Unsupervised Segmentation and Classification of Mixtures of Markovian Sources

Yevgeny Seldin Gill Bejerano [*] Naftali Tishby

School of Computer Science and Engineering
Hebrew University, Jerusalem 91904, Israel
E-mail: {seldin, jill, tishby}@cs.huji.ac.il

Abstract

We describe a novel algorithm for unsupervised segmentation of sequences into alternating Variable Memory Markov sources, first presented in [SBT01]. The algorithm is based on competitive learning between Markov models, when implemented as *Prediction Suffix Trees* [RST96] using the MDL principle. By applying a model clustering procedure, based on rate distortion theory combined with deterministic annealing, we obtain a hierarchical segmentation of sequences between alternating Markov sources. The method is applied successfully to unsupervised segmentation of multilingual texts into languages where it is able to infer correctly both the number of languages and the language switching points. When applied to protein sequence families (results of the [BSMT01] work), we demonstrate the method's ability to identify biologically meaningful sub-sequences within the proteins, which correspond to signatures of important functional sub-units called domains. Our approach to proteins classification (through the obtained signatures) is shown to have both conceptual and practical advantages over the currently used methods.

1 Introduction

Unsupervised segmentation of sequences has become a fundamental problem with many important applications such as analysis of texts, handwriting and speech, neural spike trains and bio-molecular sequences. The most common statistical approach to this problem, using hidden Markov models (HMM), was originally developed for the analysis of speech signals, but became the method of choice for statistical segmentation of most natural sequences. HMMs are predefined parametric models - their architecture and topology are predetermined and the memory is limited to first order in most common applications. The success of HMMs thus crucially depends on the correct choice of the state model. It is rather difficult to generalize these models to hierarchical structures with unknown a-priory state-topology (see [FST98] for an attempt).

An interesting alternative to the HMM was proposed in [RST96] in the form of a sub class of *probabilistic finite automata*, the variable memory Markov (VMM) sources. These models have several important advantages over the HMMs: (i) they capture longer correlations and higher order statistics of the sequence; (ii) they

[*]Supported by a grant from the Ministry of Science, Israel

can be learned in a provably optimal PAC like sense using a construction called *prediction suffix tree (PST)* [RST96]; (iii) they can be learned efficiently by linear time algorithm [AB00]; and (iv) their topology and complexity are determined by the data.

In the first half of the paper we review a powerful extension of the VMM model and the PST algorithm to a stochastic mixture of such models, suggested in [SBT01]. The algorithm learns the models in a hierarchical competitive way using a deterministic annealing (DA) [Ros98] approach. The problem is generally computationally hard, similarly to data clustering. Only very simple sequences can be correctly segmented efficiently in general [FR95]. Our model can in fact be viewed as an HMM with a VMM attached to each state, but the learning algorithm allows a completely adaptive structure and topology both for each state and for the whole model. The approach we take is information theoretic in nature. The goal is to enable short description of the data by a (soft) mixture of VMM models, each one controlled by an MDL principle (see [BRY98] for a review). This we do by modifying the original PST algorithm using the MDL formulation, while preserving its good learnability properties. The mixture model is then learned via a generalized *rate distortion theory* (see [CT91, Ch. 13]) approach. Here we take the *log-likelihood* of the data by each model as an effective *distortion measure* between the sequence and its representative model and apply the Blahut-Arimoto (BA) algorithm (see [CT91]) to optimally partition the sequence(s) between the VMM model centroids. Just like in many clustering algorithms we then update the models based on this optimal partition of the sequence(s). In this way a natural resolution parameter is introduced through the constraint on the expected tolerated distortion. This "temperature" like Lagrange multiplier is further used in the deterministic annealing loop to control the resolution of the model. The hierarchical structure is obtained by allowing the models to split (the *refinement* step) after convergence of the iterations between the BA algorithm and the VMM centroids update. The algorithm exhibits several interesting features that will be only mentioned here due to space limitations. Their discussion is left to [Sel01].

In the second part of the paper we cite the results of an interesting application of the algorithm to the problem of protein sequences classification, mainly taken from the [BSMT01] work. The function of a protein is determined by its sequence. Numerous proteins exhibit a modular architecture, consisting of several sequence domains that often carry specific biological functions (reviewed in [BK96]). For proteins whose structure has been solved, it can be shown that in many cases the characterized sequence domains are associated with autonomous structural domains. In proteins of various organisms we find domains that are responsible for similar biochemical functionality. The sequences of those domains are usually resembling, but not identical. Characterization of a protein family by its distinct sequence domains either directly or through the use of domain 'motifs', or 'signatures' (short subsegments of the domain that are typical for most members of that family) is crucial for functional annotation and correct classification of newly discovered proteins.

Many methods have been proposed for classification of proteins based on their sequence characteristics. Most of them are based on a seed multiple sequence alignment (MSA) of proteins that are known to be related (see [DEKM98]). They strongly rely on the initial selection of the related protein segments for the MSA, usually hand crafted by experts, and on the quality of the MSA itself. Besides being in general computationally intractable, when remote sequences are included in a group of related proteins, establishment of a good MSA ceases to be an easy task and delineation of the domain boundaries proves even harder. This becomes nearly impossible for heterogeneous groups where the shared motifs are not necessarily abundant or do not come in the same order.

In the earlier work of [BY01] PSTs were shown to be a powerful tool for supervised classification of proteins. This work extends our abilities by allowing to perform this task in unsupervised manner. The advantage of our algorithm is that it does not attempt any alignment, but rather clusters together regions with similar *statistics*. The regions need not come in the same order, nor they need to be identical. In addition there is no need in prior selection of groups of related proteins, the algorithm finds them even in a bunch of unrelated stuff, as we will show here. This is even more attracting since the algorithm may find some new structure or correlations in the data we possibly have not thought about. Thus our approach opens a new promising way to protein sequence analysis, classification and functional annotation.

