BIOINFORMATICS

Vol. 17 no. 1 2001
Pages 23–43

Variations on probabilistic suffix trees: statistical modeling and prediction of protein families

Gill Bejerano[1] and Golan Yona[2,3,*]

[1]School of Computer Science and Engineering, Hebrew University, Jerusalem 91904, Israel and [2]Department of Structural Biology, Fairchild Bldg. D-109, Stanford University, CA, 94305, USA

Received on November 1, 1999; revised and accepted on June 7, 2000

ABSTRACT

Motivation: 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 patterns can be of arbitrary length, and 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. Basic biological considerations such as amino acid background probabilities, and amino acids substitution probabilities can be incorporated to improve performance.

Results: 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 the Pfam database of protein families with more than satisfactory performance. Exhaustive evaluations show that the PST model detects much more related sequences than pairwise methods such as Gapped-BLAST, and is almost as sensitive as a hidden Markov model that is trained from a multiple alignment of the input sequences, while being much faster.

Availability: The programs are available upon request from the authors.

Contact: jill@cs.huji.ac.il; golan@cs.cornell.edu

INTRODUCTION

In the last few years there is a growing effort to organize the known protein sequences into families and establish databases of protein motifs and domains. Such efforts have led to the compilation of databases such as ProDom (Corpet *et al.*, 1999), Pfam (Bateman *et al.*, 1999), PROSITE (Hofmann *et al.*, 1999), PRINTS (Attwood *et al.*, 1999), Blocks (Henikoff *et al.*, 1999), Domo (Gracy and Argos, 1998), IDENTIFY (Nevill-Manning *et al.*, 1998) and SMART (Ponting *et al.*, 1999). 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 hidden Markov models (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 known 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. PROSITE, PRINTS, Blocks, IDENTIFY) and those that seek whole domains and try to infer domain boundaries (e.g. ProDom, Pfam, Domo, SMART). Some are based on manual or semi-manual procedures (e.g. PROSITE, PRINTS), others are generated semi-automatically (Pfam, SMART) and the rest—in a fully automatic manner (e.g. ProDom, Blocks, Domo, IDENTIFY). The methods used to represent motifs and domains vary. Among the common forms are consensus patterns and regular expressions (Hofmann *et al.*, 1999; Attwood *et al.*, 1999; Nevill-Manning *et al.*, 1998), position-specific scoring matrices, or profiles (Gribskov *et al.*, 1987; Henikoff *et al.*, 1999) and HMMs (Krogh *et al.*, 1996). These forms differ in their mathematical complexity, as well as in their sensitivity and selectivity.

To model a motif, a domain or a protein family, many approaches start off by building a multiple alignment (such are all the methods mentioned above). 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 (Samudrala and Moult, 1997; Bates and Sternberg, 1999). Moreover, the optimal alignment itself is not guaranteed to be biologically accurate. When the related sequences

*To whom correspondence should be addressed.

[3]Address starting from January 2001: Department of Computer Science, Cornell University, Ithaca, NY 14853, USA.

are diverged and the common pattern cannot be easily identified and distinguished 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 or HMM) may be of poor quality. Delineation of domain boundaries makes the problem even more difficult. Therefore, fully automated 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 (Hofmann *et al.*, 1999) serves as an excellent example. Another example is the Pfam database (Bateman *et al.*, 1999) that uses a semi-automatic procedure, which combines manual analysis and experts knowledge with automatic tools. Some of the other databases use these manually-inspected databases as a seed (e.g. ProDom is based on Pfam, PRINTS is based on PROSITE, Blocks is based on PROSITE, PRINTS. Pfam-ProDom and DOMO, and IDENTIFY is based on PRINTS and Blocks). However, the sheer size of current databases calls for the development of high performance automated methods.

Not all methods for building a model of a protein family require prealigned sequences. For example, the SAM software package (Hughey and Krogh, 1998) builds an HMM from a set of input sequences by using the EM (Expectation Maximization) algorithm (Dempster *et al.*, 1977). An initial model is constructed based on some background distributions, and the sequences are aligned to that model. The model is improved iteratively, by aligning the sequences to the current model and recalculating the transition and emission probabilities based on these alignments. The process is repeated till convergence of the parameters. The Gibbs sampling method (Lawrence *et al.*, 1993) is another method for finding short conserved patterns within a set of unaligned sequences. Each pattern is represented as a probabilistic model of specified length w, with residue frequencies for each position along the pattern. An initial model is constructed by choosing random starting positions within the input sequences, and calculating residue frequencies. The most probable common pattern is searched by iteratively applying a two-step sampling algorithm. First, one of the sequences is selected at random and is excluded from the set and a model is calculated based on all other sequences in the set. This model is then used to calculate the probabilities of all segments of length w in the excluded sequence, and one of them is selected with probability that corresponds to the ratio of model and background probabilities. Another algorithm for discovering multiple motifs in Protein or DNA sequences is MEME (Bailey and Elkan, 1995). To model a motif, this algorithm fits a two-component finite mixture model to the set of sequences using the EM technique, one component describes the motif (fixed length, ungapped subsequence) and the second describes the background (other positions in the sequence). Multiple motifs are found by iteratively applying this algorithm, after erasing the occurrences of those motifs that have already been found, effectively fitting a mixture of two-component models to the data.

Other approaches that do not require prealigned sequences were tested as well (Smith *et al.*, 1990; Neuwald and Green, 1994; Suyama *et al.*, 1995; Hanke *et al.*, 1996), but most of them are limited to short patterns with limited flexibility (i.e. some constraints are imposed on the pattern, such as exact or high conservation at pattern positions and fixed spacing between the main pattern elements). Such techniques were used to construct the Blocks database (Henikoff *et al.*, 1999), 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. Jonassen *et al.* (1995) proposed an improved method that allows greater ambiguity at partially conserved pattern positions and limited variable spacing between pattern elements.

Here we present an alternative approach for detecting significant patterns, based on probabilistic suffix trees (PSTs), originally introduced in Ron *et al.* (1996). 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. In particular, 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 which is more complex and allows to capture subtler features of the original source. These two families, although able of modeling 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

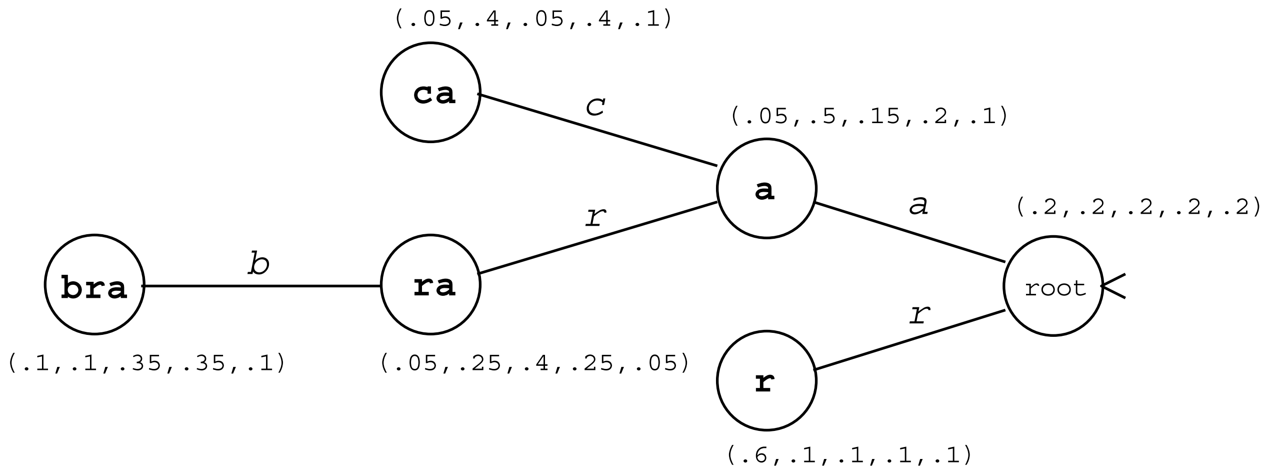


Fig. 1. An example of a PST over the alphabet $\Sigma = \{a, b, c, d, r\}$. The tree is shown in landscape mode, which makes the prediction step easier to follow. The root is the rightmost node. The vector that appears near each node is the probability distribution over the next symbol. For example,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

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 and Warmuth, 1992; Gillman and Sipser, 1994), and consequently, the derived model is not guaranteed to be optimal (this may suggest that for diverged families 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 of models can model rich sources with high efficiency using a reasonable amount of memory. The PST is essentially a variable length Markov model and its strength stems from its memory length which is *context dependent*, a phenomena observed in 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 the learning algorithm, and introduce variants of the PST model that incorporate biological considerations. Then we demonstrate its power by applying it to known protein families. For each family in our reference set we learn a PST and then use these PSTs to classify database protein sequences. A performance evaluation procedure is applied to compare the quality of the PST model with the quality of the common methods. Few examples in which the PST is being used to predict segments of presumably functional or structural importance, as well as other biological implications are then discussed.

THEORY

The definitions and subsequently implemented algorithms are variants on the learning algorithm originally presented in (Ron *et al.*, 1996).

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 (i.e. segments of high probability), this probability distribution vector comes into play. It corresponds to the probabilities the tree assigns to a query symbol, given that the longest subsequence of symbols that have been observed before it in the query matches that particular node's label. An example of a PST is given in Figure 1.

It should be noted that the PST differs from, albeit related to, the classical suffix tree, which contains all the suffixes of a given string (Gusfield, 1997). Consider the PST in Figure 1, and allow us to refer to a node using the (unique) label associated with it. In a suffix tree the father of node (*bra*) would have been node (*br*), whereas in a PST the father of a node is a node without the first (as opposed to last) symbol. Here the father of node (*bra*) is node (*ra*). The following observation specifies the relation between the two data structures: the skeleton (nodes, edges and labels) of a PST for a given input string is simply a subtree of the suffix tree associated with the *reverse* of that string. The differences become clear when following the tree construction procedure, described in Section *Building the PST*.

Definitions

Let Σ be the alphabet (e.g. the alphabet of 20 amino acids for protein sequences, or four nucleotides for DNA sequences), and let $r^1, r^2, \ldots, r^m$ be the sample set of m strings over the alphabet Σ, where the length of the ith ($i = 1 \cdots m$) string is l_i (i.e. $r^i = r_1^i r_2^i \cdots r_{l_i}^i \quad r_j^i \in \Sigma$).

