

Incremental Construction of Structured Hidden Markov Models

Hidden Markov Model, Sequence Analysis, Data Mining

Abstract

This paper presents an algorithm for inferring a Structured Hidden Markov Model (S-HMM) from a set of sequences. The S-HMMs are a sub-class of the Hierarchical Hidden Markov Models and are well suited to problems of process/user profiling. The learning algorithm is unsupervised, and follows a mixed bottom-up/top-down strategy, in which elementary facts in the sequences (motifs) are progressively grouped, thus building up the abstraction hierarchy of a S-HMM, layer after layer. The algorithm is validated on a suite of artificial datasets, where the challenge for the learning algorithm is to reconstruct the model that generated the data. Then, an application to a real problem of molecular biology is briefly described.

1 Introduction

Several data mining applications, such as genome analysis [Durbin *et al.*, 1998], intrusion detection [Lee and Stolfo, 1998; Lee *et al.*, 2002] and agent modeling, are characterized by the presence of *sparse patterns* hidden in symbolic sequences. By *sparse patterns* we mean a chain of interrelated *motifs* separated by gaps, i.e., irrelevant subsequences.

Discovering this kind of patterns becomes harder when the length of the motifs decreases and the length of gaps increases. In fact, short motifs have a high probability of occurring in a long random sequences. Then, by considering motifs in isolation, short subsequences corresponding to true regularities are easily missed, as they cannot be distinguished from random ones. On the contrary, short motifs can be discovered when they systematically co-occur together with other motifs, previously identified.

This paper presents an algorithm, EDY, which explicitly addresses this problem. EDY models sparse patterns by means of *Structured Hidden Markov Models* (S-HMM), which can be automatically induced from a database of sequences. S-HMMs benefit from interesting compositional properties, which allow for their incremental construction. More specifically, EDY's discovering algorithm executes a cycle, at each iteration of which new motifs are detected and added to the current model, starting from the most evident ones.

A suite of artificial problems has been designed to test EDY; the algorithm accuracy is correlated to the problem degree of difficulty. Furthermore, an application to a DNA analysis problem is described; in this application EDY's incremental capabilities allow short motifs, forming together complex patterns, to be discovered inside very long DNA strings. The results have been considered significant by the biologists.

2 The Structured HMM

Assuming that possible instances of a sparse pattern can be represented by a first order Markov chain, the global structure of the pattern can be modeled by a forward graph as the one in Figure 1. Formally, the graph is defined by a 5-ple $\gamma = \langle M, G, A, b, e \rangle$, where M and G are two sets of abstract states (nodes of the graph), denoted by circles and squares, respectively. A is a probability distribution governing the transitions from one state to another. States b and e are the initial and the final states of any path on γ , representing an instance of the pattern. By definition, b and e are dummy states. Under the restriction that γ is a strictly forward graph, distribution A is such that no loops are generated.

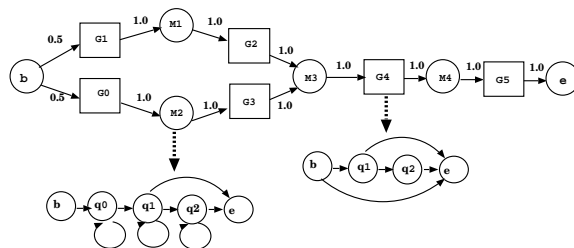


Figure 1: Example of *sparse pattern* model. Circles denote motifs and squares denote gaps.

In turn, motifs and gaps are sub-patterns that are modeled by left-to-right Hidden Markov Models (LR-HMMs) [Rabiner, 1989]. An LR-HMM corresponds to a forward graph where self-loops are allowed. Formally, an LR-HMM is a 6-ple $\mu = \langle Q, O, A, B, b, e \rangle$, where Q is a set of states, O a set of observations (symbols of an alphabet Σ), A is a probability distribution governing the transitions from state to state, B is a probability distribution governing the emission of ob-

servation $o_j \in O$ in state $q_i \in Q$, and b, e are the initial and final states, respectively. We assume that b and e do not have observable emissions.

The model γ , together with the motif and gap models associated to its states, defines a Structured HMM (S-HMM)¹ An S-HMM is a particular case of a Hierarchical Hidden Markov Model (HHMM) (see [Fine *et al.*, 1998], for details), with only two levels in the hierarchy. An HHMM is a multilevel extension of the basic HMM [Rabiner, 1989], in which emissions at higher levels are sequences computed by sub-models at a lower level in the hierarchy.

Under the given restrictions, two important properties (not proven here) hold for S-HMMs, whereas they do not hold for unrestricted HHMMs.

Property 1: Any S-HMM can be compiled into a single level Left-to-Right HMM without losing the structure enforced by the hierarchy.

In other words, flattening an S-HMM into a single level LR-HMM, denoted by *LR-HMM'*, can be simply done by replacing the abstract states defined in γ with the models LR-HMMs of motifs and gaps.

Property 2: The complexity of the Forward-Backward and Viterbi algorithms for LR-HMMs is $O(k, |S|, n)$, where n is the length of the observed sequence, k is the average branching factor, and $|S|$ is the number of states of the corresponding *LR-HMM'*.

