Protein classification and domain detection with Probabilistic Suffix Trees

Grégoire Lejay

May 27, 2010

Contents

1 Proteins and their classification

Products of the genetical code, proteins are organic compounds made of amino acids arranged in a linear chain and folded into a globular form. They are involved in almost every activities of the cell from atoms to the development of an organism.Their action is provoked by the binding of those macromolecules to other molecules. Its action to other molecules is called the function of a protein. There are four fundamental functions associated with proteins:
The first one is the binding, the ability to recognize other molecules. The myoglobin which fixes the oxygen molecule to iron atoms.An other one is the catalyze, the ability to trigger and accelerate chemical reaction.The DNA polymerase that catalyze the DNA replication is one example. They can be switches, this means that their shape can change under particular conditions and have a different function under different conditions. Finally, they can have a structural function inside the cell. Protein subunits can associate them-self with other proteins. This allows the construction of more complex systems and the definition of structures. For example, the F actin associates with other related proteins to build the cytoskeleton. An important fact is the function of the proteins are close related to its shape and the chemicals properties of the amino acids which the proteins are composed of.
Despite the fact that each protein possess its own tridimensional structure,many different proteins adopt the same shape, or architecture, a combination of secondary structure elements or domains(compact region of a the structure of the protein which is composed of the continuous segment of amino acids and which is able to fold in sufficiently stable manner to be in an aqueous solution). Thus, if two proteins fold in the same manner, we can suppose that they belong to a same set called a superfamily, and that they could be homologous. This set can also be divided into families if some functions of the members of a superfamily are sufficiently close and if their similarity is greater than 50 percent.

2 How to represent a family

A protein family is a group of evolutionary related protein sequences that exhibit many identical short statistical patterns of amino acids. Considering the evolution process, we can suppose that every protein family members descend from a single ancestral sequence. Thus, use statistical model can be a good idea to represent a group of related objects. Here, we will first present a first method to represent a protein family and classify proteins :

the Hidden Markov Models. Then, we will present the Probabilistic Suffix Tree the object we are interested in.

2.1 Hidden Markov Models

blahblah blah

2.1.1 How does it work ?

A hidden Markov Model describes a series of observations generated by hidden Markov Process where each state emits an observed symbol with a probability. In the case of proteins, the state of the Markov process are positions and the observations are amino acids.
To represent a family, we need to build a HMM that identifies a set of positions that describe the conserved subsequences of the family we consider. Let Fam be a family of protein sequence, the common structure of all the sequences within Fam can be view as a sequence of positions where amino acids occur. For each of these positions a probability distribution can defined over each amino acids that represents the probability of an amino acids to appear at this particular position and the probability that there is a gap at this position.
Thus, a HMM can be build with three different kind of states : the first one are the match states ($M = \{m_0, m_1, \cdots, m_n\}$), the second one are the insert states ($I = \{i_0, \cdots, \imath_{n-1}\}$) and the third one are the delete states ($D = \{d_1, \cdots, d_{n-1}\}$). There is a unique starting state m_0, a unique ending state m_n, the transitions always go from left to right such that a match or a delete are never visited more than once and for each state there are three possible transition.Delete states emit a symbol of deletion with a probability of one. Match and insertion states have different probabilities distributions, this is important to differentiate random to family related proteins.
An example of HMM:

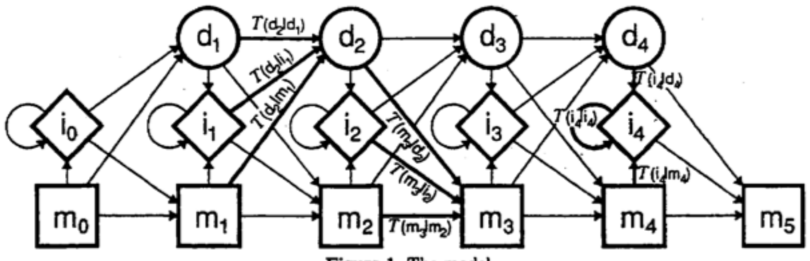


Figure 1. The model.

