BIOINFORMATICS ORIGINAL PAPER Vol. 22 no. 11 2006, pages 1302–1307
doi:10.1093/bioinformatics/btl088

*Sequence analysis*

# A generalization of the PST algorithm: modeling the sparse nature of protein sequences

Florencia G. Leonardi

Instituto de Matemática e Estatística, Universidade de São Paulo, Rua do Matão 1010 CEP 05508-090, São Paulo, Brazil

Received on October 26, 2005; revised on February 19, 2006; accepted on March 6, 2006
Advance Access publication March 9, 2006
Associate Editor: Thomas Lengauer

**ABSTRACT**

**Motivation:** A central problem in genomics is to determine the function of a protein using the information contained in its amino acid sequence. Variable length Markov chains (VLMC) are a promising class of models that can effectively classify proteins into families and they can be estimated in linear time and space.

**Results:** We introduce a new algorithm, called Sparse Probabilistic Suffix Trees (SPST), that identifies equivalences between the contexts of a VLMC. We show that, in many cases, the identification of these equivalences can improve the classification rate of the classical Probabilistic Suffix Trees (PST) algorithm. We also show that better classification can be achieved by identifying representative fingerprints in the amino acid chains, and this variation in the SPST algorithm is called F-SPST.

**Availability:** The SPST algorithm can be freely downloaded from the site http://www.ime.usp.br/~leonardi/spst/

**Contact:** leonardi@ime.usp.br

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 INTRODUCTION

Efficient family classification of newly discovered protein sequences is a central problem in bioinformatics (Rust *et al*., 2002). Nowadays, the most popular methods for generating a hypothesis about the function of a protein are BLAST and hidden Markov models (HMM). These methods, although very useful, are time-consuming considering the actual size of the databases. Recently, Bejerano and Yona (2001) applied VLMC in protein classification. They showed that variable length Markov chains (VLMC) detects much more related sequences than Gapped-BLAST and it is almost as sensitive as HMM. A major advantage of VLMC is that it does not depend on multiple alignments of the input sequences and it can be estimated in linear time and space (Apostolico and Bejerano, 2000).

In the present work we affront the problem of protein family classification creating a model for each protein family $\mathcal{F}$ and testing if each query sequence belongs to $\mathcal{F}$ or not. The model is a variation of VLMC, called sparse Markov chains (SMC). These two models are a class of stochastic processes that relies on the estimation of conditional probabilities of symbols in a sequence given the preceding symbols. The remarkable property of VLMC is that the number of symbols in the conditional sequence is variable, and depends on the symbols involved. This property attempts to capture variable dependencies in different configurations of amino acids in a protein family. A VLMC model can be represented by a tree in which each branch corresponds to a conditional sequence of symbols. For example, a VLMC tree for a set of sequences in the binary alphabet {0, 1} is illustrated in Figure 1. The conditional probabilities are associated to each branch, indicating the probability of the next symbol in a query sequence given that the preceding symbols match the sequence in the branch. The property of variable length in the conditional sequences is maintained in SMC. The difference between SMC and VLMC is that in SMC the conditional sequences are given by sequences of subsets of amino acids. This property attempts to identify which positions are more relevant than others in the configurations of amino acids that define the conditional sequences. The sequences composed by subsets of amino acids are called sparse.

VLMC was first introduced in Rissanen (1983) as an universal model for data compression. Rissanen also introduced an estimation algorithm for this class of models called Context Algorithm (CA). It was proven in Bülhmann and Wyner (1999) that this algorithm is consistent. This means that if we suppose that the data were generated with a VLMC, with a sufficiently large sample the CA will find the real model. PST is another algorithm to estimate a VLMC (Ron *et al*., 1996), and it has the advantage of being computationally more economical than the CA (Apostolico and Bejerano, 2000). The PST algorithm was successfully used in protein classification, even though its performance decreases with less conserved families (Bejerano and Yona, 2001). For that reason some attempts were made to use VLMC related models for sparse sequences (Eskin *et al*., 2000; Bourguignon and Nicolas, 2004). Although very attractive, these two methods have the major disadvantage of having computationally very expensive algorithms.

In this work we introduce a variation in the PST algorithm to estimate an SMC for sparse sequences, called SPST. Another variation is introduced to improve the classification of PST and SPST, and it is called F-SPST. The paper is organized as follows. In Section 2 the formal definitions of VLMC and SMC and a simple example are given. In Section 3 we introduce the two variations in the PST algorithm and we explain how to classify protein sequences using these algorithms. In Section 4 we present the experimental results and in Section 5 we discuss these results and some ideas for future work.

## 2 THEORY

Let $\mathcal{A}$ be a finite alphabet (e.g. the alphabet of twenty amino acids for protein sequences, or four nucleotides for DNA sequences).