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

$$\chi_s^{i,j} = \begin{cases} 1 & \text{if } s_1 s_2 \cdots s_l = r_j^i r_{j+1}^i \cdots r_{j+(l-1)}^i \\ 0 & \text{otherwise} \end{cases}$$

for each $i = 1 \cdots m$ and $j = 1 \cdots l_i - (l-1)$. Such indicator variable $\chi_s^{i,j}$ has a value of one if and only if the string s is a subsequence of r^i starting at position j.

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

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

The total number of (overlapping) subsequences of length $|s| = l$ within the set $\{r^i\}$ is

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

We choose to define the empirical probability of observing string s as the ratio between these last two quantities

$$\tilde{P}(s) = \frac{\chi_s}{N_{|s|}}.$$

The exact empirical probability depends on the number of possible occurrences of s in the sample set. In general, computing the maximal number of possible occurrences of a *specific* string s is more complicated and is dominated by the period of that string (the minimal interval which will allow it to overlap itself). Our definition implicitly disregards the fact that the subsequences accounted for are indeed overlapping, and therefore are not independent of each other. However, this definition suffices for our purposes. Note that it also leads to a natural definition of a probability distribution over all strings of length l since $\Sigma_{s \in \Sigma^l} \tilde{P}(s) = 1$.

We go on to define the conditional empirical probability of observing a symbol right after a given subsequence. This probability is defined as the number of times that symbol has shown up right after the given subsequence divided by the total number of times that subsequence 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

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

Finally, we define $suf(s) = s_2 s_3 \cdots s_l$, and $s^R = s_l \cdots s_2 s_1$.

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 subsequences of lengths 1 through L, starting at single letter subsequences, and as abstaining from further extending a subsequence 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 (in L) search space.

At the beginning of the search we hold a PST consisting of a single root node. Then, for each subsequence 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 subsequence 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 subsequence[†]. Whenever these two conditions hold, the subsequence, 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.

[†] 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 node prediction functions are added to the resulting PST skeleton, 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 subsequence (even though the empirical counts may attest differently).

We now present the procedure for building a PST out of a sample set. The procedure uses five 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, $\gamma_{\min}$ the smoothing factor, and α, a parameter that together with the smoothing probability defines the significance threshold for a conditional appearance of a symbol (and example of an effective set of parameters is given in the legend of Table 1).

We use $\bar{T}$ to denote the tree, $\bar{S}$ to denote the set of (unique) strings that 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** $(P_{\min}, \alpha, \gamma_{\min}, r, L)$

```
(1) Initialization: let $\bar{T}$ consist of a single root node (with an empty label), and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma \text{ and } \tilde{P}(\sigma) \leq P_{\min}\}$.
(2) Building the PST skeleton: while $\bar{S} \neq \phi$, 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
            $\tilde{P}(\sigma \mid s) \geq (1+\alpha)\gamma_{\min}$
        and
            $\frac{\tilde{P}(\sigma \mid s)}{\tilde{P}(\sigma \mid suf(s))} \begin{cases} \geq r \\ \text{or} \\ \leq 1/r \end{cases}$
        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 add the strings $\{\sigma' s \mid \sigma' \in \Sigma \text{ and } \tilde{P}(\sigma' s) \geq P_{\min}\}$ (if any) to $\bar{S}$.
(3) Smoothing the prediction probabilities: for each $s$ labeling a node in $\bar{T}$, let
        $\bar{\gamma}_s(\sigma) \equiv (1 - |\Sigma|\gamma_{\min})\tilde{P}(\sigma \mid s) + \gamma_{\min}$.   (1)
```

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 $\gamma_{\min}$ 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[‡].

Returning to the PST of Figure 1, we can now make a couple of exemplary observations on the data set from which the model was learned:

- In the training set there must be an overall clear preference for observing the letter b after the letter a (as $\bar{\gamma}_a(b) = 0.5$), unless the a itself was preceded by an r, in which case the preference is for c (as $\bar{\gamma}_{ra}(c) = 0.4$).
- Assuming for a moment that $\gamma_{\min}$ was set to 0.05, and examining the probability vector associated with node (ca), we can infer that only three different symbols, b, d and r, were observed in the training set succeeding the subsequence ca, in quantities that obey the ratio 7 : 7 : 1, respectively.

A variant—incorporating biological considerations in 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 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 q_{ab} be the probability that amino acid a is replaced by amino acid $b (a, b \in \Sigma)$. Also, define $\bar{\chi}_s$ as the number of *distinct* sequences which include the subsequence 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 subsequence s is required to exist in at least 20% of the member proteins). Alternatively, this parameter can be set to a constant value regardless of the actual size of the training set.

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

[‡] Specifically, solving the set of equations $\forall\sigma \in \Sigma \bar{\gamma}_s(\sigma) = k * \tilde{P}(\sigma \mid s) + \gamma_{\min}$ with $\Sigma_{\sigma\in\Sigma}\bar{\gamma}_s(\sigma) = 1$ yields $k = (1 - |\Sigma|\gamma_{\min})$ with the constraint $\gamma_{\min} < \frac{1}{|\Sigma|}$ and equation (1) follows.

the method suggested by Henikoff and Henikoff (1996)). This method adds hypothetical counts 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* (the number of different amino acids observed after the corresponding subsequence), as well as the counts of actually observed amino acids.

For a node represented by the subsequence s, denote by R_s the diversity at s (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 to node s. We set $B_s = \mu R_s$, as suggested in Henikoff and Henikoff (1996), where μ can be optimized for best performance[§]. Then, the number of pseudo-counts of amino acid a at this node is given by

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

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

The modified procedure is described next:

The algorithm: **Build-Bio-PST** ($N_{\min}, \gamma, r, L$)

```
1. Initialization: let $\bar{T}$ consist of a single root node (with an empty label), and let $\bar{S} \leftarrow \{\sigma \mid \sigma \in \Sigma$ and $\bar{\chi}_\sigma \geq N_{\min}\}$.
2. Building the PST skeleton: while $\bar{S} \neq \phi$, 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
            $\tilde{P}(\sigma \mid s) \geq \gamma$
        and
            $\frac{\tilde{P}(\sigma \mid s)}{\tilde{P}(\sigma \mid suf(s))} \begin{cases} \geq r \\ \text{or} \\ \leq 1/r \end{cases}$
        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 add the strings $\{\sigma' s \mid \sigma' \in \Sigma$ and $\bar{\chi}_{\sigma' s} \geq N_{\min}\}$ (if any) to $\bar{S}$.
3. Smoothing the prediction probabilities: for each $s$ labeling a node in $\bar{T}$, let
```

$$\begin{aligned} \bar{\gamma}_s(\sigma) &\equiv \frac{\chi_{s*}}{\chi_{s*} + B_s} \tilde{P}(\sigma \mid s) + \frac{B_s}{\chi_{s*} + B_s} P_{\mathrm{pse}}(\sigma \mid s) \\ &= \frac{\chi_{s*}}{\chi_{s*} + B_s} \frac{\chi_{s\sigma}}{\chi_{s*}} + \frac{B_s}{\chi_{s*} + B_s} \frac{b_\sigma}{B_s} = \frac{\chi_{s\sigma} + b_\sigma}{\chi_{s*} + B_s}. \end{aligned}$$

§ We have, ad hoc, used the same value as in Henikoff and Henikoff (1996).

We briefly summarize the changes with respect to the external parameters:
$N_{\min}$ substitutes $P_{\min}$, while the pseudo-counts matrix replaces $\gamma_{\min}$ in smoothing, leaving γ to take over for α and $\gamma_{\min}$ in threshold determination.

Incremental model refinement. A useful and very practical feature of the PST learning algorithm is the ability to refine any given PST model (over the same training set it was originally grown from) without the need to re-iterate calculations which have already been made while building the original model. Given the PST model to be further expanded, denoted T_0, one can, by relaxing the original learning parameters, in any of the ways described below, extend the model without repeating any of the calculations taken in building T_0 itself. In order to increase the number of nodes to be considered for inclusion in the resulting PST model one may lower $P_{\min}$ or increase L. In order to alleviate the criteria for acceptance into the tree one may lower r towards 1, or α towards (-1). Once the relaxed set of parameters has been chosen (any subset of them may be relaxed simultaneously) and T_0 has been passed to Build-PST, the initialization step of the algorithm should be altered to the following:

```
1. Initialization: if $T_0$ is empty
    (a) As before.
    (b) Else, let $T \leftarrow T_0$, and let $\bar{S} \leftarrow \{s \mid suf(s) \in T_0$ and $s \notin T_0$ and $\tilde{P}(s) \geq P_{\min}$ and $|s| \leq L\}$.
```

The second variant can be improved incrementally much in the same way.

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 Figure 1 the following procedure is carried out:

$$\begin{aligned} &P^T(abracadabra) \\ &= P^T(a)P^T(b \mid \underline{a})P^T(r \mid ab)P^T(a \mid ab\underline{r})P^T(c \mid a\underline{bra}) \\ &\quad \times P^T(a \mid abrac) \cdots P^T(a \mid abracadab\underline{r}) \\ &= \bar{\gamma}_{\text{root}}(a)\, \bar{\gamma}_a(b)\, \bar{\gamma}_{\text{root}}(r)\, \bar{\gamma}_r(a)\, \bar{\gamma}_{bra}(c)\, \tilde{\gamma}_{\text{root}}(a) \cdots \bar{\gamma}_r(a) \\ &= \quad 0.2 \quad 0.5 \quad 0.2 \quad 0.6 \quad 0.35 \quad 0.2 \cdots 0.6 \\ &= 4.032 \cdot 10^{-6}. \end{aligned}$$

The underlined subsequences represent the longest suffices that appeared in the tree (no characters are underlined when the longest suffix is the empty string), and the probability of each letter is given by the prediction function that is associated with the corresponding node ($\bar{\gamma}_{\text{root}}()$, $\bar{\gamma}_a()$, $\bar{\gamma}_{bra}()$ etc.).