!!!Voir plus tard s'il faut plus de details !!!
Three values can be compute :

The first one is the evaluation problem $P(O|Model)$,compute the probability that $Model$ generates the sequence of observations O. This can be computed using the following formula :
$P(O|Model) = \sum\limits_{Q \in S^T} P(Q|Model)P(O|Q, Model)$, where Q is a sequence of states, this mean compute $P(Q|Model)P(O|Q, Model)$ for every possible transition sequence. So, this is a sum with an exponential number of members (N^T where N is the number of states). An idea to compute this is to use the forward algorithm that reduce the sum to $O(N^2T)$.
The second one is the decoding problem, given a sequence of observations O, find the most probable sequence of corresponding states. This mean compute : $Q^* = argmax_Q P(Q|Model)P(O|Q, Model)$.
Finally, the learning problem, find the model that represent a given series of observations. In order to model a family of related sequence, the goal is to find the parameters (i.e.. transition and emission probabilities) that fit with the statistical properties of the considered family, indeed we can easily see that each set of parameters produce different set of related sequence, thus different families. This mean solve the learning problem and this can be done thanks to the Baum-Welsh algorithm.

2.1.2 Example of use: the Pfam database

Pfam is a database available on the internet.

2.2 The Probabilistic Suffix Trees

Many of the known approaches used to model a motif, a domain or a protein family starts by a multiple alignment which is not easy to do,and does not guarantee to find an optimal alignment. The idea is to identify short segments among protein sequences that reflects statistical properties of the corresponding family. Thus, we can compute a probability distribution on the next symbol to appear given the preceding subsequence by observing no more than a certain number of preceding symbols. So , a first idea could be to model those families with an order L Markov chain. But, one of the main drawback of the Markov chains is their trend to grow in an exponential way in the number of states. However, many natural phenomenon an event depends on past events contexts with variable length. So, it can be relevant to use variable length Markov chain. This one can be modeled by a structure called probabilistic suffix automaton which is equivalent to an other one called Probabilistic suffix tree.

2.2.1 The idea

So,what is a Probabilistic suffix Tree? A PST T over an alphabet Σ is a rooted tree where the degree of each node vary between zeros and $|\Sigma|$. Each vertex in the tree is associated with a symbol σ such that $\sigma \in \Sigma$ and that each path from a node v to the root define an unique sequence s. The node v is labelled by s.So, each parent node is the longest suffix of its children.
Moreover, each node in the tree is associated with a probability distribution which describes the probability $P^T(\sigma|s)$;the probability that the symbol σ appears given the longest subsequence of symbols that have been observed right before it (given s) . Let $\bar{\gamma}_s : \Sigma \rightarrow [0,1]$ be a function such that $\bar{\gamma}_s(\sigma) = P^T(\sigma|s)$, for each node in the PST T, we must have $\sum\limits_{\sigma \in \Sigma} \bar{\gamma}_s(\sigma) = 1$.
This distribution will be used by the algorithm to predict significant patterns among a sequence.

2.2.2 How to build a PST

Here we present and explain the algorithm used to build a Probabilistic Suffix Tree. To do that we first need some definitions and notation which will be used to describe the algorithm.

Definitions and notations First, let $S_{sample} = \{r^i | i \in [\![1,m]\!]\}$ be a set (the training set) of strings over the alphabet Σ. In our context S_{sample} is the set of all the amino-acid sequences we want to use to produce a PST.
Each node of a PST is associated with a probability distribution which describes the probability that the symbol σ appears right after the sequence s,so few definitions are required to represent it:
We need an indicator variable $\chi_s^{i,j}$ which has value one if the string s such that $|s| = l$is a subsequence of the string r^i starting a the position j.