 

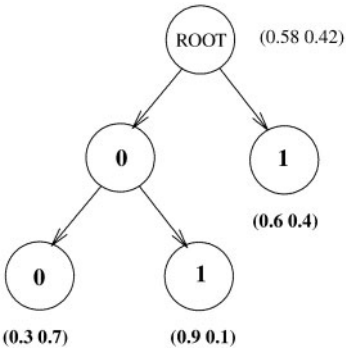


**Fig. 1.** A tree representation of a VLMC over the alphabet $\mathcal{A} = \{0, 1\}$. The branches represent the sequences that are relevant to predict the next symbol in a sequence, and the probability distribution associated to each branch gives the probability of having 0 or 1 given that the preceding symbols match the sequence in the branch. For example, the probability of having symbol 0 in any position of a query sequence given that the preceding symbols are 0 and 1 respectively is $P(0|10) = 0.9$.

Let $(X_n)_{n\in\mathbb{N}}$ be a stationary stochastic process with values $X_n \in \mathcal{A}$. We say that the process $(X_n)$ is a variable length Markov chain (VLMC) if

$$P[X_n = x_n \,|\, X_0 = x_0, \ldots, X_{n-1} = x_{n-1}] = P[X_n = x_n \,|\, X_{n-\ell} = x_{n-\ell}, \ldots, X_{n-1} = x_{n-1}],$$

where $\ell$ is a function of the sequence $x_0, \ldots, x_{n-1}$; i.e. $\ell : \cup_{k=0}^{\infty} \mathcal{A}^k \to \mathbb{N}$. This function is called length function and the finite sequences $(x_{n-\ell}, \ldots, x_{n-1})$ are called contexts. As we are assuming a time homogeneous process, the length function does not depend on $n$, so we simply denote the contexts by $(x_{-\ell}, \ldots, x_{-1})$.

In a VLMC the set of contexts has the suffix property: this means that no context is a suffix of another context. This makes it possible to define without ambiguity the probability distribution of the next symbol. The suffix property also permits to represent the set of contexts as a tree. In this tree, each context $(x_{-\ell}, \ldots, x_{-1})$ is represented by a complete branch, in which the first node on top is $x_{-1}$ and so on until the last element $x_{-\ell}$ which is represented by the terminal node of the branch. For example, in Figure 1 the set of contexts is {00, 10, 1}. A complete description of VLMC can be found in Bülhmann and Wyner (1999).

We introduce here the definition of an SMC. An SMC is a VLMC in which some contexts can be grouped together into an equivalence class. In a SMC the probability transitions are given by

$$P[X_n = x_n \,|\, X_0 = x_0, \ldots, X_{n-1} = x_{n-1}] = P[X_n = x_n \,|\, X_{n-\ell} \in A_{n-\ell}, \ldots, X_{n-1} \in A_{n-1}],$$

where $A_i \subset \mathcal{A}$ for all $i = n - \ell, \ldots, n-1$. This relation induces a partition of the set of contexts of the corresponding VLMC. Then, the contexts in an SMC are given by the equivalence classes denoted by the sequences of subsets $(A_{-\ell}, \ldots, A_{-1})$. A tree representation for this type of contexts can be seen in Figure 2. To each node of the tree is associated a conditional probability distribution over the next symbol given the context represented by the node. For example, if we want to compute the probability of the sequence 01001 in the model given by Figure 2 we have $P(01001) = P(0) \times P(1\,|\,0) \times P(0\,|\,01) \times P(0\,|\,10) \times P(1\,|\,00) = 9/16 \times 7/16 \times 3/10 \times 9/10 \times 7/10$.

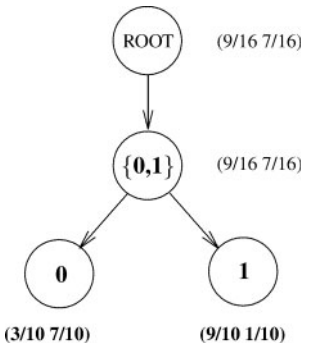


**Fig. 2.** A sparse tree over the alphabet $\mathcal{A} = \{0, 1\}$. The set of contexts of the associated VLMC is {00, 01, 10, 11}, and the equivalence classes are {00, 01} and {10, 11}. This means that $P(X_n = x_n \,|\, X_{n-2} = 0, X_{n-1} = 0) = P(X_n = x_n \,|\, X_{n-2} = 0, X_{n-1} = 1)$ and $P(X_n = x_n \,|\, X_{n-2} = 1, X_{n-1} = 0) = P(X_n = x_n \,|\, X_{n-2} = 1, X_{n-1} = 1)$, for all $x_n \in \mathcal{A}$.