The paper is built in the following way. In Sec. 2 and Sec. 3 we describe the algorithm from [SBT01]. In Sec. 4 we apply the algorithm to the problem of segmentation of a mixture of interchanging texts in 5 different European languages. Here the model identified both the correct number of languages and the segmentation of the text between the languages with resolution of a few letters. In Sec. 5 we show the results of [BSMT01] work, where we apply the algorithm to the problem of protein sequences classification. Here the algorithm was able to refine the HMM superfamily classification and identify domains that appeared in a very small percent of the input proteins (in one case only 12 proteins out of 396 input sequences). Sec. 6 holds a discussion of the algorithm and the achieved results.

2 Single Source Modeling

In this section we will define VMM processes, review an efficient data structure for their representation from [RST96] and a non-parametric learning algorithm from [SBT01] that is later used as a core for the segmentation process.

2.1 Variable Memory Markov Processes

Given a string $\bar{x}$, over a finite alphabet Σ, that was sequentially generated by some statistical source G, the probability that G has generated that particular sequence can always be written as: $P_G(\bar{x}) = P_G(x_1..x_n) = \prod_{i=1}^{n} P_G(x_i|x_1..x_{i-1})$. In this section we assume G to be stationary and ergodic [CT91]. We define a *context* of x_i to be any substring $x_{i-m}..x_{i-1}$ for $m \geq 0$. If $m = 0$ we say that the context of x_i is the empty string, denoted by λ. Further we define C to be any finite subset of strings in Σ^* that includes λ. We say that $x_{i-m}..x_{i-1}$, or λ, is the *C-context* of x_i if it is the longest suffix of $x_1..x_{i-1}$ in C. Process G respects context set C if $P_G(x_i|x_1..x_{i-1}) = P_G(x_i|C\text{-}context(x_i))$ for all i. The length of C-context(x_i) is the memory of process G at place i, and it may vary with i.

2.2 Prediction Suffix Trees (PSTs)

A context set C may be efficiently represented using a tree. By associating a distribution vector over Σ with each node of the tree we get a PST[1] (see Fig. 1). Formally, a PST T is a $|\Sigma|$-ary tree that satisfies:

1. For each node each outgoing edge is labeled by a single symbol $\sigma \in \Sigma$, while

[1]A Prediction Suffix Tree is related to, but differs from a classical suffix tree (see [AB00]).

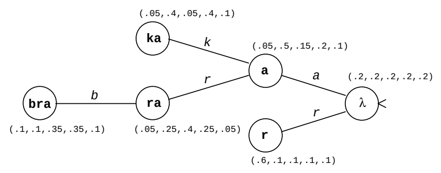


Figure 1: **An example of a PST** over the alphabet $\Sigma = \{a, b, k, l, r\}$. The vector near each node is the probability distribution for the next symbol. E.g., the probability to observe k after the substring $bara$, whose largest suffix in the tree is ra, is $P(k|bara) = P_{ra}(k) = 0.4$.

```
Learn_PST(String $\bar{x}$, Weights $\bar{w}$)

  1. $T$ = Build_PST($\bar{x}$, $\bar{w}$)
  2. Prune($T$, $\lambda$)

The two steps:
I. Build_PST (String $\bar{x}$, Weights $\bar{w}$)

  1. Start with $T$ having a single node $\lambda$.
  2. Recursively for each $s \in T$ and $\sigma \in \Sigma$
       If $Size(\sigma s) < H(P_s) \cdot w(\sigma s)$ Then
           Add node $\sigma s$ to $T$.

II. Prune (Tree $T$, node $s$)

  Recursively for each $\sigma \in \Sigma$ such that $\sigma s \in T$:
    1. Prune($\sigma s$)
    2. If $TotalSize(T_{\sigma s}) > H(P_s) \cdot w(\sigma s)$ Then
         Delete subtree $T_{\sigma s}$
```

Figure 2: **PST learning algorithm.**

there is at most one edge labeled by each symbol.

2. Each node of the tree is labeled by a *unique* string s (a context) that corresponds to a 'walk' starting from that node and ending in the root of the tree. We identify nodes with their labels and label the root node by the empty string λ.

3. A probability distribution vector $P_s(\sigma)$ is associated with each node s. $P_s(\sigma)$ is the probability that a symbol σ will come after the context s.

We define $suf_T(x_1..x_i)$ as the longest sequence $x_{i-m}..x_i$ that makes a path in T when we start from the root and traverse the edge labeled by x_i, from there we traverse the edge labeled by x_{i-1} etc., until there is no appropriate edge to continue with or we have traversed the whole string[2]. If there is no edge labeled by x_i at the root we say that $suf_T(x_1..x_i) = \lambda$. The collection of all node labels in T make up our set of memorized contexts.

2.3 Predicting and Generating using PSTs

Here we define the probability measure that a PST T induces on the space of all strings $\bar{x} \in \Sigma^n$, for any given n. Given a string $\bar{x} \in \Sigma^n$ and a PST T the probability that $\bar{x}$ was generated by T is:

$$P_T(\bar{x}) = \prod_{i=1}^{n} P_T(x_i|x_1..x_{i-1}) = \prod_{i=1}^{n} P_{suf_T(x_1..x_{i-1})}(x_i)$$

When T is used as a generator, it generates a symbol x_i according to the distribution $P_{suf_T(x_1..x_{i-1})}$.

2.4 Learning PSTs

We now turn to describe the MDL driven algorithm for PST learning, presented in [SBT01]. The algorithm is non-parametric and exhibits self-regularization. It is

[2]Note that we do not necessarily stop at a leaf.

generalized to handle weighted data, which will appear later on.