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

Thus, we can can compute the number of occurrences of the string s in S_{Sample} with:

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

. In order to have the empirical probability of observing the string s among S_{Sample} we need to compute the number of subsequences of the same length than s (l), it means compute :

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

It is now easy to define the empirical probability of observing s by :

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

To compute the probability of a symbol σ to occur right after a given substring, we need to use a conditional probability.
$\tilde{P}(\sigma|s) = \dfrac{\chi_{s\sigma}}{\chi_{s\star}}$, where $\chi_{s\star}$ is the sum over every symbols α in the alphabet of the number of occurrences of the string $s\alpha$ in S_{Sample}.

The algorithm Here we present the algorithm which is the pillar of this report. This will allow the construction of a PST given the a set of sequences. The PST model identifies significant appearance of short segments among many of the input protein sequence regardless of the position of these segments in the proteins sequences of the training set. These segments reflect some statistical properties of the protein family (the set) and are important to be well represented by the model; this can be controlled by the parameters leading the construction of the PST that represents the family.

The first set is the initialization step:

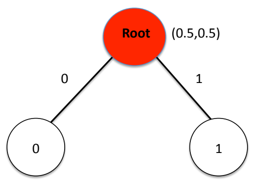
Let $\bar{T}$ be the PST. At the beginning, it will only consist of a single root node labelled by the empty word. And let $\bar{S} \leftarrow \{\sigma | \sigma \in \Sigma \text{ and } \tilde{P}(\sigma) \geq P_{min}\}$, this is the set of all the symbols that have a sufficient chance to appear in a sequence,later we will see that $\bar{S}$ is the candidates set, the set of the nodes that have a chance to be add to the tree.

The second step is the main loop of the algorithm:

While the set $\bar{S}$ is not empty, pick $s \in \bar{S}$ and:
Remove s from $\bar{S}$.
If there exists a symbol $\sigma \in \Sigma$ such that:
$\tilde{P}(\sigma|s) \geq (1+\alpha)\gamma_{min}$
and
$\frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suf(s))} \geq r$ or $\leq \frac{1}{r}$
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.
This means that if s is in the $\bar{S}$ set, the probability that s occurs in the sequence is non negligible. So, if there exists symbols such that the probability

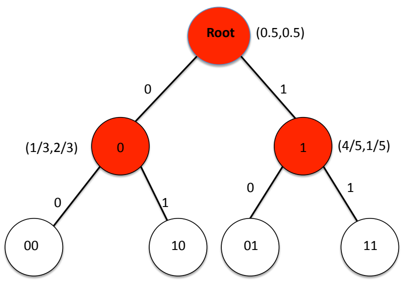
of being observed right after s is also sufficient and such that the probability of being observed right after s is different of the probability of observing this symbol right after $suf(s) = s_2 \cdots s_l$ (this corresponds to the direct father of that node) then we can a new node to the tree labelled by s. Otherwise, the two probabilities are similar and that node contains no new informations, so it do not need to be add to the tree.
To end the loop, the algorithm must update the set of candidates.
Finally,the algorithm must assures that no symbol in the tree is predicted to have a zero probability.In the same time, it will compute the probability distribution associated to each node.
For each string s labeling a node in the tree, we compute:
$\bar{\gamma}_s(\sigma) = (1 - |\Sigma|\gamma_{min})\tilde{P}(\sigma|s) + \gamma_{min}$

Example If we follow the algorithm proposed by Gill Bejerano and Golan Yona with the following parameters $P_{min} = 0.1, \alpha = 0, \gamma_{min} = 0.1$ (the others have no real importance in this short example), we can easily build a PST from a short set of sequences.
Let $A = \{0111, 1001, 0101\}$ be the set of sequences which will be represented by a PST. The algorithm starts with at the root (ϵ) and labels it with an uniform distribution. $a_i = \frac{1}{|\Sigma|}$.
Then it computes the empirical probability of observing 1 or 0 right after the empty word ($\tilde{P}(0|\epsilon) = \frac{5}{12}$) and ($\tilde{P}(1|\epsilon) = \frac{7}{12}$) those probabilities are not negligible, we can add the 0 and 1 nodes as candidates (white nodes).
At this step the tree looks like that:

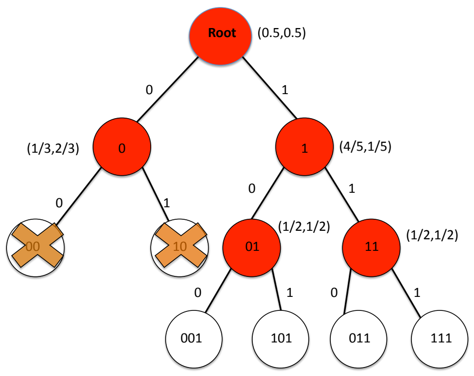


Now we compute for each node the corresponding distribution which is for each letter in the alphabet the probability of observing this letter right after the word that labels the node.

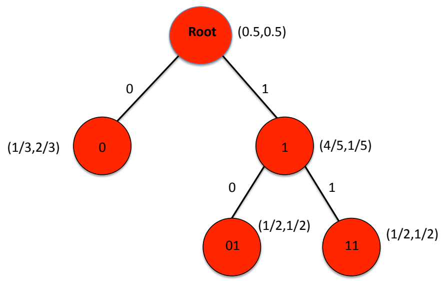
Thus, we compute:
$\tilde{P}(0|0) = \frac{1}{3}$
$\tilde{P}(1|0) = \frac{2}{3}$
$\tilde{P}(0|1) = \frac{4}{5}$
and $\tilde{P}(1|1) = \frac{1}{5}$.
Those two new distribution: $(\frac{1}{3}, \frac{2}{3})$ for the node 0 and $(\frac{4}{5}, \frac{1}{5})$ for the node 1 are different of their father node, so they are validated. An other step is to compute for each possible child of the considered nodes the empirical probabilities to observe them in the set of sequences.
We get: $\tilde{P}(00) = \frac{1}{9}, \tilde{P}(10) = \frac{2}{9}, \tilde{P}(01) = \frac{4}{9}, \tilde{P}(11) = \frac{1}{9}$, they are all greater than P_{min} so they are added to the tree.
We obtain the following tree:



Now, it's a new iteration of the algorithm and we compute the same things for all the nodes recently added to the tree.
Here, the nodes 00 and 10 are not validated because $\tilde{P}(0|00) = 0$ and $\tilde{P}(1|10) = 0$.However,the nodes 01 and 11 are validated because their distributions both equals to $(\frac{1}{2}, \frac{1}{2})$ that is different of $(\frac{4}{5}, \frac{1}{5})$ (the distribution of 1). Their child have all a sufficient probability of being observed in A, thus they added to the tree.



Finally, no of the last nodes added to the PST are validated because the strings : 0100, 1100, 0010,0001,1101,1011 and 1111 never appears in A.
At the end, the PST that represents A looks like that:



2.2.3 Prediction using a suffix tree

In order to know if an amino acid sequence s belongs to a family, we need to compute the probability P^T that s is generated by the PST corresponding to the family. And then, compare P^T with the probability that s is generated by an uniform distribution.
To compute the P^T, we need to travel s letter by letter. At each step, we look for the longest suffix θ that appears in the tree (θ is the label of a node of the tree) and that finish just before the last letter travelled.This represents the longest significant context influencing the probability of the next symbol (!!en parler dans la partie sur l'algorithme!!).In fact, out of this context the

probability of the next symbol is quite independent of all other symbol.