Note that a random uniform distribution over the alphabet would assign a probability of $0.2^{11} = 2.048 \cdot 10^{-8}$ to the string s (making s roughly 200 times more plausible under T than under the simple uniform model).

Two-way prediction. The prediction step described in the previous paragraph proceeds from left to right, starting from the leftmost letter. An obvious side effect of this property is that letters that mark the beginning of significant patterns are predicted with non significant probabilities. Only after a significant subsequence has been observed (i.e. once sufficient information is accumulated) the subsequent letters are predicted with high probability. Consequently, the (left) boundaries between significant and non significant patterns (i.e. domain/motif boundaries) are somewhat blurred. To accommodate for this property we have implemented a variant of the prediction step. Given a set of input sequences, two PST models are created, T and T^R. T is built from the sequences as they are, and T^R is built from the reversed sequences. The prediction step is repeated twice. The input sequence is predicted using T, and the reverse sequence is predicted using T^R. Then, the predictions are combined by taking the maximal prediction assigned by the two models. Thus, for $s = \tau\sigma\rho$ where $\sigma \in \Sigma$ and $\tau, \rho \in \Sigma^*$, $P_{T,T^R}(\sigma \mid s) = \max\{P_T\{\sigma \mid \tau), P_{T^R}(\sigma \mid \rho^R)\}$.

Complexity, run time and availability

Denote the total length of the training set by n, the depth bound on the resulting PST by L, and the length of a generic query sequence by m. In these terms, the learning phase of the algorithm can be bounded by $O(Ln^2)$ time and $O(Ln)$ space, as there may be $O(Ln)$ different subsequences of lengths 1 through L, each of which can be searched for, in principle, in time proportional to the training set length, while each contributes but a single node beyond its father node to the resulting structure. The prediction phase is bounded by $O(Lm)$ time, since every symbol predicted requires traversing a path from the root down a tree of maximal depth L. A speed up in prediction can always be achieved by a one time investment, of time complexity $O(Ln^2)$, in the straightforward conversion of the tree into a not much larger probabilistic finite automation, of identical predictions (see Ron *et al.*, 1996). We refer the readers to a subsequent work (Apostolico and Bejerano, 2000), where an alternative, slightly more powerful, algorithmic scheme is devised to achieve the optimal bounds of learning in $O(n)$ time and space (regardless of L, allowing thus for unbounded trees) and predicting in $O(m)$ time.

The implementation described in this paper was coded in ANSI C, compiled using `gcc`, and ran on Linux and BSDI based Pentium II and III machines (compiled versions for different platforms can be made available upon request). Actual run time on a Pentium II 300 Mhz PC, for a protein family, varied between 30 s and 90 min, resulting in models of size 10–300 Kb. In general, a tree of an ‘average’ Pfam family contains 6556 nodes (with the set of parameters given in Table 1). This is a small portion of all ‘potential nodes’ that are checked during the learning algorithm (on average, 32 765 potential nodes were inspected per model).

RESULTS AND DISCUSSION

To test our approach, a PST was created for each family in the Pfam database (Bateman *et al.*, 1999) release 1.0. This database contains 175 protein families¶ derived from the SWISSPROT 33 database (Bairoch and Apweiler, 1999). Each family is divided into a training set and a test set, in ratio 4 : 1, such that 4/5 of the family members (in the form of complete, unaligned sequences) serve as the PST training set. Then, for each sequence in the SWISSPROT 33 database we calculate 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 average 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 sets a threshold at the point where the number of false positives (the number of non member proteins with probability above the threshold) equals the number of false negatives (the number of member proteins with probability below the threshold), i.e. it is the point of balance between selectivity and

¶ 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 seed alignments. The HMMs can be used for searching close relatives in the database, and the most significant hits are included in the seed alignment of the corresponding family. The process iterates until it converges, while being traced manually.

Table 1. PST performance for all Pfam families (part 1). Families are ordered alphabetically. The names of the families are abbreviated as in the Pfam database. The number of proteins in the family is given in the second column. *Coverage* (third column) is the total portion of the sequences which is 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 built from the training set. To test the quality of this PST, we calculate the probability that the PST induces on each sequence in the SWISSPROT 33 database, and each family sequence (from the training set and the test set) whose probability is above the iso-point is considered successfully detected (see text for more details). The number of sequences missed (i.e. the equivalence number) and the percentage of true positives detected are given in the fourth and fifth columns respectively. The other columns give the number of sequences missed by: the Pfam HMM; an HMM that was trained on a multiple alignment in a local search mode (see text); an HMM that was trained on a multiple alignment in a global search mode; an HMM that was trained on the unaligned sequences; best Gapped-BLAST search, and average Gapped-BLAST search. The sign '-' denotes that results were not available (the program crashed). The set of parameters used to rain the PST is $P_{\min} = 0.0001$ $\alpha = 0$ $\gamma_{\min} = 0.001$ $r = 1.05$ and $L = 20$. Additional improvement in the performance is expected if the parameters are tuned for each family (see text). To train a PST on a typical family with this set of parameters it takes about half an hour on average, on a pentium II 300 Mhz (the variance is between 30 s and 90 min). Additional 5 min are needed to predict all sequences in the SWISSPROT database. For comparison, training an HMM from unaligned sequences takes about two hours on average, and searching the SWISSPROT database with a typical HMM may take several hours

Family	Size	Coverage	No. sequences missed by PST	% true positives detected by PST	No. of sequences missed by					
					HMM Pfam	HMM local	HMM global	HMM SAM	BLAST best	BLAST average
7tm_1	515	0.707	36	93.0	13	1	7	1	12	64
7tm_2	36	0.735	2	94.4	0	0	0	0	0	0
7tm_3	12	0.805	2	83.3	0	0	0	0	0	0
AAA	66	0.378	8	87.9	0	1	1	1	1	2
ABC_ tran	269	0.518	44	83.6	1	3	2	6	5	12
actin	142	0.965	4	97.2	4	2	2	1	0	3
adh_short	180	0.661	20	88.9	0	8	2	3	8	55
adh_zinc	129	0.970	6	95.3	1	2	1	3	2	7
aldedh	69	0.907	9	87.0	0	0	0	0	0	1
alpha-amylase	114	0.750	14	87.7	0	2	1	3	2	18
aminotran	63	0.942	7	88.9	0	1	0	1	16	28
ank	83	0.151	10	88.0	3	9	26	3	9	39
arf	43	0.951	4	90.7	0	0	0	0	0	0
asp	72	0.771	12	83.3	7	1	5	1	0	3
ATP-synt_A	79	0.649	6	92.4	3	1	1	0	1	11
ATP-synt_ab	180	0.694	6	96.7	6	1	3	0	1	4
ATP-synt_C	62	0.855	5	91.9	12	0	1	1	0	6
beta-lactamase	51	0.863	7	86.3	0	0	0	0	9	17
bZIP	95	0.217	10	89.5	1	4	6	2	22	46
C2	78	0.175	6	92.3	3	7	16	7	23	47
cadherin	31	0.503	4	87.1	0	1	1	1	2	5
cellulase	40	0.584	6	85.0	0	1	1	2	8	17
cNMP_binding	42	0.466	3	92.9	2	1	7	2	2	15
COesterase	60	0.900	5	91.7	7	1	4	1	0	2
connexin	40	0.687	1	97.5	0	0	0	0	0	0
copper-bind	61	0.835	3	95.1	0	0	0	0	14	26
COX1	80	0.215	13	83.8	1	4	3	5	2	6
COX2	109	0.897	2	98.2	11	0	2	0	0	3
cpn10	57	0.953	4	93.0	1	0	1	0	0	1
cpn60	84	0.948	5	94.0	0	0	0	0	0	0
crystall	53	0.851	1	98.1	0	0	0	0	0	2
cyclin	80	0.635	9	88.8	2	2	2	2	4	12
Cys-protease	91	0.682	11	87.9	11	2	9	0	0	4
cystatin	53	0.742	4	92.5	1	1	20	3	13	27

Table 1 Continued . . .

sensitivity (we use the term *iso-point* to denote this point). A hit that belongs to the family (true positive) and scores above this threshold, is considered successfully detected. The quality of the model is measured by the number of true positives 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 ten members each, are given in Table 1. When averaged over all 170 families, the PST detected 90.7% of the true positives.