The inputs to the algorithm are a string $\bar{x} = x_1..x_n$ and a vector of weights $\bar{w} = w_1..w_n$, where each w_i is a weight associated with x_i $(0 \le w_i \le 1)$[3]. We will denote $w(x_i) \equiv w_i$. You may think of $w(x_i)$ as a measure of confidence we give to the observation x_i. For now you may assume all $w_i = 1$.

For a string s we say that $sx_i \in \bar{x}$ if it is a substring of $\bar{x}$ ending at place i. We define:

$$w_s(\sigma) \equiv \sum_{x_i=\sigma \ and \ sx_i \in \bar{x}} w(x_i)$$

and $w(s) \equiv \sum_{\sigma \in \Sigma} w_s(\sigma)$. Clearly $\frac{w_s(\sigma)}{w(s)}$ is an empirical estimate for $P_s(\sigma)$.

The idea behind MDL is to minimize the total length (in bits) of model description together with the code length of the data when it is encoded using the model. When coding a single node s we should enumerate its sons and encode the distribution vector P_s. The first takes $|\Sigma|$ bits - bit σ denotes the presence of son σ. For the second it is sufficient to code all the counts $w_s(\sigma)$. Since the total amount of data "passing through" node s[4] is $w(s)$ the counts should be coded to within accuracy $\sqrt{w(s)}$. Thus the description size of s is:

$$Size(s) = |\Sigma| + \frac{|\Sigma|}{2} \cdot \log_2(w(s))$$

Denoting by T_s the subtree of T rooted at node s:

$$Size(T_s) = Size(s) + \sum_{\sigma s \in T} Size(T_{\sigma s})$$

($s \in T$ means that s is a node in T). The minimal average code length per symbol, for all symbols coded using node s, is given by the *entropy* of P_s, $H(P_s) \equiv -\sum_{\sigma \in \Sigma} P_s(\sigma) \cdot \log_2(P_s(\sigma))$. The equivalent quantity for a subtree T_s is thus a weighted sum given by:

$$H(T_s) = \sum_{\sigma s \in T} \frac{w(\sigma s)}{w(s)} \cdot H(T_{\sigma s}) + \sum_{\sigma s \notin T} \frac{w(\sigma s)}{w(s)} \cdot H(P_s)$$

Summing this altogether we get:

$$TotalSize(T_s) = Size(T_s) + w(s) \cdot H(T_s)$$

Our goal is to minimize $TotalSize(\lambda)$ which is the total description length of the whole tree together with all coded data (as all data passes through the root node λ). The algorithm works in two steps. In step I we extend all the nodes that are potentially beneficial, i.e. by using them we *may* decrease the total size. Clearly only those nodes whose description size is smaller than the code length of data passing through them when that data is coded using the parent node distribution are of interest. In step II the tree is recursively pruned so that only truly beneficial nodes remain. If a child subtree $T_{\sigma s}$ of some node s gives better compression (respecting its own description length) than that of its parent node, that subtree is left, otherwise it is pruned. The algorithm is given in Fig. 2

[3]Generalization to a set of multiple strings is straightforward and therefore omitted here for ease of notation. See [RST96] for an example of such generalization on the original algorithm.

[4]$suf(x_1..x_{i-1})$ ends with s.

3 Sequence Segmentation Algorithm

In this section we describe the unsupervised sequence segmentation algorithm, presented in [SBT01]. We suppose that a given string $\bar{x}$ was generated by repeatedly switching between several different PST models with some upper bound on the alternation rate. I.e., there are k PSTs and a partition of $\bar{x}$ into continuous segments, such that each segment was generated by a single PST out of k. We assume that the segments are significantly long, so that if we have trained a PST for each source using all of its segments, we could say with high confidence about each segment which model it belongs to. Our goal is to find k' PST models and a segmentation of $\bar{x}$ that will be as close as possible to the original ones.

This problem is similar to the problem of finding the best number and parameters for a Gaussian mixture model of points in R^n. Given a string $\bar{x}$ and a vector of assignment probabilities we can build a PST model and estimate its parameters. Alternatively, a given model induces probabilities on all substrings of $\bar{x}$. Alternating between these two estimations is the essence of the EM algorithm in any mixture model. This alternating estimation algorithm can be embedded in a deterministic annealing (DA) procedure to allow for increasing resolution, or number of mixture components. In our case, however, we do not allow our PST models to switch at every symbol, but rather require continuous segments. The fundamental reason for limiting the model switching frequency is that too short segments do not enable reliable discrimination between different models. DA helps us to avoid many local minima and provides an elegant framework for hierarchical structures (see [Ros98]).

Next we give some definitions and describe the Blahut-Arimoto and our soft clustering algorithm. We then embed it in the DA framework to obtain the hierarchical segmentation.

3.1 Definitions

Let $\mathcal{T} = \{T_j\}_{j=1}^k$ be the set of PSTs of size k we are currently working with. We define $w^j(x_i) \equiv P(T_j|x_i)$ to be the probability that x_i is assigned to model T_j[5].

In order to estimate the quality of a given partition we define a distance (local distortion) between a symbol x_i and a model T_j to be minus log likelihood of T_j on a window of size $2M+1$ around x_i:

$$d(x_i, T_j) = -\sum_{\alpha=i-M}^{i+M} \ln P_{T_j}(x_\alpha | x_1..x_{\alpha-1}) \ .$$

The role of the window is to smooth the segmentation and to enable reliable estimation of the log-likelihood. The *global distortion*, i.e. the average distance between segments and the corresponding models, of an assignment is given by:

$$\langle d \rangle = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} d(x_i, T_j) \cdot P(T_j|x_i) \ .$$

3.2 The Blahut-Arimoto Algorithm

First we want to find the optimal assignment probabilities $P(T_j|x_i)$ for a *fixed* set of PST models $\mathcal{T}$, constrained by the allowed distortion level D. Rate distortion

