	53	0.760	5	0.09
Phosphoglycerate kinases	51	0.984	3	0.06
Beta-lactamases	51	0.863	7	0.14
Thrombospondin type 1 domain	51	0.152	6	0.12
Peptidyl-prolyl cis-trans isomerases	50	0.780	3	0.06
4Fe-4S iron sulfur cluster binding proteins, NifH/frxC family	49	0.928	2	0.04
C-5 cytosine-specific DNA methylases	48	0.846	8	0.17
Interferon alpha and beta domains	47	0.987	2	0.04
Pou domain - N-terminal to homeobox domain	47	0.234	2	0.04
DNA polymerase family B	46	0.650	9	0.20
Purine/pyrimidine phosphoribosyl transferases	45	0.831	5	0.11
Bacterial transferase hexapeptide (four repeats)	45	0.184	8	0.18
Myosin head (motor domain) (contains ATP/GTP binding P-loop)	44	0.439	10	0.23
Arf family (contains ATP/GTP binding P-loop)	43	0.951	4	0.09
Legume lectins alpha domain	43	0.356	3	0.07
Pyridine nucleotide-disulphide oxidoreductases class-I	43	0.938	7	0.16
Cyclic nucleotide-binding domain	42	0.466	3	0.07
Triosephosphate isomerase	40	0.973	3	0.07
Cellulases (glycosyl hydrolases)	40	0.584	6	0.15
Connexin	40	0.687	1	0.03
Enolases	40	0.983	0	0.00
picornavirus capsid proteins	40	0.431	2	0.05
Fibroblast growth factors	39	0.691	1	0.03
Beta-ketoacyl synthases	38	0.741	7	0.18
Kringle domain	38	0.298	2	0.05
Legume lectins beta domain	38	0.749	7	0.18
Ribosome inactivating proteins	37	0.716	2	0.05
7 transmembrane receptor (Secretin family)	36	0.735	2	0.06
KH domain family of RNA binding proteins	36	0.195	4	0.11
Molybdopterin binding domain in oxidoreductases	35	0.487	1	0.03
tRNA synthetases class I	35	0.743	7	0.20
Thyroglobulin type-1 repeat	32	0.161	3	0.09
Cadherin	31	0.503	4	0.13
Hemopexin	31	0.458	3	0.10
Low-density lipoprotein receptor domain class A	31	0.150	5	0.16
TNFR/NGFR cysteine-rich region	29	0.353	4	0.14
Ligand-gated ionic channels	29	0.836	1	0.03
tRNA synthetases class II	29	0.702	5	0.17
von Willebrand factor type A domain	29	0.277	6	0.21
Zona pellucida-like domain	26	0.484	3	0.12
Thiolases	25	0.965	3	0.12
Methyl-accepting chemotaxis protein (MCP) signaling domain	24	0.107	4	0.17
Lipases	23	0.779	3	0.13
von Willebrand factor type C domain	23	0.090	6	0.26
Fibronectin type II domain	20	0.141	2	0.10
Trefoil (P-type) domain	20	0.361	3	0.15
Laminin G domain	19	0.248	2	0.11
Fibrinogen beta and gamma chains, C-terminal globular domain	18	0.469	4	0.22
Laminin EGF-like (Domains III and V)	16	0.215	3	0.19
Fibronectin type I domain	15	0.107	2	0.13
u-PAR/Ly-6 domain	14	0.908	2	0.14
Double-stranded RNA binding motif	14	0.226	2	0.14
Low-density lipoprotein receptor domain class B	14	0.209	1	0.07
WAP-type (Whey Acidic Protein) 'four-disulfide core'	13	0.566	2	0.15
7 transmembrane receptor (metabotropic glutamate family)	12	0.805	2	0.17

Table 1: **PST performance for all Pfam families (part 3)**.