Family	Size	Coverage	No. sequences missed by PST	% true positives detected by PST	No. of sequences missed by					
					HMM Pfam	HMM local	HMM global	HMM SAM	BLAST best	BLAST average
Cys-knot	61	0.502	4	93.4	0	1	-	6	12	25
cytochrome_b_C	130	0.313	27	79.2	2	20	19	18	1	17
cytochrome_b_N	170	0.658	3	98.2	22	2	3	0	0	1
cytochrome_c	175	0.891	11	93.7	2	4	6	5	30	66
DAG_PE-bind	68	0.112	7	89.7	1	7	5	13	9	33
DNA_methylase	48	0.846	8	83.8	2	0	2	0	0	2
DNA_pol	46	0.650	9	80.4	1	-	-	0	0	3
dsrm	14	0.226	2	85.7	1	0	6	1	7	10
E1-E2_ATPase	102	0.636	7	93.1	3	0	2	0	0	2
efhand	320	0.401	25	92.2	27	28	52	25	42	105
EGF	169	0.133	18	89.3	28	120	118	53	46	98
enolase	40	0.983	0	100.0	3	0	2	0	0	1
fer2	88	0.785	5	94.3	2	1	2	1	0	5
fer4	152	0.559	18	88.2	7	3	6	2	16	46
fer4_NifH	49	0.928	2	95.9	5	0	3	0	0	1
FGF	39	0.691	1	97.4	0	0	0	0	0	0
fibrinogen_C	18	0.469	4	77.8	0	1	-	0	0	1
filament	139	0.607	5	96.4	14	0	7	0	3	10
fn1	15	0.107	2	86.7	1	1	1	4	6	9
fn2	20	0.141	2	90.0	0	1	2	2	0	10
fn3	161	0.242	23	85.7	1	85	95	43	61	116
GATase	69	0.605	8	88.4	0	2	1	1	2	17
gln-synt	78	0.807	5	93.6	3	0	1	0	1	6
globin	681	0.974	15	97.8	5	3	3	2	48	136
gluts	144	0.849	14	90.3	0	3	1	2	23	59
gpdh	117	0.977	3	97.4	3	0	3	0	0	4
GTP_EFTU	184	0.802	15	91.8	0	3	1	2	1	5
heme_1	55	0.250	4	92.7	0	6	17	0	0	4
hemopexin	31	0.458	3	90.3	0	1	0	0	0	2
hexapep	45	0.184	8	82.2	1	2	10	1	5	16
histone	178	0.887	6	96.6	0	0	0	1	52	104
HLH	133	0.194	7	94.7	1	8	14	2	27	70
homeobox	383	0.333	27	93.0	13	2	16	3	9	57
hormone	111	0.961	4	96.4	0	2	0	2	2	4
hormone2	73	0.613	2	97.3	0	0	0	0	4	18
hormone3	53	0.760	5	90.6	0	0	0	0	2	5
hormone_rec	127	0.313	7	94.5	0	1	1	2	3	5
HSP20	129	0.625	7	94.6	1	0	1	0	10	34
HSP70	163	0.906	7	95.7	24	0	9	0	3	6
HTH_1	101	0.476	16	84.2	0	3	1	2	11	34
HTH_2	63	0.348	9	85.7	0	1	7	1	3	18
ig	884	0.414	51	94.2	12	-	-	35	248	553
il8	67	0.662	4	94.0	0	0	0	0	2	18
ins	132	0.715	3	97.7	0	0	0	0	7	20
interferon	47	0.987	2	95.7	0	0	0	0	0	0
kazal	110	0.735	6	94.5	1	1	1	2	3	13
ketoacyl-synt	38	0.741	7	81.6	0	0	0	0	0	2
KH-domain	36	0.195	4	88.9	4	6	23	13	21	28
kringle	38	0.298	2	94.7	0	1	2	1	0	8
Kunitz-BPTI	55	0.665	5	90.9	0	2	38	0	0	4
laminin_EGF	16	0.215	3	81.2	1	0	-	-	2	5
laminin_G	19	0.248	2	89.5	0	1	-	1	3	11
ldh	90	0.910	6	93.3	16	1	14	4	11	27
ldl_recept_a	31	0.150	5	83.9	0	2	6	2	5	16
ldl_recept_b	14	0.209	1	92.9	0	1	1	0	1	1
lectin_c	106	0.478	14	86.8	1	1	5	1	3	44

Table 1 Continued . . .

Family	Size	Coverage	No. sequences missed by PST	% true positives detected by PST	No. of sequences missed by					
					HMM Pfam	HMM local	HMM global	HMM SAM	BLAST best	BLAST average
lectin_legA	43	0.356	3	93.0	0	1	5	0	0	0
lectin_legB	38	0.749	7	81.6	6	5	5	10	2	5
lig_chan	29	0.836	1	96.6	2	0	0	0	0	0
lipase	23	0.779	3	87.0	6	0	0	0	0	0
lipocalin	115	0.858	7	93.9	4	1	4	0	50	74
lys	72	0.907	1	98.6	5	0	3	0	1	3
MCPsignal	24	0.107	4	83.3	0	0	0	0	0	0
metalthio	56	0.963	0	100.0	5	0	0	0	2	4
MHC_I	151	0.494	3	98.0	1	0	0	0	0	0
mito_carr	61	0.874	7	88.5	1	0	0	0	0	1
myosin_head	44	0.439	10	77.3	1	4	-	0	0	6
NADHdh	57	0.933	4	93.0	5	0	2	0	0	0
neur	55	0.799	2	96.4	3	0	2	0	0	2
neur_chan	138	0.882	4	97.1	5	1	2	2	1	4
oxidored_fad	101	0.244	12	88.1	0	3	19	1	15	49
oxidored_molyb	35	0.487	1	97.1	2	0	0	0	0	1
oxidored_nitro	75	0.800	8	89.3	20	3	18	3	5	33
p450	204	0.917	17	91.7%	0	2	1	2	2	7
peroxidase	55	0.745	7	87.3%	6	0	3	0	12	32
PGK	51	0.984	3	94.1%	1	0	1	0	0	0
PH	75	0.150	5	93.3%	1	4	17	15	43	67
phoslip	122	0.938	3	97.5%	8	0	5	0	0	3
photoRC	73	0.888	1	98.6%	2	0	1	0	0	1
pilin	56	0.700	6	89.3%	10	2	3	2	0	3
pkinase	725	0.523	108	85.1%	23	-	-	1	6	52
pou	47	0.234	2	95.7%	1	0	0	0	0	0
Pribosyltran	45	0.831	5	88.9%	1	0	2	3	18	23
pro_isomerase	50	0.780	3	94.0%	1	0	0	0	0	1
pyr_redox	43	0.938	7	83.7%	0	0	0	0	0	0
ras	213	0.930	8	96.2%	1	2	1	2	1	3
recA	72	0.928	3	95.8%	7	0	4	0	0	0
response_reg	128	0.424	19	85.2%	1	25	20	2	5	27
rhv	40	0.431	2	95.0%	0	2	-	1	0	5
RIP	37	0.716	2	94.6%	8	0	7	0	4	15
rnaseA	71	0.926	1	98.6%	1	0	1	0	0	1
rnaseH	87	0.186	12	86.2%	0	7	7	7	8	13
rrm	141	0.353	22	84.4%	2	-	-	11	21	79
RuBisCO_large	311	0.995	4	98.7%	20	0	19	0	0	0
RuBisCO_small	99	0.721	3	97.0%	0	0	0	0	0	0
rvp	82	0.156	12	85.4%	2	14	-	13	12	26
rvt	147	0.237	17	88.4%	0	10	12	10	24	56
S12	60	0.958	2	96.7%	3	0	3	0	0	1
S4	54	0.952	4	92.6%	0	2	1	2	1	3
serpin	98	0.908	9	90.8%	8	1	2	1	0	3
SH2	128	0.157	5	96.1%	3	23	18	29	16	50
SH3	137	0.126	16	88.3%	0	43	36	57	18	72
sigma54	56	0.663	9	83.9%	6	3	4	0	2	6
sigma70	61	0.679	5	91.8%	0	0	0	0	0	2
sodcu	66	0.912	5	92.4%	7	1	4	1	3	6
sodfe	69	0.931	5	92.8%	7	0	6	0	2	5
ST phosphatase	86	0.810	5	94.2%	2	0	0	0	0	0
subtilase	82	0.563	9	89.0%	0	1	1	1	0	13
sugar_tr	107	0.880	15	86.0%	2	6	5	6	14	33
sushi	75	0.454	8	89.3%	0	1	21	1	3	31
TGF-beta	79	0.296	6	92.4%	3	1	4	1	1	4
thiolase	25	0.965	3	88.0%	0	0	0	0	0	0

Table 1 Continued . . .

Family	Size	Coverage	No. sequences missed by PST	% true positives detected by PST	No. of sequences missed by					
					HMM Pfam	HMM local	HMM global	HMM SAM	BLAST best	BLAST average
thiored	76	0.723	11	85.5%	4	1	3	0	2	6
thyroglobulin_1	32	0.161	3	90.6%	0	2	27	3	0	7
TIM	40	0.973	3	92.5%	1	0	0	0	0	0
TNFR_c6	29	0.353	4	86.2%	0	0	3	2	2	7
toxin	172	0.936	4	97.7%	2	1	2	1	2	9
trefoil	20	0.361	3	85.0%	0	3	11	0	1	8
tRNA-synt_1	35	0.743	7	80.0%	0	0	0	0	1	2
tRNA-synt_2	29	0.702	5	82.8%	0	0	0	0	0	2
trypsin	246	0.730	22	91.1%	0	1	0	1	0	4
tsp_1	51	0.152	6	88.2%	0	3	4	1	2	30
tubulin	196	0.943	1	99.5%	13	0	5	0	1	2
UPAR_IY6	14	0.908	2	85.7%	0	0	0	0	2	5
vwa	29	0.277	6	79.3%	0	2	5	4	10	20
vwc	23	0.090	6	73.9%	0	8	-	2	8	15
wap	13	0.566	2	84.6%	0	0	0	0	1	3
wnt	102	0.936	6	94.1%	66	0	0	0	0	0
Y_phosphatase	92	0.577	8	91.3%	34	0	3	1	4	19
zf-C2H2	297	0.362	23	92.3%	4	13	22	11	15	56
zf-C3HC4	69	0.093	10	85.5%	2	3	28	2	18	48
zf-C4	139	0.152	6	95.7%	1	1	3	1	0	1
zf-CCHC	105	0.072	12	88.6%	1	11	7	12	7	40
zn-protease	148	0.029	21	85.8%	4	110	111	13	92	125
Zn_clus	54	0.061	10	81.5%	0	2	8	1	3	24
zona-pellucida	26	0.484	3	88.5%	0	0	0	0	0	7

Comparison with HMM and performance evaluation

The performance evaluation procedure that we applied assesses the quality of the PST model in terms of its ability to predict the correct assignments of proteins to *a priori defined groups*, and our reference set here is the HMM based Pfam database. Note that this assessment does not measure the relative merit of the PST model with respect to the HMM in general, since the reference set depends on the HMM itself (see footnote ¶ on page 29). In order to compare the performance of the PST model to the performance of the HMM in a more objective manner, we built an HMM for each family, out of the same training set that was used to build the PST model, and tested its ability to detect family members using the same equivalence number criterion. These HMMs were built automatically, without any manual calibration, using two public domain software packages which are available through the web, namely, the SAM package version 2.2 (Hughey and Krogh, 1998), and the HMMER package version 2.1 (Eddy, 1998).

The compared methods. For each family, three HMMs were built. The first one was built directly from the set of unaligned sequences using the program 'buildmodel' of the SAM package. The model was then used to search the database using the 'hmmscore' program. The other two models were built after the sequences were aligned first using the ClustalW program, version 1.75 (Higgins *et al.*, 1996). These models were created using the program 'hmmbuild' which is part of the HMMER package‖. In this package the mode of the search (local or global) is part of the model itself. Therefore, one of the models was created in a local/global mode (allows local match with respect to the sequence, and a global match with respect to the model, i.e. only a complete match with the model is reported), and the second was created in a local/local mode. Both allow multiple matches with the same sequence (corresponding to multiple copies of the same domain). These models were then used to search the database using the 'hmmsearch' program of the HMMER package. The results of our assessment are summarized in Table 1.