[5]The vector of weights $\bar{w}^j$ is later used to retrain T_j.

```
Blahut-Arimoto($P(T_*)$, $\beta$)

Repeat until convergence:
  1. $\forall i,j:\ P(T_j|x_i) = \frac{P(T_j)e^{-\beta d(x_i,T_j)}}{\sum_{\alpha=1}^{k} P(T_\alpha)e^{-\beta d(x_i,T_\alpha)}}$
  2. $\forall j:\ P(T_j) = \frac{1}{n}\sum_{i=1}^{n} P(T_j|x_i)$
```

Figure 3: **The BA algorithm**

```
Soft_Clustering($\mathcal{T}$, $P(T_*)$, $\beta$)

Repeat until convergence:
  1. Blahut-Arimoto($P(T_*)$, $\beta$)
  2. $\forall j:\ T_j$ = Learn_PST($\bar{x}$, $\bar{w}^j$)
```

Figure 4: **Soft Clustering**

theory [CT91, Ch. 13] provides us with the optimal assignment via:

$$\min_{\{P(T_j|x_i)\ :\ \langle d\rangle \leq D, \sum_{j=1}^{k} P(T_j|x_i)=1\}} I(\bar{x}, \mathcal{T}) \tag{1}$$

where $I(\bar{x}, \mathcal{T}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} P(T_j|x_i) \cdot \log \frac{P(T_j|x_i)}{P(T_j)}$ is the *mutual information* between $\bar{x}$ and $\mathcal{T}$, and $P(T_j) = \frac{1}{n}\sum_{i=1}^{n} P(T_j|x_i)$ is the proportion of data assigned to model j. In rate distortion theory (1) is called the *rate distortion function*, $R(D)$.

By minimizing the mutual information we in fact enable minimal description length of the sequences using the PST models, subject to a given distortion constraint. Since our distortion, an expected log-likelihood, is also the optimal code length by the model, it is fully consistent with the MDL framework. We thus try to find a mixture of PSTs that enable short description of the complete observation sequence, under some continuity requirements from the resulting segmentation.

We employ the alternating minimization procedure, known as the Blahut-Arimoto (BA) algorithm, which is guaranteed to converge to the optimal assignment (see Fig. 3). There the distortion constraint D is imposed by the corresponding Lagrange multiplier β.

3.3 Soft Clustering

Now we go one step further by allowing to modify the PST models. This is analogous to the centroid re-estimation in clustering. We want to obtain a good (low distortion) segmentation of $\bar{x}$ for a given value of β (the assignment probabilities are given by 1. in the BA algorithm).

We approach this problem using a soft clustering procedure. Given an initial set of k PSTs $\mathcal{T}$, we partition the sequence using the BA algorithm and then retrain all k PSTs, using the assignment probabilities $P(T_j|x_i)$ obtained from the BA as weight vectors $\bar{w}^j$ for the Learn_PST procedure. These two steps are repeated until convergence (see Fig. 4). Here the Lagrange multiplier β plays the role of *resolution* parameter and prevents from falling into local minima.

At every given distortion level D a limited number of PSTs K is sufficient to achieve D. When $k > K$ some of the PSTs collapse into a single model - a phenomenon clearly described in [Ros98] - or remain without data ($P(T_j) = 0$). The latter is caused by the requirement of having continuous segments in the final segmentation. Because of this requirement the competition between the models "pushes out" the models who do not "acquire" enough data in favor of those having more data. In this manner the algorithm "self regulates" its global complexity.

```
The segmentation algorithm:

1. For all $i$, $w^0(x_i) = 1$; $T_0$ = Learn_PST($\bar{x}$, $\bar{w}^0$)          // Initialization
   $\mathcal{T} = \{T_0\}$, $P(T_0) = 1$, $\beta = \beta_0$, $k_{prev} = 0$

2. While $|\mathcal{T}| < \frac{n}{2M+1}$                                          // Annealing loop

   (a) While $|\mathcal{T}| > k_{prev}$                   // If $|\mathcal{T}|$ not increased, increase $\beta$

       i. $k_{prev} = |\mathcal{T}|$
       ii. Split_PSTs($\mathcal{T}$, $P(T_*)$)
       iii. Soft_Clustering($\mathcal{T}$, $P(T_*)$, $\beta$)
       iv. Remove all $T_j$ such that $P(T_j) = 0$ from $\mathcal{T}$.

   (b) Increase $\beta$
```

Figure 5: **Unsupervised Sequence Segmentation Algorithm.**

As appeared in practical applications, the algorithm achieves better results when the BA loop is limited to a single pass. This gives the algorithm a possibility to correct the current set of PSTs $\mathcal{T}$ *while* looking for the optimal segmentation of $\bar{x}$ (see [Sel01]).

3.4 DA and the Segmentation Algorithm

The landscape of the problem defined in this section is typically riddled with local minima and it is computationally difficult to obtain the optimal solution. Usually a successful way of finding a good solution is through DA: a *series* of solutions to the soft clustering problem is found, starting from a low value of *resolution* parameter β and gradually increasing it, while allowing models to split in two when necessary.

The splitting procedure is straightforward: for each PST T in $\mathcal{T}$ we create two copies of T and perform random antisymmetric perturbations of the count vectors in each node of the two copies. Then we *replace* T with the two obtained PSTs while distributing $P(T)$ equally among them.

Now we are finally ready to outline the complete algorithm (see Fig. 5). We start with $\mathcal{T}$ including a single "average" PST T_0 that is trained on the whole sequence $\bar{x}$ with $w^0(x_i) = 1$ for all i. We pick an initial value of β, split $\mathcal{T}$ and proceed with the soft clustering procedure when initialized with the two models we got after split. We then split $\mathcal{T}$ again and repeat. If a model is found to have lost all its data it is eliminated. When the number of survived models stops increasing we increase β and then repeat the whole process. β is increased at most till the limit when the clusters become one window size.