In this work we propose a variation in the PST algorithm to identify the equivalence classes given by the sparse contexts $(A_{-\ell}, \ldots, A_{-1})$. This algorithm is presented in the next section.

## 3 ALGORITHM

Let $p^1, p^2, \ldots, p^m$ be the sample sequences belonging to a protein family $\mathcal{F}$. For each $i = 1, \ldots, m$ we denote $p^i = p^i_1, \ldots, p^i_{n_i}$, where $n_i$ is the length of sequence $p^i$.

Given a sparse context $w = (A_{-\ell}, \ldots, A_{-1})$ and a symbol $a \in \mathcal{A}$, we denote by $N(w, a)$ the number of occurrences of context $w$ followed by symbol $a$ in the sample set. That is,

$$N(w, a) = \sum_{i=1}^{m} \sum_{j=\ell+1}^{n_i} \mathbf{1}_{\{p^i_{j-\ell} \in A_{-\ell}, \ldots, p^i_{j-1} \in A_{-1}, p^i_j = a\}},$$

where $\mathbf{1}$ is the indicator function that takes value 1 if $p^i_{j-\ell} \in A_{-\ell}, \ldots, p^i_{j-1} \in A_{-1}, p^i_j = a$, or 0 otherwise. We also define

$$N(w, \cdot) = \sum_{a\in\mathcal{A}} N(w, a).$$

Given a sparse context $w = (A_{-\ell}, \ldots, A_{-1})$ and symbols $a_1$, $a_2 \in \mathcal{A}$ we denote by $a_1 w$ and $a_2 w$ the sparse contexts $(\{a_1\}, A_{-\ell}, \ldots, A_{-1})$ and $(\{a_2\}, A_{-\ell}, \ldots, A_{-1})$, respectively. We also denote by $[a_1, a_2]w$ the sparse context $(\{a_1, a_2\}, A_{-\ell}, \ldots, A_{-1})$. Using this notation we define the operator $\Delta_w(a_1, a_2)$ as the logarithm of the ratio between the estimated probability of the sequences in the model that has the contexts $a_1 w$ and $a_2 w$ as equivalent and the model that distinguishes the two contexts as different. That is,

$$\begin{aligned}\Delta_w(a_1, a_2) &= \log \prod_{a\in\mathcal{A}} \frac{\hat{\boldsymbol{P}}(a \,|\, [a_1, a_2]w)^{N([a_1,a_2]w,\,a)}}{\hat{\boldsymbol{P}}(a \,|\, a_1 w)^{N(a_1 w,\,a)}\,\hat{\boldsymbol{P}}(a \,|\, a_2 w)^{N(a_2 w,\,a)}} \\ &= \sum_{a\in A} N([a_1, a_2]w, a) \log\left(\frac{N([a_1,a_2]w,\,a)}{N([a_1,a_2]w,\,\cdot)}\right) \\ &\quad - \sum_{a\in A} N(a_1 w, a) \log\left(\frac{N(a_1 w,\,a)}{N(a_1 w,\,\cdot)}\right) \\ &\quad - \sum_{a\in A} N(a_2 w, a) \log\left(\frac{N(a_2 w,\,a)}{N(a_2 w,\,\cdot)}\right).\end{aligned}$$

Note that $N([a_1, a_2]w, a) = N(a_1 w, a) + N(a_2 w, a)$ and $N([a_1, a_2]\, w, \cdot) = N(a_1 w, \cdot) + N(a_2 w, \cdot)$.

Using the preceding definitions we can specify how the SPST algorithm works. The free parameters that must be specified by the user are the maximum depth of the tree, $L_{\max}$, the minimum number of times that a sparse context has to be seen in the sample to be considered, $N_{\min}$, and a cutoff parameter that establishes the equivalence between two contexts, $r_{\max}$. The SPST algorithm works as follows. It starts with a tree $T$ consisting of a single root node. At each step, for every terminal node in $T$ labeled by a sparse sequence $w$ with depth less than $L$ and for every symbol $a \in \mathcal{A}$, the child $a$ is added to node $w$ if $N(aw,\cdot) \geq N_{\min}$. Then, for every pair of children of $w$, $a_1$ and $a_2$, we test the equivalence of the contexts $a_1w$ and $a_2w$ using the $\Delta$ operator. That is, we compute $\Delta_w(a_1, a_2)$ for every pair of symbols $(a_1, a_2) \in \mathcal{A}^2$ added to node $w$, and choose the minimum between all the pairs. If this minimum is smaller than $r_{\max}$, the corresponding nodes are merged together into a single node. This procedure is iterated with the new set of children of $w$ until no more nodes can be merged. Taking the minimum of $\Delta_w(a_1, a_2)$ between all the possible pairs $(a_1, a_2)$ ensures the independence of the order in which the tests are performed. To conclude the construction of the tree we assign to each node a transition probability distribution. This distribution gives the probability of a symbol in $\mathcal{A}$ given the sparse context between the node and the root of the tree. The transition probabilities are estimated using the maximum likelihood estimates. That is, given a sparse context $w = (A_{-\ell}, \ldots, A_{-1})$, the estimated probability of a symbol $a \in \mathcal{A}$ given the context $w$ is given by