For reference, we have also evaluated the performance of the HMMs which are part of the Pfam database itself (available through the Pfam homepage at http://www.sanger.ac.uk/Pfam/). These HMMs are based on manually calibrated alignments and are tested on all family sequences (see footnote ∗∗ on page 35). As another

‖ In its current version, this program can build an HMM only from a multiple alignment of the input sequences.

reference test we run Gapped-BLAST (Altschul *et al.*, 1997) with each of the member sequences in these families as a query. The performance of this pairwise search algorithm is given both in terms of the best query, and the average query.

Evaluation results. Overall, the manually calibrated Pfam HMMs detected 96.1% of the true positives (averaged over 170 families with more than ten members). This is only slightly better than the average performance of HMMs that were built from a multiple alignment of the input sequences, in a local/local mode of comparison (96.0% over 166 families). When the HMMs were built from the same multiple alignments, but in a local/global search mode, the performance dropped to 91.5% (averaged over 158 families). The HMMs that were built directly from the unaligned sequences using the SAM package performed surprisingly well, with 96.7% success over 169 families. This is slightly more discriminative in comparison with the manually calibrated Pfam HMMs. (This may be explained by the fact that Pfam HMMs are based on seed alignments of a small sample set of the family, while the SAM HMMs are trained on 4/5 of the family members.) A marginal HMMs are trained on 4/5 of the family members.) A marginal improvement (0.1%) was observed with the SAM package when the rest of the sequences were used as a test set by the program 'build-model' (results not shown). Gapped-BLAST searches performed quite well (92.5% over 170 families), when the 'best' query was chosen for each family. However, a typical BLAST search (average performance for all member proteins) performed much worse (78.5% over 170 families).

According to our assessment, already in its very simplistic form and with a preliminary set of parameters, the PST model has detected 90.7% of the true positives in the reference set. This is much better than a typical BLAST search, and almost as good as an HMM that was trained from a multiple alignment of the input sequences in a global search mode. This result is surprising in view of the fact that the model was built in a fully automated manner, without any human intervention or biological consideration, and without utilizing any prior knowledge, such as multiple alignments or scoring matrices.

When the *build-bio-pst* procedure was applied (with substitution probabilities from which the blosum62 scoring matrix (Henikoff and Henikoff, 1992) was derived) to Pfam families, the overall performance was improved and 91.7% of the true positives were detected (results not shown). For many families, this procedure resulted in much smaller trees that performed same or better than the trees of the basic procedure (for example, the detection rate for the peroxidase family improved from 87.3% with 10 226 nodes using the *build-pst* procedure, to 96.4% with 5579 nodes, using the *build-bio-pst*). This is not true for all families, probably since our pruning criteria still need to be refined. In some families we observe minor fluctuations and one or two member proteins were missed by the new model. For small families each protein missed equals to few percentages less in performance, and consequently, the overall performance is affected. However, overall the new model performed better and detected 178 more sequences than the original model. Our results are yet preliminary, but in this view, this direction seems promising.

Critique of the evaluation methodology. A reservation regarding the evaluation procedure is in place. Comparing the PST model with HMM based on Pfam families is not a totally objective test, since the groups are defined based on an HMM. An optimal evaluation test would be based on an independent 'true' classification of proteins into families. However, no such classification clearly exists. Another reservation is that false positives may be over counted. It happens often that supposedly false positives with respect to the reference database are actually true positives (Yona *et al.*, 1999; Ponting *et al.*, 1999). For example, among the first 250 hits reported by the PST of the pkinase family four are supposedly false positives. However, these four are short proteins (21–42 amino acids long) documented as various types of kinases and their sequence similarity to other kinases is very significant. This problem is inherent in any evaluation procedure which assesses a new classification by means of another, man-made classification. However, as no clear baseline distribution exists and in the absence of other reliable evaluation procedures, this so called 'external validation test' is commonly used (Yona *et al.*, 1999; Gracy and Argos, 1998; Pearson, 1995). Our specific choice of the Pfam database was motivated by the high quality of this database.

Factors affecting the performance of the PST model. 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 run 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 (see Section *Incremental model refinement*).

The effect of parameter tuning on the performance is demonstrated in Figure 2 for the gyceraldehyde 3-

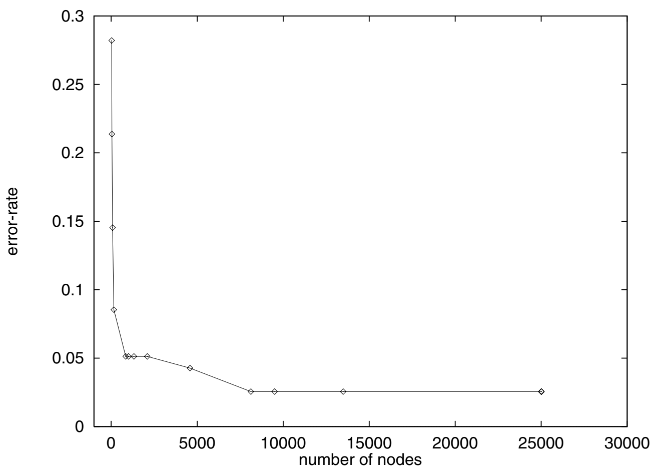


Fig. 2. Improving prediction by increasing the number of nodes. A PST was built for the gyceraldehyde 3-phosphate dehydrogenases family, for different values of $P_{\min}$, and the quality of the PST was estimated by applying the equivalence number criterion (see text). The graph plots the error rate (the number of family members which were not identified, as 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.

phosphate dehydrogenases family. In general, adding more nodes would tend to increase sensitivity without decreasing selectivity, simply because more (longer) subsequences 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, hopefully, refined. However, since long subsequences observed in the training set are not expected to occur in unrelated sequences, the prediction of unrelated sequences is based on short subsequences corresponding to internal nodes close to the root, and therefore it 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 run time, may increase significantly 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 should improve its prediction as well**.

Screening short sequences and score normalization. The performance of the PST model is even better if very short protein sequences are ignored. Since the ‘score’ assigned to each string by the PST model is normalized by the string’s length, very short sequences (i.e. shorter than 20 amino acids) may be assigned a relatively high probability. These sequences are too short to ‘activate’ the long memory prediction (i.e. the variable memory does not take effect), and hence the prediction is based on nodes closer to the root of the tree. Consequently, these strings may get a high probability simply by chance and are harder to distinguish from random strings. By discarding all sequences in the SWISSPROT database which are shorter than 20 amino acids (754 sequences), better prediction results are obtained. Specifically, the performance of the PST model improves by 2.3% (272 sequences) to detect 93% of the true positives in the reference set†† (results not shown). The performance of all HMM models remain unaffected.

While discarding these sequences can be justified because very short peptides are usually biologically meaningless, a different scoring scheme can neutralize to some extent the effect of these sequences. For example, a log odds score normalizes the raw probability of a string s by the chance probability to observe that string (Altschul, 1991). Formally speaking, let $P^T(s)$ be the probability assigned by the PST and let $P_0(s)$ be the chance probability defined as $P_0(s) = \Pi_{i=1\cdots|s|} P_0(s_i)$ (where $P_0(s_i)$ are defined by the background probabilities in the SWISSPROT database). The log odds score is defined as

$$\log \frac{P^T(s)}{P_0(s)}.$$

This ratio compares the probability of an event under two alternative hypotheses. Thus, the score for each amino acid along the sequence is defined as the logarithm of the amino acid’s PST probability divided by its probability of occurrence under independent selection.

The evaluation procedure was repeated with the normalized log odds scores, this time without screening the short

**In the Pfam database the model is built from a seed alignment which may contain only a fraction of the family. However, the procedure essentially uses all family members. The model is tested by searching the database for all other members of the family. If a true member is missed, it is added to the seed alignment, and the process is repeated. Finally, a full alignment is constructed for the family by aligning all members to the HMM. This full alignment is checked again (manually) and if it is not correct, the alignment method is modified or the whole process is restarted with a new improved seed.

†† Out of 15 604 sequences in the Pfam database release 1.0, only 5 are shorter than 20 amino acids, so the evaluation results are hardly affected by discarding those sequences.

Table 2. Improvement of PST performance in local-prediction mode. This variant was tested on five families on which the PST model performed worst in global-prediction mode (see text). Four out of the five families were better detected with the model that was tuned to local-prediction mode

Family	L_{domain}	N_{local}	P_{local}	N_{gap}	% True positives	
					Global prediction	Local prediction
myosin_hed	20	0.5	0.1	3	77.3	86.4
vwa	10	0.7	0.15	3	79.3	86.2
cytochrome_b_C	30	0.7	0.2	3	79.2	83.1
vwc	10	0.9	0.15	3	73.9	82.6

sequences. Surprisingly, with these new scores the success rate improved only by 0.2 to 90.9%, what may suggest that more sophisticated normalization schemes should be applied.

Performance for protein families vs. domain families and local predictions. As described in Section *Prediction using a PST*, the probability the PST model assigns to a protein sequence accounts for all symbols in the sequence. Specifically, it is the product of symbol probabilities along the *whole sequences*. Therefore, it may happen that family members which are similar to other family members along a relatively small fraction of their sequence will be assigned a low overall probability. Consequently, families in which the common pattern is only a small part of the sequence (i.e. domain families) are expected to perform worse than families that are conserved along all or most of the sequence. We tested the effect of the size of the domain on the performance. We define the *coverage* as the total portion of the family sequences which is included in the multiple alignment used to define the domain or the family in the Pfam database. It ranges between 0 and 1. We expect better detection with the PST model for families with high coverage. Indeed, the performance of the PST model improves once only families with high coverage are considered (see Table 1). It detects 92% of the true positives in 108 families with more than 0.5 coverage (while the Pfam HMMs detect 94.9%). Surprisingly, the PST detects more true positives (94.2%) than the HMM (93.9%) for all 38 families with at least 0.9 coverage. This last feature of the PST model can lead to a better performance for proteins with several repeats of the same domain. Two such cases are the EF-hand and the EGF families, that were best detected with the PST model. The proteins in both these families are known for having several copies of these domains.