Sets of segments that are assigned with high probability to the same model over a long range of β are *stable clusters* that contain important information about the statistical structure of our sample.

4 Multilingual Text Segmentation

In our first example we construct a synthetic text composed of alternating fragments of five other texts in five different languages: English, German, Italian, French and

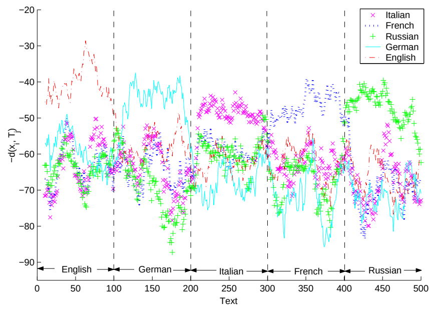


Figure 6: **Multilingual text segmentation**. The graph shows $-d(x_i, T_j)$ for the first 500 letters of text, for all T_j $\in \{\ T_{English},\ T_{German},\ T_{Italian},\ T_{French},\ T_{Russian}\ \}$. The true segmentation appears at the bottom of the graph.

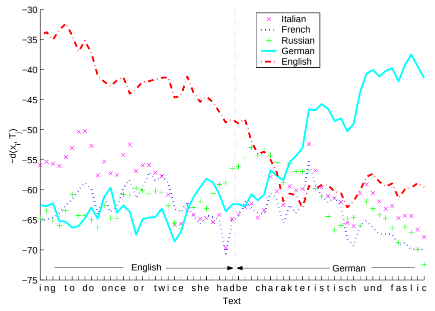


Figure 7: **Zoom in on a transition region from English to German**. The text on the x-axis corresponds to $i = 70..130$. Note the correspondence of the transition point into German to the English-like beginning "*be chara*kteristisch..." of the German segment.

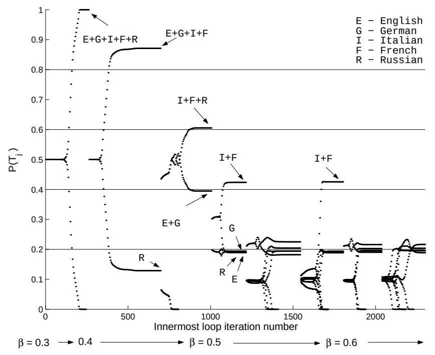


Figure 8: **Source separation**. The proportion of text assigned to each model is shown for all models as a function of the iterations of our innermost loop. Bifurcations are the splitting events and curves dropping off to zero are models dying out. Increments in β occur after the number of models converges at a given temperature. Languages captured by each model after the soft clustering convergence are pointed out. Notice how the order in which the languages separate from the primary joint model matches language relatedness. Unlike in a similar graph in [SBT01], here we limit the BA loop to single pass. The $\propto$-like curves after bifurcations (unlike $<$-like in [SBT01]) indicate a more thorough search over possible $\mathcal{T}$-s.

Russian, using standard transcripts to convert all into lower case Latin letters with blank substituting all separators. The length of each fragment taken is 100 letters, which means that we are switching languages every two sentences or so. The total length of the text is 150000 letters (30000 from each language).

We made several independent runs of our algorithm. In every run, after 2000-3000 accumulated innermost (BA) iterations we got a clear-cut, correct segmentation of the text into segments corresponding to the different languages, accurate up to a few letters (See Fig. 6, 7 for a typical example)[6]. Moreover, in all runs further splitting of all 5 language models resulted in starvation and subsequent removal of 5 extra models, taking us back to the same segmentation as before. Also, in most runs linguistically similar languages (English and German, French and Italian) separated at later stages of the segmentation process (Fig. 8 gives an example), suggesting a hierarchical structure over the discovered data sources.

[6]Correct segmentation was achieved even at a switching rate of 50 letters per segment, but of poorer quality.

5 Protein Sequences Classification

In this section we demonstrate the results of application of our algorithm to several protein families. The different training sets were constructed using the Pfam [BBD$^+$00, release 5.4] and Swissprot [BA00, release 38] databases. Various sequence domain families were collected from Pfam. In each Pfam family all members share a domain. An HMM detector is built for that domain based on an MSA of a seed subset of the family domain regions. The HMM is then verified to detect that domain in the remaining family members. Multi-domain proteins therefore belong to as many Pfam families as there are different characterized domains within them. In order to build realistic, more heterogeneous sets, we collected from Swissprot the *complete sequences* of all chosen Pfam families. Each set now contains a certain domain in all its members, and possibly various other domains appearing anywhere within some members.

There were two types of PST models we got in the process of clustering of the protein data: models that significantly outperform others on relatively short regions - these we call detectors; and models that perform averagely over all sequence regions - these are "protein noise" models. In what is following we analyse what kind of protein segments were selected by the detectors on three exemplary families. In general the "highlighted" segments may be characterized as "segments with highly conserved statistics (sequence), common to at least small amount of the input proteins". Being such, the detected segments may be seen as signatures (or fingerprints) of the domains, though in the cases of very conserved domains the complete domain may be covered by detector(s). The conservation usually indicates the key importance of the detected segment for the functioning of the whole domain. The amount (or percentage) of proteins sharing a similar segment among all the input proteins may be miserable and the similarity will still be found (in one example we have a domain that is common to only 12 out of 396 input proteins, and it still was altered). This is a clear and strong advantage of our approach compared to MSA, as will be demonstrated here.

In all the following examples we made several independent runs on each chosen family. For each family the different runs converged to the same final (stable) segmentation. In the presented graphs we show the segmentation of *single* representative protein sequences out of the explored families. The Swissprot accession number of the representative sequences shown will be written at the top of each graph.