$$\hat{P}(X_n = a \,|\, X_{n-\ell} \in A_{n-\ell}, \ldots, X_{n-1} \in A_{n-1}) = \frac{N(w,a)}{N(w,\cdot)}$$

Given a sparse context $w = (A_{-\ell}, \ldots, A_{-1})$ we denote by $w'$ the sparse context $(A_{-\ell+1}, \ldots, A_{-1})$ and by $|w|$ the length of context $w$. Summarizing the steps of the SPST algorithm, we have the following:

**SPST-Algorithm** ($N_{\min}$, $r_{\max}$, $L_{\max}$)

(1) *Initialization*: let $T$ be a tree consisting of a single root node, and let

$$\bar{S} = \{a : a \in \mathcal{A} \text{ and } N(a,\cdot) \geq N_{\min}\}$$

(2) *Iteration*: while $\bar{S} \neq \emptyset$ do

(a) Remove $u$ of $\bar{S}$ and add $u$ to $T$. Then remove all sparse contexts $w \in \bar{S}$ such that $w' = u'$ and add them to $T$. Note that the context $w$ is of the form $a_iu'$ for some symbol $a_i \in \mathcal{A}$. Let $C$ denote the set of contexts added to $T$ in this step.

(b) Compute

$$r = \min\{\Delta_{u'}(a_i, a_j) : a_iu', a_ju' \in C\},$$

and

$$(a_{i^*}, a_{j^*}) = \text{argmin}\{\Delta_{u'}(a_i, a_j) : a_iu', a_ju' \in C\}.$$

(c) If $r < r_{\max}$ merge $a_{i^*}$ and $a_{j^*}$ in a single node. Replace the contexts $a_{i^*}u'$ and $a_{j^*}u'$ in $C$ by the context $[a_{i^*}, a_{j^*}]u'$.

(d) Repeat steps (b) and (c) until no more changes can be made in $C$.

(e) For each sparse context $w \in C$, if $|w| < L_{\max}$ then add the set $\{aw : a \in \mathcal{A} \text{ and } N(aw,\cdot) \geq N_{\min}\}$ (if any) to $\bar{S}$.

(3) *Estimation of the transition probabilities*: assign to each node in $T$, associated with a sparse context $w$, the transition probability distribution over $\mathcal{A}$ given by

$$\left\{\frac{N(w,a)}{N(w,\cdot)} : a \in A\right\}.$$

### 3.1 Application of the SPST algorithm to classify protein sequences

Given a protein family $\mathcal{F}$ and a new sequence of amino acids $p = p_1, \ldots, p_n$, we want to know if $p$ belongs to $\mathcal{F}$ or not. To answer this question we first construct a model for the family $\mathcal{F}$ using the sequences already classified into the family. Then we compute a score, and depending on this value we classify the sequence $p$ as belonging to the family $\mathcal{F}$ or not. The model constructed for the family $\mathcal{F}$ is an SMC model obtained by estimating the sparse contexts and the transition probabilities given by the SPST algorithm, as explained above. There is no restriction about the minimum number of sequences needed to estimate an SMC, because SPST will only estimate the next symbol probability conditioned on sequences that appear a minimum number of times in the sample. But it is also true that with a very small family the estimation of the probability distributions could be very poor. For that reason we only test the SPST algorithm with families containing more than 10 sequences, as explained in the next section.

Given a query sequence $p$, its score in the estimated SMC model for $\mathcal{F}$ is given by

$$S(p) = \frac{1}{n}\log[\bar{P}(p)],$$

where $\bar{P}$ is the smoothed probability distribution derived from $\hat{P}$. That is

$$\bar{P}(p) = \prod_{i=1}^{n}((1 - |\mathcal{A}|\gamma_{\min})\hat{P}(p_i \,|\, w(p_1, \ldots, p_{i-1})) + \gamma_{\min}),$$

where $w(p_1, \ldots, p_{i-1})$ is the sparse context corresponding to the sequence $p_1, \ldots, p_{i-1}$. The parameter $\gamma_{\min}$ is a smoothing parameter to avoid zero probabilities, and therefore, a $-\infty$ score.