In view of the discussion above, the PST model is a prediction tool with a ‘global’ flavor. However, in many cases the similarity of two sequences is limited to a specific motif or domain, the detection of which may yield valuable structural and functional insights, while outside of this motif/domain the sequences may be essentially unrelated. Moreover, many proteins contain several different domains. In such cases ‘global’ prediction may not be the appropriate scheme. Local similarities may be masked by long unrelated regions. Consequently, the probability assigned to such sequences can be nearly as low as that assigned to totally unrelated sequences.

To accommodate for the multi-domain trait of proteins we have tested a variant of our prediction step. We calculate the probability only for those regions which are at least L_{domain} amino acids long, and in which at least N_{local} percent of the amino acids have probability above P_{local}. Each such subsequence is considered a domain or a motif. A continuous subsequence of at least N_{gap} amino acids with probability below P_{local} each, marks the end of a domain. We tested this variant on five families on which we performed worst (vwc, myosin_head, fibrinogen_C, cytochrome_b_C and vwa, with performance ranging between 73.9% and 79.3% using the original prediction procedure). Clearly, these parameters need to be optimized for each family separately. For each family we evaluated the performance with several sets of parameters, and picked the best one. For four families the performance improved using the new prediction procedure (see Table 2).

PST as a prediction tool and its biological implications

To demonstrate the general performance of the PST model, two typical examples are given in Figure 3 for the Neurotransmitter-gated ion-channels and the MHC I family. The cumulative log odds (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 lengths. 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, hence are well distinguished from the unrelated (effectively random) sequences. Taking a more permissive threshold for P_{min}, resulted in an improved model with better predictions (Figure 3 right), i.e. better separation between the family members and the unrelated sequences.

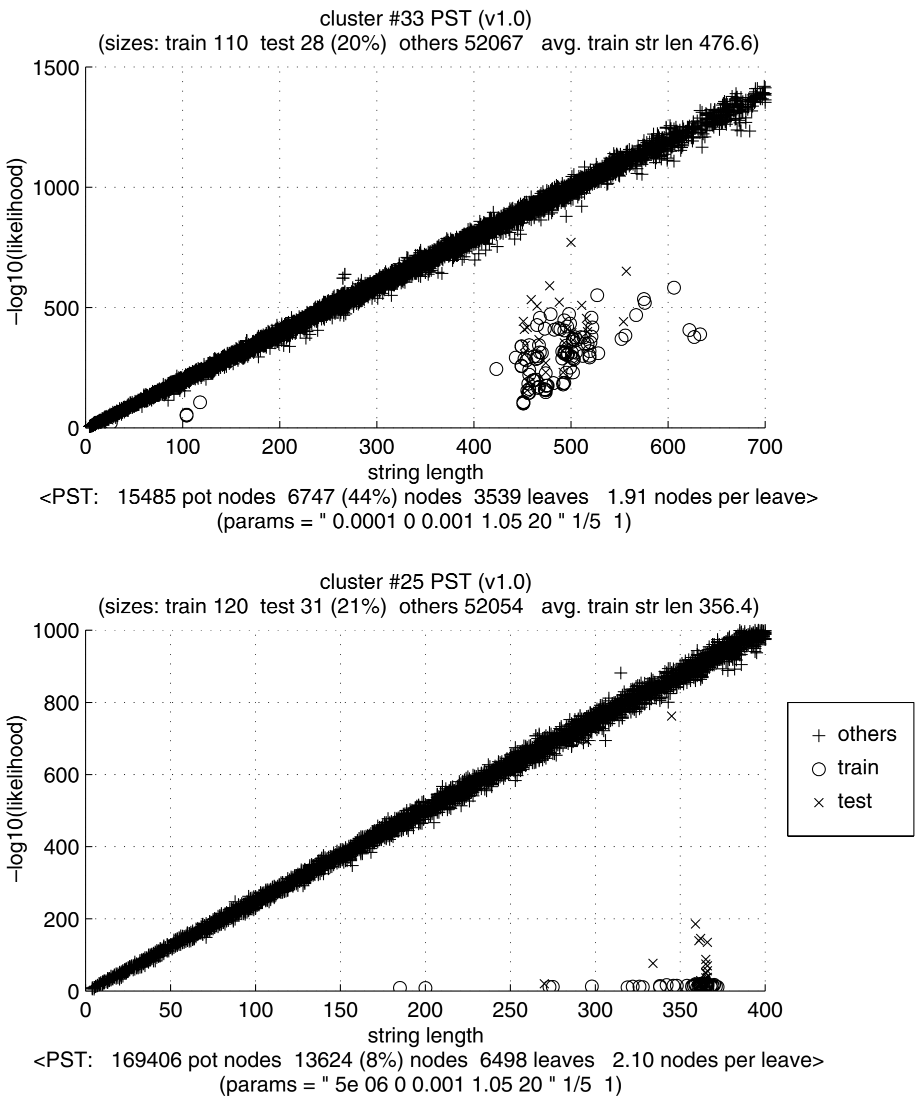
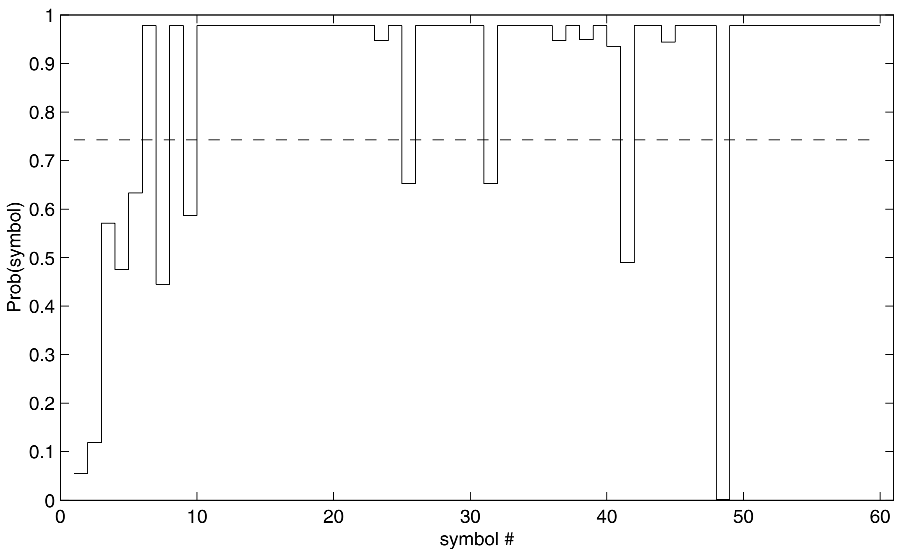


Fig. 3. Above: performance of a PST of the Neurotransmitter-gated ion-channels ($P_{\text{min}} = 0.0001$). Below: performance of a PST of the MHC I family ($P_{\text{min}} = 0.000\,005$). The graphs plot minus the log likelihood of each database sequence (including the training set and the test set) as a function of the sequence length. Likelihood is the product of symbol probabilities along the whole sequence. The PST of the MHC I family, 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 of the MHC I family was trained with the same set of parameters as the PST above, its performance graph resembled the graph above.

Family conservation. One possible implication of the PST model is that it can be used to quantify the degree of sequence conservation within protein families. Several measures are of interest here: the average coding length of unrelated sequences, the average coding length of related sequences vs. model size, and the average depth of the PST nodes used for prediction, in both cases.

The slope of the line in Figure 3 (in $\log_2$ base) is a measure of the average number of bits per letter needed to code unrelated (essentially random) sequences using a specific PST. For example, 6.64 bits per letter are required when using the PST of the Neurotransmitter-gated ion-channels family, while 8.3 bits are required when using the PST of the MHC I family (for comparison, 4.18 bits are required to code random sequences, using the background distribution[‡‡]). The slope is correlated with the *uniqueness*

[‡‡]Given any source distribution, the average source symbol coding length is lower bounded by the source entropy $-\Sigma_i p_i \log p_i$ (see, e.g. in Cover and Thomas (1991)). We define the background distribution as the amino acid distribution in the SWISSPROT 33 database.



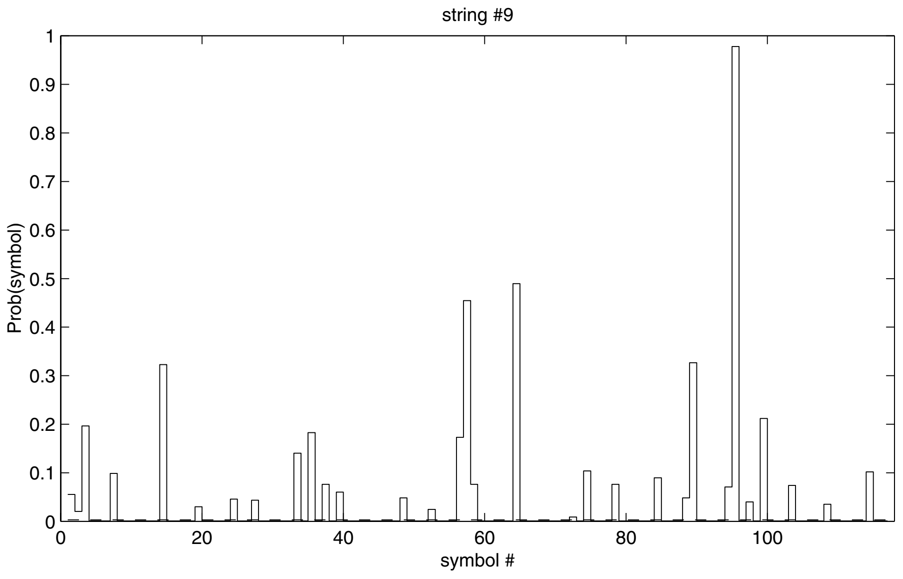


Fig. 4. Prediction using PSTs. The PST of the snake toxins family was used to calculate the probability, letter by letter, of a protein sequence (sw:P01445) from the snake toxins family test set (above), and of a protein sequence (sw:P09837) from the WAP-type (Whey Acidic Protein) family (below). The average probability is 0.75 and 0.001 respectively.

of the source distribution, and particularly, it is higher for families for which the amino acid distribution differs markedly from the overall amino acid distribution in the database (the background distribution).