The Pax Family

Pax proteins (reviewed in [SKG94]) are eukaryotic transcriptional regulators that play critical roles in mammalian development and in oncogenesis. All of them contain a conserved domain of 128 amino acids called the paired or paired box domain (named after the *drosophila* paired gene which is a member of the family). Some contain an additional homeobox domain that succeeds the paired domain. Pfam nomenclature names the paired domain "PAX".

The Pax proteins show a high degree of sequence conservation. One hundred and sixteen family members were used as a training set for the segmentation algorithm. In Fig. 9 we superimpose the prediction of all resulting PST detectors over one representative family member. This Pax6 SS protein contains both the paired and homeobox domains. Both have matching signatures. This also serves as an example where the signatures exactly overlap the domains. The graph of family members not

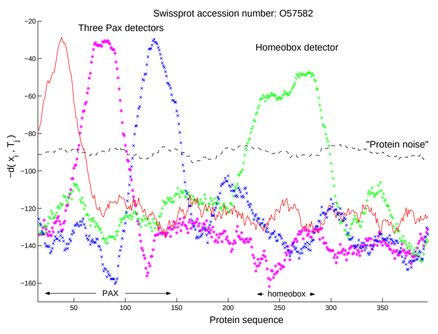


Figure 9: **Paired/PAX + homeobox signatures.** The graph shows the segmentation of PAX6 SS protein we got. At the bottom we denote in Pfam nomenclature the location of the two experimentally verified domains. These are in near perfect match here with the high scoring sequence segments.

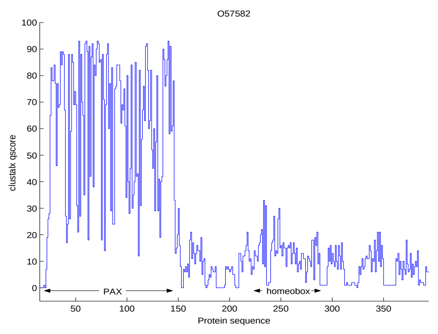


Figure 10: **Pax MSA profile conservation.** We plot the clustal X conservation score of the PAX6 SS protein against an MSA of all Pax proteins. While the predominant paired/PAX domain is discerned, the homeobox domain (appearing in about half the sequences) is lost in the background noise.

having the homeobox domain contains only the paired domain signature. Note that only about half of the proteins contain the homeobox domain and yet its signature is very clear.

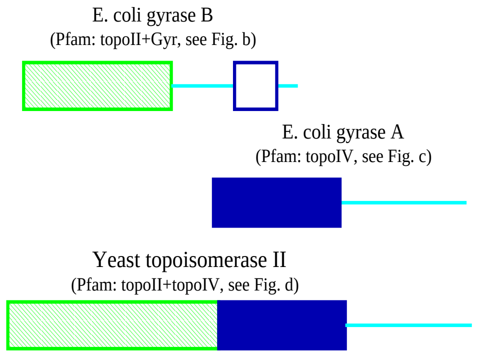
DNA Topoisomerase II

Type II DNA topoisomerases are essential and highly conserved in all living organisms (see [Roc95] for a review). They catalyze the interconversion of topological isomers of DNA and are involved in a number of mechanisms, such as supercoiling and relaxation, knotting and unknotting, and catenation and decatenation. In prokaryotes the enzyme is represented by the *Escherichia coli* gyrase, which is encoded by two genes, gyrase A and gyrase B. The enzyme is a tetramer composed of two gyrA and two gyrB polypeptide chains. In eukaryotes the enzyme acts as a dimer, where in each monomer two distinct domains are observed. The N-terminal domain is similar in sequence to gyrase B and the C-terminal domain is similar in sequence to gyraseA (Fig. 11.a). In Pfam 5.4 terminology gyrB and the N-terminal domain belong in the "DNA_topoisoII" family[7], while gyrA and the C-terminal domain belong in the "DNA_topoisoIV" family[8]. Here we term the pairs gyrB/topoII and gyrA/topoIV.

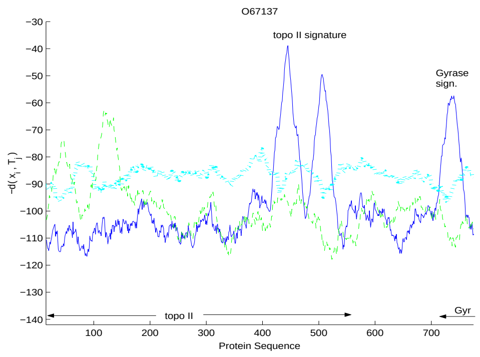
For the analysis we used a group of 164 sequences that included both eukaryotic topoisomerase II sequences and bacterial gyrase A and B sequences (gathered from the union of the DNA_topoisoII and DNA_topoisoIV Pfam 5.4 families). We successfully differentiate them into sub-classes. Fig. 11.d describes a representative of the eukaryotic topoisomerase II sequences and shows the signatures for both domains, gyrB/topoII and gyrA/topoIV. Fig. 11.b and Fig. 11.c demonstrate the results for representatives of the bacterial gyrase B and gyrase A proteins, respectively. The *same* two signatures are found in all three sequences, at the appropriate

[7]Apparently this family has been sub-divided in Pfam 6 releases.

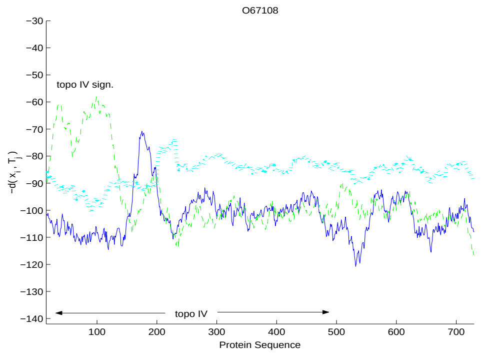
[8]The name should not be confused with the special type of topoisomerase II found in bacteria, that is also termed topoisomerase IV, and plays a role in chromosome segregation.