Sometimes the region of high similarity between the sequences in a protein family is considerably smaller than the length of the sequences. This is because a protein sequence can be composed by several domains, performing different functions in the cell. Then, computing the score $S$ over the entire sequence $p$ may not be appropriate. For this reason we propose a change in the computation of the score $S$ and called it $S'$. In this case we fix an integer $M$, and for sequences with length $n > M$ we compute the score $S'(p)$ by

$$S'(p) = \max_{j=0,\ldots,n-M}\left\{\frac{1}{M}\log[\bar{P}(p_{j+1} \cdots p_{j+M})]\right\}.$$

In the case $n \leq M$, the score is computed using $S$ as before. The algorithm that implements the score $S'$ is called F-SPST.

## 4 IMPLEMENTATION

In order to test our algorithm and to compare it with PST published results (Bejerano and Yona, 2001) we use protein families of the Pfam database (Bateman *et al*., 2004) release 1.0. This database contains 175 families derived from the Swiss-Prot 33 database

**Table 1.** Performance comparison between PST, SPST and F-SPST.

| Family | Size | Percentage of true pos. detected by PST | No. PST false positives | Percentage of true pos. detected by SPST | No. SPST false positives | Percentage of true pos. detected by F-SPST | No. F-SPST false positives |
|---|---|---|---|---|---|---|---|
| 7tm_1 | 515 | 93.0 | 36 | 96.3 | 19 | 97.7 | 12 |
| 7tm_2 | 36 | 94.4 | 2 | 97.2 | 1 | 100.0 | 0 |
| 7tm_3 | 12 | 83.3 | 2 | 100.0 | 0 | 100.0 | 0 |
| AAA | 66 | 87.9 | 8 | 90.9 | 6 | 93.9 | 4 |
| ABC_tran | 269 | 83.6 | 44 | 85.9 | 38 | 89.2 | 29 |
| actin | 142 | 97.2 | 4 | 97.2 | 4 | 99.3 | 1 |
| adh_short | 180 | 88.9 | 20 | 89.4 | 19 | 92.8 | 13 |
| adh_zinc | 129 | 95.3 | 6 | 91.5 | 11 | 95.4 | 6 |
| aldedh | 69 | 87.0 | 9 | 89.9 | 7 | 92.8 | 5 |
| alpha-amylase | 114 | 87.7 | 14 | 91.2 | 10 | 94.7 | 6 |
| aminotran | 63 | 88.9 | 7 | 88.9 | 7 | 90.5 | 6 |
| ank | 83 | 88.0 | 10 | 86.8 | 11 | 86.6 | 11 |
| arf | 43 | 90.7 | 4 | 93.0 | 3 | 93.0 | 3 |
| asp | 72 | 83.3 | 12 | 90.3 | 7 | 91.7 | 6 |
| ATP-synt_A | 79 | 92.4 | 6 | 94.9 | 4 | 97.5 | 2 |
| ATP-synt_ab | 180 | 96.7 | 6 | 96.7 | 6 | 98.3 | 3 |
| ATP-synt_C | 62 | 91.9 | 5 | 95.2 | 3 | 95.2 | 3 |
| beta-lactamase | 51 | 86.3 | 7 | 90.2 | 5 | 94.1 | 3 |
| bZIP | 95 | 89.5 | 10 | 90.5 | 9 | 93.7 | 6 |
| C2 | 78 | 92.3 | 6 | 92.3 | 6 | 96.2 | 3 |
| cadherin | 31 | 87.1 | 4 | 87.1 | 4 | 93.6 | 2 |
| cellulase | 40 | 85.0 | 6 | 85.0 | 6 | 90.0 | 4 |
| cNMP_binding | 42 | 92.9 | 3 | 92.9 | 3 | 95.2 | 2 |
| COesterase | 61 | 91.7 | 5 | 90.0 | 6 | 93.3 | 4 |
| connexin | 40 | 97.5 | 1 | 97.5 | 1 | 100.0 | 0 |
| copper-bind | 61 | 95.1 | 3 | 96.7 | 2 | 98.4 | 1 |
| COX1 | 80 | 83.8 | 13 | 85.0 | 12 | 85.0 | 12 |
| COX2 | 109 | 98.2 | 2 | 98.2 | 2 | 98.2 | 2 |
| cpn10 | 57 | 93.0 | 4 | 98.3 | 1 | 98.3 | 1 |
| cpn60 | 84 | 94.0 | 5 | 94.0 | 5 | 95.2 | 4 |
| crystall | 53 | 98.1 | 1 | 98.1 | 1 | 98.1 | 1 |
| cyclin | 80 | 88.8 | 9 | 86.3 | 11 | 91.3 | 7 |
| Cys-protease | 91 | 87.9 | 11 | 89.0 | 10 | 94.5 | 5 |
| cystatin | 53 | 92.5 | 4 | 90.6 | 5 | 90.6 | 5 |
| Cys_knot | 61 | 93.4 | 4 | 93.4 | 4 | 93.4 | 4 |
| cytochrome_b_C | 130 | 79.2 | 27 | 90.8 | 12 | 87.7 | 16 |
| cytochrome_b_N | 170 | 98.2 | 3 | 97.1 | 5 | 98.2 | 3 |
| cytochrome_c | 175 | 93.7 | 11 | 94.3 | 10 | 95.4 | 8 |
| DAG_PE-bind | 68 | 89.7 | 7 | 89.7 | 7 | 89.7 | 7 |
| DNA_methylase | 48 | 83.3 | 8 | 83.3 | 8 | 93.8 | 3 |
| DNA_pol | 46 | 80.4 | 9 | 82.6 | 8 | 82.6 | 8 |
| dsrm | 14 | 85.7 | 2 | 85.7 | 2 | 85.7 | 2 |
| E1-E2_ATPase | 102 | 93.1 | 7 | 95.1 | 5 | 96.1 | 4 |
| efhand | 320 | 92.2 | 25 | 92.2 | 25 | 92.2 | 25 |
| EGF | 169 | 89.4 | 18 | 88.8 | 19 | 89.4 | 18 |
| enolase | 40 | 100.0 | 0 | 97.5 | 1 | 100.0 | 0 |
| fer2 | 88 | 94.3 | 5 | 95.5 | 4 | 97.7 | 2 |
| fer4 | 152 | 88.2 | 18 | 90.1 | 15 | 91.5 | 13 |
| fer4_NifH | 49 | 95.9 | 2 | 98.0 | 1 | 98.0 | 1 |
| FGF | 39 | 97.4 | 1 | 100.0 | 0 | 100.0 | 0 |