On the contrary, the training set and test set can be encoded with fewer bits of information per letter. This reflects the compression obtained by the PST model and can also serve as a measure of *divergence* within a protein family. According to Shannon's source coding theorem (see footnote ‡‡) an upper bound on the source entropy can be derived by the effective (empirical) coding length of family members as provided by the PST model. A small average coding length reflects strong conservation within the protein family, while a large coding length reflects high divergence. The effective coding length of MHC I sequences, as predicted by the PST of that family, is 0.68 bits per letter while the effective coding length of the Neurotransmitter-gated ion-channels is 2.21 bits per letter (using the PST of that family with the same set of parameters), suggesting that the former is more conserved. The divergence may be related to structural diversity, suggested that the transmembrane proteins of the

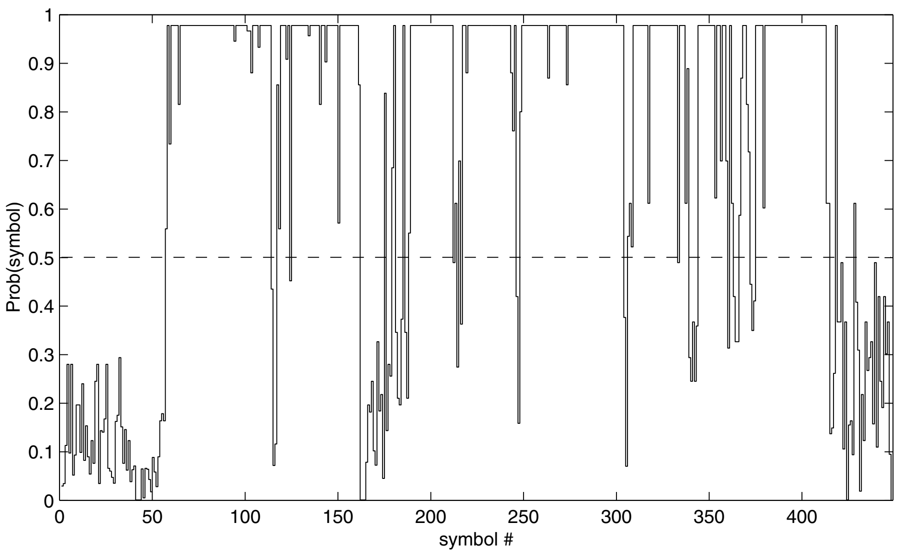


Fig. 5. Identifying significant patterns using PSTs. The PST of the zinc finger, C4 type family was used to predict significant patterns in proteins sw:P10826. The protein belongs to the test set and its average probability per symbol is 0.5. Two domains are detected (see text).

Neurotransmitter-gated ion-channels may adopt a larger variety of shapes than the MHC I family.

Prediction of significant patterns. The obvious use of the PST model is its application in prediction of family membership. A good model will discern family members from unrelated sequences. Our evaluation procedure shows that the PST model is indeed successful in this respect. The actual prediction process, letter by letter (as shown for the snake toxins family in Figure 4) can further help in assessing the relevance of high scoring hits (and possibly screen out false positives).

The PST can also be used to predict which segments of a given query sequence are suspected to be functionally or structurally important and suggest domain boundaries. These segments correspond to regions of high probability. This is demonstrated in Figure 5 for a protein from the zinc finger, C4 type family. The Pfam zf-C4 domain starts at position 79 and ends at position 154 (according to SWISSPROT annotation, this region contains two fingers in positions 81–101 and in positions 117–141). The PST assigns high probabilities to residues in this region, though its starts a bit closer to the N-terminus (at position 59). Note that another domain is predicted with high probability (between positions 190 and 413). This region corresponds to the ligand-binding domain of the nuclear hormone receptors (which according to SWISSPROT annotation starts at position 193 and ends at position 412). Out of the 139 proteins in the zf-C4 family, 127 also belong to the hormone receptors family. Therefore, both domains were recorded in the PST model during the learning phase. This explains why the PST predicted both domains with high probability (see also Section **Conclusions**). Note that along these regions not all letters are essentially of high probability, which may point to 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.

In Figure 6 the PST model of the EGF family was used to predict EGF motifs within the protein sw:fbp3_strpu (570 residues long). This protein has 8 EGF repeats (the starting positions are marked by arrows). Note that these domains correspond to regions of high probability as predicted by the PST model (excluding the first EGF domain[§§]). The probabilities are derived from the combined two-way prediction (see Section *Prediction using a PST*), to better predict domain boundaries. The prediction was smoothed using a sliding window of length 20 by averaging over the probability of symbols within each window.

[§§] A possible explanation is that this domain contains residues that are observed less frequently than in the other EGF domains. This is a consequence of the ambiguous PROSITE definition of the EGF domain C-x-C-x(5)-G-x(2)-C that allows a great diversity between the conserved cysteines and glycine.

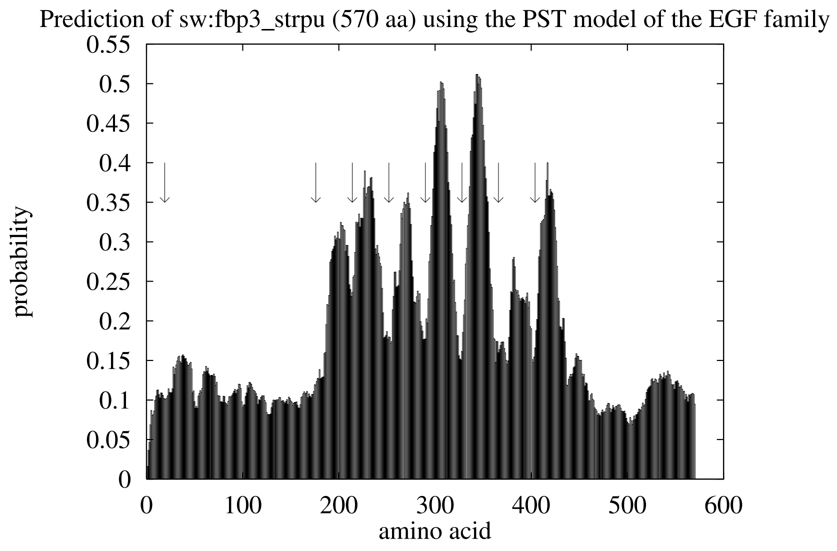


Fig. 6. Prediction of EGF domains. The PST of the EGF model was used to predict the sequence of sw:fbp3_strpu, letter by letter. The prediction combines both the PST that was trained on the training set and the PST that was trained on the reversed sequences. The starting positions of the EGF domains (according to SWISSPROT records) are marked by arrows.

Left–right causality in protein sequences? It is interesting to see if there is a sense of directionality in protein sequences. i.e. whether protein sequences are ‘generated’ by a source which has a clear directionality. Obviously proteins are assembled from left to right. They are built at the Ribosome, being translated from an RNA molecule, one amino acid at a time, as the translation process propagates from the N terminus to the C terminus. While it has been speculated by some that this process affects the folding process of the amino acids chain (Fedorov and Baldwin, 1995; Kolb *et al.*, 1995), there has been no rigorous proof whether or not left–right causality is encoded in the corresponding gene. That is, a causality that dictates the next amino acid given that we have observed a certain sequence of amino acids. Such causality may follow some physical–chemical rules and constraints that govern the processes of creating secondary structure elements in proteins, and in general, the whole folding process.

If left–right causality exists in protein sequences then one might expect differences in prediction when we propagate from left to right along the sequence compared with when we propagate along the other direction. To test this hypothesis, we trained a PST on the reversed sequences of each Pfam family. These PSTs were used to predict the sequences in the SWISSPROT database (after being reversed) in the same way described in Section *Prediction using a PST* and the performance was evaluated using the equivalence number criterion. Surprisingly, perhaps, our evaluation points that there is no difference in performance between left–right prediction and right-left prediction. Both perform almost exactly the same with some minor fluctuations. Such observation is consistent with current knowledge and perceptions of protein evolution, according to which, the genetic pressure is mainly at the protein level, and the constraints on the sequence are determined by spatial considerations, sometimes between residues which are far apart in the sequence. These perceptions are further supported by latest evidences of correlated mutations in protein sequences (Gouldson *et al.*, 1997; Pazos *et al.*, 1997), stability under circular permutations (Feng *et al.*, 1999; Hennecke *et al.*, 1999; Otzen and Fersht, 1998; Tsuji *et al.*, 1999), and experimental studies on the initiation of the protein folding process (Rabow and Scheraga, 1993).

CONCLUSIONS

In this paper we present 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 subsequences 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. Practically, 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). The tree structure itself, when examined, can tell by its very nature, what are the significant patterns found in the group of sequences from which it was constructed.

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. Our evaluation provides an objective mean of comparison between the PST model and the HMM. According to our evaluation procedure, already in its basic form the PST model outperforms classic pairwise comparison methods such as Gapped-BLAST, and performs as well as an HMM that is built in a global search mode. These results were obtained when the PST was trained using a fixed set of parameters for all families. The performance of the PST model is expected to improve if all sequences of a family are to be included in the training set. Additional improvement in the performance is expected if the parameters are tuned for each family separately, or if more permissive parameters are used in the learning procedure (see Section *Factors affecting the performance of the PST model*).

It should be noted that relaxing the parameters 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 **Results and discussion**). 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, following, say, the minimum description length principle (Rissanen, 1989). 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 its corresponding sample set, the distance (e.g. the Kullback Leibler divergence (Cover and Thomas, 1991)) between these two distributions is expected to increase and the generalization power of the model is not expected to improve further.

Another aspect which should be addressed is the multi-domain trait of protein sequences When training a PST on a set of input sequences, some of which are multi-domain, all existing domains (given that they are observed several times) will be recorded in the tree (see Section *Prediction of significant patterns*). Usually, we would like to model only one of these domains (most protein families in databases such as Pfam, ProDom, PROSITE, Domo, etc. are actually domain families). However, the PST model cannot discern on its own which parts of the input sequences are 'relevant'. A major task would be to discern those parts automatically, in an unsupervised manner.