E. coli gyrase B
(Pfam: topoII+Gyr, see Fig. b)
E. coli gyrase A
(Pfam: topoIV, see Fig. c)
Yeast topoisomerase II
(Pfam: topoII+topoIV, see Fig. d)

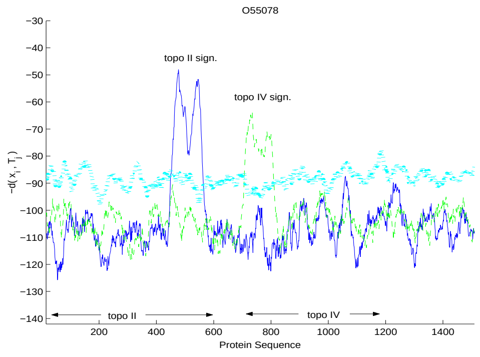
(a) Fusion event illustration



(b) E. coli gyrase B



(c) E. coli gyrase A



(d) Yeast topoisomerase II

Figure 11: **DNA topoisomerase II.** (a) - Fusion event illustration, adapted from [MPN+99]. The Pfam domain names are added in brackets, together with a reference to our results on a representative homolog. Compare the PST signatures in figures (b)-(d) with the schematic drawing in (a). It is clear that the eukaryotic signature is indeed composed of the two prokaryotic ones, in the correct order, omitting the C-terminus signature of gyrase B (short termed here as "Gyr").

locations. Interestingly, in Fig. 11.b in addition to the signature of the gyrB/topoII domain *another* signature appears at the C-terminal region of the sequence. This signature is compatible with a known conserved region at the C-terminus of gyrase B,[9] that is involved in the interaction with the gyrase A molecule.

The relationship between the *E. coli* proteins gyrA and gyrB and the yeast topoisomerase II (Fig. 11.a) provides a prototypical example of a fusion event of two proteins that form a complex in one organism into one protein that carries a similar function in another organism. Such examples have lead to the idea that identification of those similarities may suggest the relationship between the first two proteins, either by physical interaction or by their involvement in a common pathway. The computational scheme we present can be useful in search for these relationships.

[9]Corresponding to the Pfam "DNA_gyraseB_C" family.

The Glutathione S-Transferases (GST)

The glutathione S-transferases (GST) represent a major group of detoxification enzymes (reviewed in [HP95]). There is evidence that the level of expression of GST is a crucial factor in determining the sensitivity of cells to a broad spectrum of toxic chemicals. All eukaryotic species possess multiple cytosolic GST isoenzymes, each of which displays distinct binding properties. A large number of cytosolic GST isoenzymes have been purified from rat and human organs. On the basis of their sequences they have been clustered into five separate classes designated class alpha, mu, pi, sigma, and theta GST. The hypothesis that these classes represent separate families of GST is supported by the distinct structure of their genes and their chromosomal location. The class terminology is deliberately global, attempting to include as many GSTs as possible. However, it is possible that there are sub-classes that are specific to a given organism or a group of organisms. In those sub-classes the proteins may share more than 90% sequence identity, but these relationships are masked by their inclusion in the more global class. The classification of a GST protein with weak similarity to one of these classes is sometimes a difficult task. In particular, the definition of the sigma and theta classes is imprecise. Indeed in the PRINTS [ACF$^+$00] database only the three classes, alpha, pi, and mu have been defined by distinct sequence signatures, while in Pfam all GSTs are clustered together, for lack of sequence dissimilarity.

Three hundred and ninety six Pfam family members were segmented jointly by our algorithm, and the results were compared to those of PRINTS (as Pfam classifies all as GSTs). Five distinct signatures were found (not shown due to space limitations): (1) A typical weak signature common to many GST proteins that contain no sub-class annotation. (2) A sharp peak after the end of the GST domain appearing exactly in all 12 out of 396 (3%) proteins where the elongation factor 1 gamma (EF1G) domain succeeds the GST domain. (3) A clear signature common to almost all PRINTS annotated alpha and most pi GSTs. The last two signatures require more knowledge of the GST superfamily. (4) The theta and sigma classes are abundant in nonvertebrates. As more and more of these proteins are identified it is expected that additional classes will be defined. The first evidence for a separate sigma class was obtained by sequence alignments of S-crystallins from mollusc lens. Although these refractory proteins in the lens probably do not have a catalytic activity they show a degree of sequence similarity to the GSTs that justifies their inclusion in this family and their classification as a separate class of sigma [BE92]. This class, defined in PRINTS as S-crystallin, was almost entirely identified by the fourth distinct signature. (5) Interestingly, the last distinct signature, is composed of two detector models, one from each of the previous two signatures (alpha + pi and S-crystallin). Most of these two dozens proteins come from insects, and of these most are annotated to belong to the theta class. Note that many of the GSTs in insects are known to be only very distantly related to the five mammalian classes. This putative theta sub-class, the previous signatures and the undetected PRINTS mu sub class are all currently further investigated.

Comparative results

In order to evaluate our findings we have performed three unsupervised alignment driven experiments using the same sets described above: an MSA was computed for each set using clustal X [JTG$^+$98, Linux version 1.81]. We let clustal X compare the level of conservation between individual sequences and the computed MSA profile in each set. Qualitatively these graphs resemble ours, apart from the fact that they do not offer separation into distinct models.

We briefly recount some results: the Pax alignment did not clearly elucidate the homeobox domain existing in about half the sequences (see Fig. 10). For type II topoisomerases the Gyrase B C-terminus unit from Fig. 11.b can be discerned from the main unit, but with a much lower peak. And the clear sum of two signatures we obtained for the eukaryotic sequences (Fig. 11.d) is lost. In the last and hardest case the MSA approach tells us nothing. All GST domain graphs look nearly identical precluding any possible subdivision. And the 12 (out of 396) instances of the EF1G domain are completely lost at the alignment phase.