Families are ordered alphabetically, and correspond to the first 50 families with more than 10 sequences in the Pfam database, version 1.0. The number of sequences in each family is given in the second column. The other six columns, two for each algorithm, indicate the percentage of true positives detected with respect to the size of each family and the number of false positives, when using the equivalence number criterion. This method sets the threshold at the point where the number of false positives equals the number of false negatives. PST results where taken from (Bejerano and Yona, 2001). The set of parameters to train the SPST and F-SPST algorithms where: $L = 20$, $N_{\min} = 3$, $\gamma_{\min} = 0.001$ and $r_{\max} = 3.8$. The value of $M$ used in the F-SPST algorithm was $M = 80$ for all families.

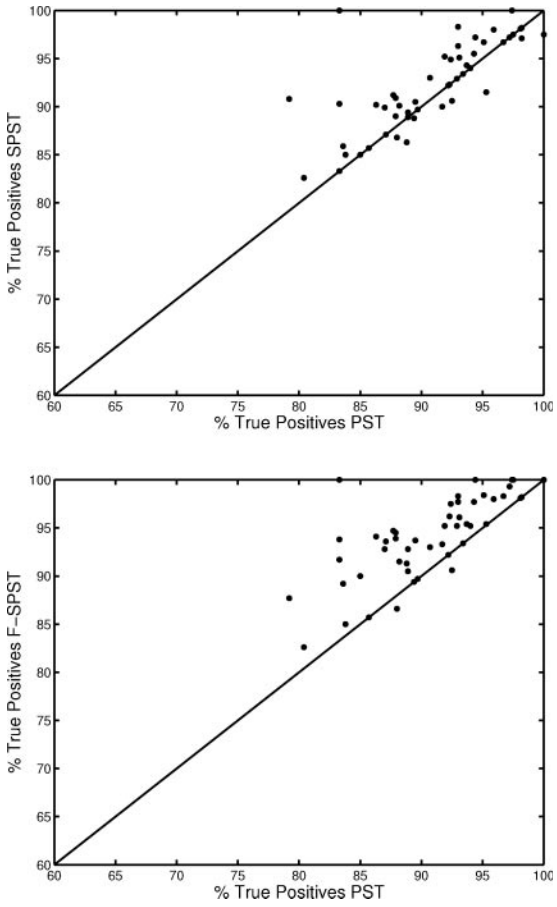


**Fig. 3.** Scatter-plots of performances from PST, SPST and F-SPST protein classification methods. Above: SPST versus PST. Below: F-SPST versus PST.

(Boeckmann *et al.*, 2004). We selected the first 50 families (in alphabetical order) that contain more than 10 sequences. For each family in this set, we trained an SMC model with the SPST algorithm. We used four-fifths of the sequences in each family for training, and then we applied the resulting model to classify all the sequences in the Swiss-Prot 33 database. To establish the family membership threshold we used the equivalence number criterion (Pearson, 1995). This method sets the threshold at the point where the number of false positives (the number of non-member proteins with score above the threshold) equals the number of false negatives (the number of member proteins with score below the threshold), i.e. it is the point of balance between selectivity and sensitivity. A member protein that scores above the threshold (true positive) is considered successfully detected. The quality of the model is measured by the percentage of true positives detected with respect to the total number of proteins in the family.