Our evaluation results can also help to assess the effect of multiple alignments and the manual calibration on the quality of a derived HMM. Though our findings may suggest that prealignment of the input sequences is not necessary to obtain a reliable HMM, we suspect that an extended, more diverse reference set would detect differences in performance, which may suggest otherwise. The Pfam database release 1.0, associated with SWISSPROT 33 serves as a well defined and confined sample of the protein space, and therefore was chosen as a reference set. However, apparently, this reference set is composed mainly of well known families which are quite successfully detected even with a simple BLAST search (when the best query is selected). Therefore, a more elaborated test that incorporates also highly diverged families should be performed to test whether a multiple alignment of the input sequences can improve the quality of the model. A better benchmark would be the latest Pfam release (release 5.2) which is associated with SWISSPROT 38 + TrEMBL 11 databases and contains 2,128 families. However, the computation time needed to perform this evaluation test for all 2,128 families prevents us at the moment from running the evaluation procedure again.

Our assessment points the potential of the PST model as a predictive tool for sequence and genome analysis. It can discriminate family members from non-family proteins and identify significant segments. The model can also be applied for detection of approximate repeats and approximate tandem repeats. All these aspects can be achieved in an efficient manner. For example, while searching the SWISSPROT database with a typical HMM may take several hours (on a pentium II 300 Mhz), prediction with a PST takes only a few minutes for all sequences in the database. As the size of sequence database increases rapidly, such efficient search and prediction methods come to play an important role in large-scale analyses of complete genomes.

Finally, the method does not assume any further biological information, but such information can be easily incorporated to improve its sensitivity. We have suggested a variant of the PST model that is based on biological considerations (see Section *A variant–incorporate biological* *considerations in the PST model*). Indeed, when biological knowledge is taken into account (such as the amino acid substitution probabilities) similar or better separation is usually achieved using smaller trees.

Similarly, it is possible to tune the algorithm to a 'local' search mode. While this variant is a heuristic that introduces more parameters that need to be optimized for each family, it opens the possibility of local predictions with the PST model. We are currently testing other variants, based on more rigorous approaches. Additional criteria that should be incorporated may include, for example, the level of the node used to predict each letter (that is, the depth of the node in the tree, or the number of edges from that node to the root). By weighing the probability by the level of the corresponding node, the predictions which are made based on nodes that are closer to the root become less significant than predictions based on deeper nodes. This scheme is expected to deal better with chance similarities. Further improvements should also 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.

ACKNOWLEDGMENTS

The idea to apply PSTs to model protein families was originally brought up by Naftali Tishby and we thank him for helpful suggestions. We thank Dana Ron for valuable advice and for critically reading an early version of this manuscript, and our anonymous referees for their useful comments. We also thank Amnon Barak for making the MOSIX parallel system available for us, and Richard Hughey for the SAM software.

GB is supported by a grant from the ministry of science, Israel, GY is supported by a Burroughs-Wellcome Fellowship from the Program in Mathematics and Molecular Biology (PMMB).

REFERENCES

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

Altschul,S.F. (1991) Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, **219**, 555–565.

Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Apostolico,A. and Bejerano,G. (2000) Optimal amnestic probabilistic automata or how to learn and classify proteins in linear time and space. *J. Comp. Biol.*, **7(3/4)**, 381–343.

Attwood,T.K., Flower,D.R., Lewis,A.P., Mabey,J.E., Morgan,S.R., Scordis,P., Selley,J. and Wright,W. (1999) PRINTS prepares for the new millennium. *Nucleic Acids Res.*, **27**, 220–225.

Bailey,T.L. and Elkan,C. (1995) The value of prior knowledge in discovering motifs with MEME. *ISMB*, **3**, 21–29.

Bairoch,A. and Apweiler,R. (1999) The SWISS-PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Res.*, **27**, 49–54.

Bates,P.A. and Sternberg,M. J.E. (1991) Model building by comparison at CASP3: using expert knowledge and computer automation. *Proteins*, (Suppl. 3), 47–54.

Bateman,A., Birney,E., Durbin,R., Eddy,S.R., Finn,R.D. and Sonnhammer,E.L. (1999) Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Res.*, **27**, 260–262.

Corpet,E., Gouzy,J. and Kahn,D. (1999) Recent improvements of the ProDom database of protein domain families. *Nucleic Acids Res.*, **27**, 263–267.

Cover,T.M. and Thomas,J.A. (1991) *Elements of Information Theory*. John Wiley and Sons, New York.

Dempster,A.P., Laird,N.M. and Rubin,D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc.*, B **39**, 1–38.

Eddy,S. (1998) HMMER user's guide: biological sequence analysis using profile hidden Markov models, http://hmmer.wustl.

Fedorov,A.N. and Baldwin,T.O. (1995) Contribution of cotranslational folding to the rate of formation of native prote in structure. *Proc. Natl. Acad. Sci. USA*, **92**, 1227–1231.

Feng,Y., Minnerly,J.C., Zurfluh,L.L., Joy,W.D., Hood,W.F., Abegg,A.L., Grabbe,E.S., Shieh,J.J., Thurman,T.L., McKearn,J.P. and McWherter,C.A. (1999) Circular permutation of granulocyte colony-stimulating factor. *Biochemistry*, **38**, 4553–4563.

Gillman,D. and Sipser,M. (1994) Inference and minimization of hidden Markov chains. In *Proceedings of the 7th Annual Workshop on Computational Learning Theory* pp. 147–158.

Gouldson,P.R., Bywater,R.P. and Reynolds,C.A. (1997) Correlated mutations amongst the external residues of G-protein coupled receptors. *Biochem. Soc. Trans.*, **25**, 529.

Gracy,J. and Argos,P. (1998) Automated protein sequence database classification I. Integration of copositional similarity search, local similarity search and multiple sequence alignment II. Delineation of domain boundaries from sequence similarity. *Bioinformatics*, **14**, 164–187.

Gribskov,M., Mclachlen,A.D. and Eisenberg,D. (1987) Profile analysis: detection of distantly related proteins. *Proc. Natl. Acad. Sci. USA*, **84**, 4355–4358.

Gusfield,D. (1997) *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge.

Hanke,K., Beckmann,G., Bork,P. and Reich,J.G. (1996) Self-organizing hierarchic networks for pattern recognition in protein sequence. *Protein Sci.*, **5**, 72–82.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, **89**, 10 915–10 919.

Henikoff,J.G. and Henikoff,S. (1996) Using substitution probabilities to improve position-specific scoring matrices. *Comput. Appl. Biosci.*, **12**, 135–143.

Henikoff,J.G., Henikoff,S. and Pietrokovski,S. (1999) New features of the blocks database serves. *Nucleic Acids Res.*, **27**, 226–228.

Hennecke,J., Sebbel,P. and Glockshuber,R. (1999) Random circular permutation of DsbA reveals segments that are essential for protein folding and stability. *J. Mol. Biol.*, **286**, 1197–1215.

Higgins,D.G., Thompson,J.D. and Gibson,T.J. (1996) Using CLUSTAL for multiple sequence alignments. *Meth. Enzymol.*, **266**, 383–402.

Hofmann,K., Bucher,P., Falquet,L. and Bairoch,A. (1999) The PROSITE database, its status in 1999. *Nucleic Acids Res.*, **27**, 215–219.

Hughey,R. and Krogh,A. (1998) SAM: sequence alignment and modeling software system. *Technical Report UCSC-CRL-96-22* University of California, Santa Cruz, CA.

Jonassen,I., Collins,J.F. and Higgins,D.G. (1995) Finding flexible patterns in unaligned protein sequences. *Protein Sci.*, **4**, 1587–1595.

Kolb,V.A., Markeyev,E.V., Kommer,A. and Spirin,A.S. (1995) Cotranslational folding of proteins. *Biochem. Cell Biol.*, **73**, 1217–1220.

Krogh,A., Brown,M., Mian,I.S., Sjölander,K. and Haussler,D. (1996) Hidden Markov models in computational biology: application to protein modeling. *J. Mol. Biol.*, **235**, 1501–1531.

Lawrence,C.E., Altschul,S.F., Boguski,M.S., Liu,J.S., Neuwald,A.F. and Wootton,J.C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Nevill-Manning,C.G., Wu,T.D. and Brutlag,D.L. (1998) Highly specific protein sequence motifs for genome analysis. *Proc. Natl. Acad. Sci. USA*, **95**, 5865–5871.

Neuwald,A.F. and Green,P. (1994) Detecting patterns in protein sequences. *J. Mol. Biol.*, **239**, 698–712.

Otzen,D.E. and Fersht,A.R. (1998) Folding of circular and permuted chymotrypsin inhibitor 2: retention of the folding nucleus. *Biochemistry*, **37**, 8139–8146.

Pazos,F., Helmer-Citterich,M., Ausiello,G. and Valencia,A. (1997) Correlated mutations contain information about protein–protein interaction. *J. Mol. Biol.*, **271**, 511–523.

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

Ponting,C.P., Schultz,J., Milpetz,F. and Bork,P. (1999) SMART: identification and annotation of domains from signaling and extracellular protein sequences. *Nucleic Acids Res.*, **27**, 229–232.

Rabow,A.A. and Scheraga,H.A. (1993) Lattice neural network minimization. Application of neural network optimization for locating the global-minimum conformations of proteins. *J. Mol. Biol.*, **232**, 1157–1168.

Rissanen,J. (1989) Stochastic complexity in statistical inquiry. *World Scientific*.

Ron,D., Singer,Y. and Tishby,N. (1996) The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, **25**, 117–149.

Samudrala,R. and Moult,J. (1997) Handling context-sensitivity in protein structures using graph theory: bonafide prediction. *Proteins*, (Suppl. 1), 43–49.

Smith,H.O., Annau,T.M. and Chandrasegaran,S. (1990) Finding sequence motifs in groups of functionally related proteins. *Proc. Natl. Acad. Sci. USA*, **87**, 826–830.

Suyama,M., Nishioka,T. and Oda,J. (1995) Searching for common sequence patterns among distantly related proteins. *Protein Eng.*, **8**, 1075–1080.

Tsuji,T., Yoshida,K., Satoh,A., Kohno,T., Kobayashi,K. and Yanagawa,H. (1999) Foldability of barnase mutants obtained by permutation of modules or secondary structure units. *J. Mol. Biol.*, **286**, 1581–1596.

Yona,G., Linial,N. and Linial,M. (1999) ProtoMap: automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. *Proteins*, **37**, 360–378.