6 Discussion

The sequence segmentation algorithm we describe and evaluate in this paper is a combination of several different information theoretic ideas and principles, naturally combined into one new coherent procedure. The core algorithm, the construction of PST, is essentially a source coding *loss-less compression* method. It approximates a complex stochastic sequence by a Markov model with variable memory length. The power of this procedure, as demonstrated on both natural texts and on protein sequences [RST96, BY01], is in its ability to capture short strings (suffixes) that are significant predictors - thus good features - for the statistical source. We combine the PST construction with another information theoretic idea - the MDL principle - and obtain a more efficient estimation of the PST, compared with its original learning algorithm.

Our second key idea is to embed the PST construction in a *lossy compression* framework by adopting the rate-distortion theory into a competitive learning procedure. Here we treat the PST as a model of a single statistical source and use the rate distortion framework (the BA algorithm) to partition the sequences between several such models in an optimal way. Here we specifically obtain a more expressive statistical model, as *mixtures* of (short memory, ergodic) Markov models lay outside of this class, and can be captured only by much deeper Markov models. This is a clear advantage of our current approach over mixtures of HMMs (as done in [FST98]) since mixtures of HMMs are just HMMs with constrained state topology.

The analogy with rate-distortion theory enables us to take advantage of the trade-off between compression (rate) and distortion, and use the Lagrange multiplier β, required to implement this trade-off, as a resolution parameter. The deterministic annealing framework follows naturally in this formulation and provides us with a simple way to obtain hierarchical segmentation of very complex sequences. As long as the underlying statistical sources are distinct enough, compared to the average alternation rate between them, our segmentation scheme should perform well.

Our experiments with protein families demonstrated a number of clear advantages of the proposed algorithm: it is fully automated; it does not require or attempt an MSA of the input sequences; it handles heterogeneous groups well and locates domains appearing only few times in the data; by nature it is not confused by different module orderings within the input sequences; it appears to seldom generate false positives; and it is shown to surpass HMM clustering in at least one hard instance. Together with the tremendous success on the multilingual text data we get a strong evidence that our algorithm is a new powerful tool for sequence analysis that worth further development and examination on additional types of data.

References

[AB00] A. Apostolico and G. Bejerano. Optimal amnesic probabilistic automata or how to learn and classify proteins in linear time and space. *J. Comput. Biol.*, 7(3):381–393, 2000.

[ACF+00] T. Attwood, M. Croning, D. Flower, A. Lewis, J. Mabey, P. Scordis, J. Selley, and W. Wright. PRINTS-S: the database formerly known as PRINTS. *Neuc. Acids Res.*, 28(1):225–227, 2000.

[BA00] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Neuc. Acids Res.*, 28(1):45–48, 2000.

[BBD+00] A. Bateman, E. Birney, R. Durbin, S. Eddy, K. Howe, and E. Sonnhammer. The Pfam protein families database. *Neuc. Acids Res.*, 28(1):263–266, 2000.

[BE92] T. Buetler and D. Eaton. Glutathione S-transferases: amino acid sequence comparison, classification and phylogentic relationship. *Environ. Carcinogen. Ecotoxicol. Rev.*, C10:181–203, 1992.

[BK96] P. Bork and E. Koonin. Protein sequence motifs. *Curr. Opin. Struct. Biol.*, 6(3):366–376, 1996.

[BRY98] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Trans. Info. Theory*, 44:2743–2760, 1998.

[BSMT01] G. Bejerano, Y. Seldin, H. Margalit, and N. Tishby. Markovian domain fingerprinting: statistical segmentation of protein sequences. *Bioinform.*, 17(10), 2001.

[BY01] G. Bejerano and G. Yona. Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinform.*, 17(1):23–43, 2001.

[CT91] T. Cover and J. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, 1991.

[DEKM98] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids.* Cambridge University Press, 1998.

[FR95] Y. Freund and D. Ron. Learning to model sequences generated by switching distributions. In *Comput. Learn. Theory (COLT) 8*, volume 8, pages 41–50, New York, NY, July 1995. ACM press.

[FST98] S. Fine, Y. Singer, and N. Tishby. The hierarchical Hidden Markov Model: Analysis and applications. *Mach. Learn.*, 32:41–62, 1998.

[HP95] J. Hayes and D. Pulford. The glutathione S-transferase supergene family: regulation of GST and the contribution of the isoenzymes to cancer chemoprotection and drug resistance. *Cri. Rev. Biochem. Mol. Biol.*, 30(6):445–600, 1995.

[JTG+98] F. Jeanmougin, J. Thompson, M. Gouy, D. Higgins, and T. Gibson. Multiple sequence alignment with clustal X. *Trends Biochem. Sci.*, 23:403–405, 1998.

[MPN+99] E. Marcotte, M. Pellegrini, H. Ng, D. Rice, T. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285(5428):751–753, 1999.

[Roc95] J. Roca. The mechanisms of DNA topoisomerases. *Trends in Biol. Chem.*, 20:156–160, 1995.

[Ros98] K. Rose. Deterministic annealing for clustering, compression, classification, regression and related optimization problems. *IEEE Trans. Info. Theory*, 80:2210–2239, November 1998.

[RST96] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Mach. Learn.*, 25:117–149, 1996.

[SBT01] Y. Seldin, G. Bejerano, and N. Tishby. Unsupervised sequence segmentation by a mixture of switching variable memory markov sources. *Proc.* 18^{th} *Intl. Conf. Mach. Learn. (ICML)*, 2001.

[Sel01] Y. Seldin. On unsupervised learning of mixtures of Markovian sources. Master's thesis, The Hebrew University of Jerusalem, 2001. (To appear).

[SKG94] E. Stuart, C. Kioussi, and P. Gruss. Mammalian Pax genes. *Annu. Rev. Genet.*, 28:219–236, 1994.