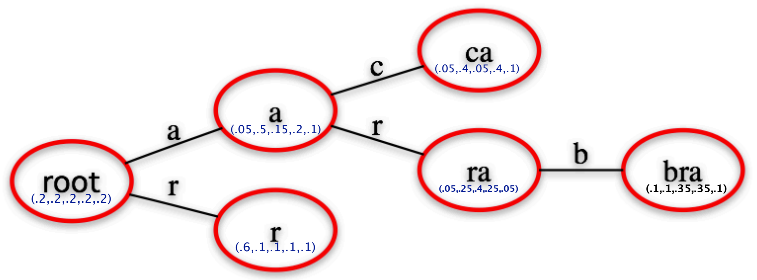
Thus, $P^T(s) = P^T(s_1).\prod_{i=2}^{|s|} P^T(s_i|(s_1 \ldots s_{i-1}))$

In the right part of each member of $P^T(s)$, we find θ which corresponds to a node of the PST.

So, $P^T(s) = \bar{\gamma}_{root}(s_1).\prod_{i=2}^{|s|} \bar{\gamma}_{\theta}(s_i)$

where $\bar{\gamma}_s$ is the distribution associated with the node labelled by s.

For exemple: The prediction of the string $s =$abradacaba using the PST below:



$P^T(abradacaba) =$
$P^T(a).P^T(b|\underline{a}).P^T(r|ab).P^T(a|ab\underline{r}).P^T(d|a\underline{bra}).P^T(a|abrad).$
$P^T(c|abrad\underline{a}).P^T(a|abradac).P^T(b|abrada\underline{ca}).P^T(a|abradacab)$
$= \bar{\gamma}_{root}(a).\bar{\gamma}_{a}(b).\bar{\gamma}_{root}(r).\bar{\gamma}_{r}(a).\bar{\gamma}_{bra}(d).\bar{\gamma}_{root}(a).\bar{\gamma}_{a}(c).\bar{\gamma}_{root}(a).\bar{\gamma}_{ca}(b).\bar{\gamma}_{root}(a)$
$= 0, 2.0, 5.0, 2.0, 6.0, 1.0, 2.0, 15.0, 2.0, 4.0, 2$
$= 5, 76.10^{-6}$
The underlined sequences are the θ's found.
An uniform distribution would give a probability :

$P_{unfiform} = 0, 2^{10} = 1, 024.10^{-7}$.

So, $\dfrac{P^T(s)}{P_{uniform}} = \dfrac{5, 76.10^{-6}}{1, 024.10^{-7}} = 56, 25$

This make s be 56 times more plausible under the PST than under the uniform model.

2.2.4 Complexity:How to build and predict in linear time

If we let n be the length of the training set,the maximal depth of the PST L and the length of the query sequence m, a PST can be learned in $O(Ln^2)$

in time and $O(Ln)$ in space as there are $O(Ln)$ different subsequences of length between 1 and L. The prediction can be done in $O(Lm)$ because every symbol predicted requires traversing the tree from the root the a node of depth L. But, the part of the algorithm that cost the most is the main loop in the tree construction. !!!FINIR!!!

3 Family classification : Methods and results

3.1 Method

To establish a classification, each family set of the Pfam database was cut in five and $\frac{4}{5}$ of the family set was use to train a PST and then use to classify the protein sequences in the SwissProt database. The goal is to decide if a sequence in a protein database belongs to the family we consider or not. Such discriminative process is called a binary classification. So, for each sequence in the database, we will compute the probability that this sequence is generated by the PST that represent the family divided by the length of the sequence. Because short sequences should have a high probability and long sequence a low probability, this normalization is important to avoid bias due to length difference.