Table 1 shows the classification rates obtained with the SPST and F-SPST algorithms, together with the published results obtained with the PST algorithm. We also show the number of false positives for each algorithm. Because of the way of establishing the family membership threshold, the percentage of false positives is equal to 100% minus the percentage of true positives (with respect to the total number of sequences in the family). For example, in the case of family 7tm_1, the percentage of true positives detected by the F-SPST algorithm is 97.7%, so the percentage of false positives is 2.3%. This gives 12 sequences erroneously classified as members of the 7tm_1 family. Figure 3 summarizes the classification rates of Table 1 in two scatter-plots.

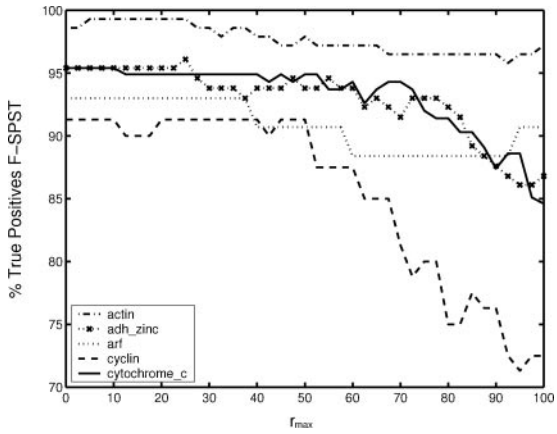


**Fig. 4.** Performance evaluation of the F-SPST algorithm as a function of the user-specific parameter $r_{\max}$. The $x$-axis shows the values of this parameter used to estimate the model. The evaluation was made for five randomly chosen families.

It was shown in Bejerano and Yona (2001) that the performance of the PST algorithm can be improved increasing the number of nodes in the tree. In the case of SPST and F-SPST, this fact depends on the values of the parameters $L$ and $N_{\min}$. In this work we study the performance of the F-SPST algorithm as a function of the other user-specific parameters $r_{\max}$, $\gamma_{\min}$ and $M$. Five families of Table 1 were randomly chosen and the SMC models were estimated varying the value of the corresponding parameter. The performance of the F-SPST algorithm was not significantly affected in the case of $\gamma_{\min}$ and $M$. The reader can see these results in the Supplementary Material for this paper. In Figure 4 we show the results for $r_{\max}$.

The implementation of the SPST algorithm described in this paper was coded in ANSI C and compiled using `gcc`. The run time spent training a model on a AMD (Athlon) 851 Mhz PC, for a protein family in the Pfam 1.0 database, varies between 2 s and 49 min. The computation of the scores for all sequences in the Swiss-Prot 33 database takes on average 1 min 40 s for the SPST algorithm and 1 min 5 s for the F-SPST algorithm.

## 5 DISCUSSION

The results presented in this paper strongly suggest that the SPST and F-SPST algorithms can improve the classification rates obtained with the PST algorithm. This probably owes to the fact that the sparse model mimics well the sparse nature of relevant domains in the amino acid chains. Another very interesting feature of SPST appears when comparing nodes in the estimated trees with the classes obtained by grouping the amino acids according to their physical and chemical properties. For instance, the estimated tree for the ATPase family associated with various cellular activities (AAA) family has as a sparse node the set of amino acids $\{I, V, L\}$. This set corresponds exactly with the group of aliphatic amino acids (Fig. 5).