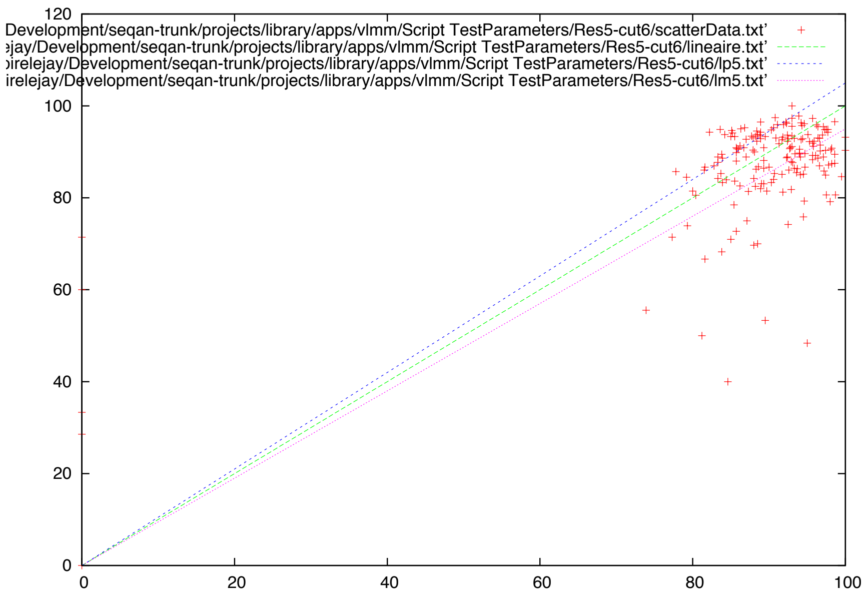
3.1.1 Equivalence number criterion

This is a method to estimate the quality of a PST. First, one will compute a score for every sequence in the database and a score for every sequence in the training set. The equivalence number criterion procedure will find the point (iso-point) where the number of False Positives equals the number of False Negatives. This also mean find the point where the number of protein that doesn't belong to the family whose probability is a above a threshold and the number of proteins that belong to the family but with a probability less than the threshold. In that case, a decision rule is quite easy to determine. In fact, if a sequence belongs to the family and its probability is above the threshold then we can consider that the PST detected the sequence. The iso-point is the point where the number of False Positives (FP) equals the number of False Negatives (FN); then it is also the point where the specificity (the ability to reject sequences that doesn't belong to the family : $\frac{TN}{TN+FP}$) equals the sensitivity (the ability to detect sequences that belong to the family :$\frac{TP}{TP+FN}$)

3.1.2 Two different kinds of scores

Here we present two different way to score a sequence given a PST an we will compare the results to those obtained by Bejerano and Yona. In fact, the implementation of the PST we use is not the original one programmed by Bejerano and Yona, and contains some important differences we will approach in the determination of the parameters section.

The likelihood: This first method is the one explained in the Prediction using a probabilistic suffix tree. The goal is to compute for each sequence in the SwissProt database the probability that this sequence is generated by the PST that reprensents the family. And then compute the sensitivity of our PST thanks to the equivalence number criterion.
After computing and finding a good set of parameters for each 175 families of the first version of Pfam,we computed the sensitity of the trained PST and we made a scatter plot to compare the Bejerano's results and the ones got with PISA toolbox(on x axe Bejerano's results, on y axe PISA).



Scatter plot of Bejerano's results against PISA,and a marging of 10percent.

Now, one can observe that the results are well balanced, there are 38 pints in the margin, 65 point Bejerano's results advantage and 72 in Pisa's advantage. A zero sentivity on bejerano's results are due to missing data. Low results for PISA's useally correspond to family with a short number of sequences whose have a lot of residues (like lamininEGF),thus it will be hard to tune correctly the parameters.

The local score:

3.1.3 Determination of the parameters

cutoff-¿depth-¿alpha-¿gammamin-¿Pmin cutoff-¿alpha-¿Pmin-¿gammamin-¿Depth

3.2 Results

safdgh

3.2.1 Comparaision with HMMer

fdsghjk

4 Local score for domain detection

4.1 The idea

asdsdasd

4.2 The algorithm

asdasdasd

4.3 Results

dfsdfsdf

5 Bibliography

Gill Bejerano and Golan Yona,Variations on probabilistic suffix trees:statistical modeling and prediction of proteins families.
Zhaohui Sun,Jitender S.Deogun,Local Prediction Approach for Protein Classification Using Probabilistic Suffix Trees.
Warren Ewens,Gregory Grant,Statisical Methods in Bioinformatics: An Introduction.