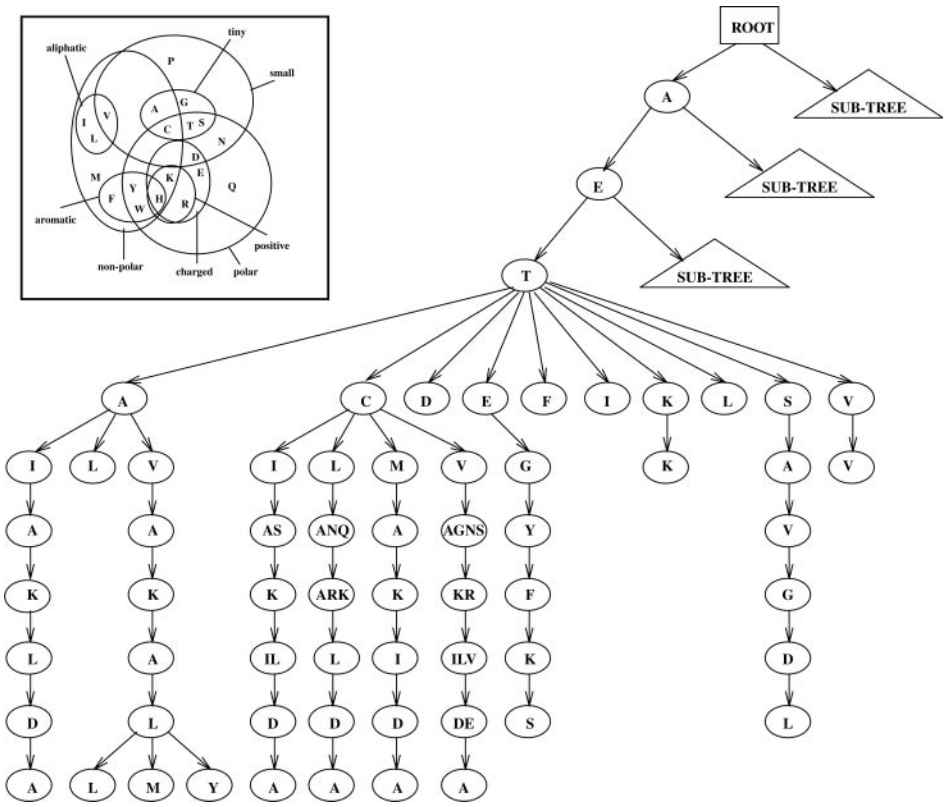


**Fig. 5.** An SMC tree estimated with the SPST algorithm. This tree corresponds to sequences in the AAA family (ATPase family associated with various cellular activities). In each node we can see the subsets of amino acids corresponding to different positions of the sparse contexts (the curly brackets of the subsets were dropped). Some nodes of the tree are in correspondence with the physico-chemical groups of the amino acids (shown in the square), as for example the set $\{I, L, V\}$ that corresponds to the set of aliphatic amino acids.

In Figure 4 we see that the performance of F-SPST as a function of the parameter $r_{\max}$ decreases when the value of $r_{\max}$ increases. This is due to the fact that when $r_{\max}$ increases, the number of nodes in the tree decreases (more nodes are merged), and an under-fitted model is obtained. But it is important to note that for all values of $r_{\max}$ inferior to 50, the performance of F-SPST for the five families is maintained over 90%.

Actually we do not know which conditions must be satisfied by the stochastic process in order for the SPST algorithm to be consistent. We know that in the simple tree configuration of Figure 2 the SPST algorithm is consistent, but in the general case it is not. For space limitations we can not present in an appropriate way these facts, but we will discuss the important issue of consistency of the SPST algorithm in a forthcoming paper.

## ACKNOWLEDGEMENTS

The author thanks her advisors, A. Galves and H. Armelin, for illuminating discussions during the preparation of this paper, P.-Y. Bourguignon for making his manuscript available for her and C. Coletti and R. Vêncio for helpful remarks. She also thanks the reviewers for their useful comments. This work is supported by CAPES and is part of PRONEX/FAPESP's project Stochastic behavior, critical phenomena and rhythmic pattern identification in natural languages (grant number 03/09930-9) and CNPq's project Stochastic modeling of speech (grant number 475177/2004-5).

*Conflict of Interest*: none declared.

## REFERENCES

Apostolico,A. and Bejerano,G. (2000) Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. *J. Comput Biol.*, **7**, 381–393.

Bateman,A. *et al.* (2004) The Pfam protein families database. *Nucleic Acids Res.*, **32**, 138–141.

Bejerano,G. (2004) Algorithms for variable length Markov chain modeling. *Bioinformatics*, **20**, 788–789.

Bejerano,G. and Yona,G. (2001) Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, **17**, 23–43.

Boeckmann,B. *et al.* (2004) The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res.*, **31**, 365–370.

Bourguignon,P.-Y. and Nicolas,P. (2004) Modèles de Markov parcimonieux: sélection de modèle et estimation. In *proceedings of JOBIM 2004*, Montreal.

Bülhmann,P. and Wyner,A.J. (1999) Variable length Markov chains. *Annal. Stat.*, **27**, 480–513.

Eskin,E. *et al.* (2000) Protein family classification using sparse Markov transducers. *Proc. Intl Conf. Intell. Syst. Mol. Biol.*, **8**, 134–145.

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

Rissanen,J. (1983) A universal data compression system. *IEEE Trans. Inform. Theory*, **29**, 656–664.

Ron,D. *et al.* (1996) The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length. *Mach. Learn.*, **25**, 117–149.

Rust,A.G. *et al.* (2002) Genome annotation techniques: new approaches and challenges. *Drug Discov. Today*, **7**, 70–76.