As the EM algorithm is based on the Forward-Backward algorithm, every EM cycle has the same complexity. This property is very important for dealing with long sequences and complex S-HMMs.

The presence of long gaps between motifs is a problem that needs to be explicitly addressed. A simple way of modeling a short gap is to use a state with a self-loop in between two motifs. However, this simple approach does not work when gaps are long. In fact, a long gap may hide a complex process that needs to be completed before entering the phase generating the next motif. Then, the gap lengths may follow a distribution different from the exponential one, which is the unique distribution associated to a self-loop. An example of a more plausible distribution is reported in Figure 2-(a), which is correctly modeled by the LR-HMM in Figure 2-(b). It is easy to verify that, given a sufficient number of states, this HMM architecture can model any distribution.

3 EDY's Discovery Strategy

The EDY algorithm exploits co-occurrence of regularities in bounded regions of a sequence in order to detect short motifs. The S-HMM λ of a sparse pattern is constructed incrementally, starting from a learning set \mathcal{LS} of sequences, going through a learning cycle in which a model is progressively extended and refined, by repeatedly incorporating new motifs

¹In the following symbol λ will be used to denote a S-HMM, whereas symbols γ will denote the upper level graph of the model, and μ will denote motif and gap models.

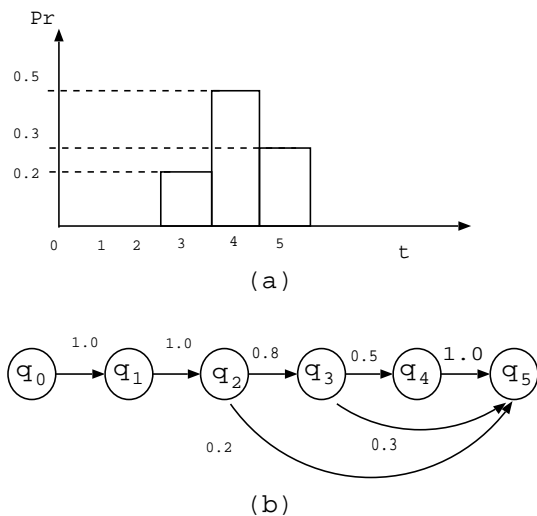


Figure 2: Hidden Markov Model for gaps. (a) Example of a probability distribution over the lengths of a gap, determined by some physical process; (b) Left-to-right unfolded model that correctly generates distribution (a).

and gaps. The cycle may initiate with a *empty model* or with a model supplied by an expert of the domain, and terminates when there is no more evidence of new motifs to incorporate. When the cycle initiates with an empty model, a first step is made in order to construct a seed; in this case, the whole length of the sequences of the learning set \mathcal{LS} is considered, searching for self-evident motifs, or for pairs of motifs, which are easier to distinguish from random noise. Afterwards, the incremental cycle begins. Suppose that EDY has built a current model λ . Abstract states in the upper level γ of λ are associated to motifs and gaps that have been found in corresponding subsequences of the sequences in \mathcal{LS} . In general, motifs and gaps are interleaved. In order to extend λ , EDY analyzes any gap searching for new motifs. Good candidates are:

1. motifs that show a high frequency of occurrence;
2. motifs that occur at approximately constant distance from one of the gap boundaries;
3. pairs of motifs whose inter-distance follows a peaked distribution.

After candidate motifs are generated (with any of the strategies), the problem arises of evaluating their likelihood, distinguishing true motifs from apparent motifs due to randomness.

Let s be the subsequence where candidate motifs have been found. The likelihood that a candidate motif m is not random is usually evaluated [Durbin *et al.*, 1998; Gussfield, 1997] by comparing the probability $P(m|s)$ of finding m in s to the probability $P(m|r)$ of finding m in a random sequence r of the same length as s . It is immediate to verify that the probability $P(m|r')$, being r' a contiguous subsequence of r , is always smaller than the probability $P(m|r)$ of finding m in the entire r . Therefore, the likelihood of m not being random

increases when the length of the region where it may occur is reduced.

However, computing $P(m|s)$ and $P(m|r)$ is not simple when the motif m is an instance of a model μ . In fact, m is just one possibility among all those governed by the emission probability distribution of μ .

On the other hand, the *forward-backward* algorithm [Rabiner, 1989] can compute the probability $P(s|\lambda)$ that a sequence s (or r) is generated by λ . In the following we will show how this algorithm can be used to estimate the reliability of accepting a motif m as not random.

Using Bayes theorem, we get: $P(s|\lambda)P(\lambda) = P(\lambda|s)P(s)$. Given the set S of the subsequences of sequences in \mathcal{LS} and another set R of random sequences, whose lengths follow the same distribution as those in S , the following relation:

$$\rho(\lambda, S) = \frac{E[P(s|\lambda)]}{E[P(r|\lambda)]} = \frac{E[P(\lambda|s)P(s)]}{E[P(\lambda|r)P(r)]}, \quad s \in S, r \in R. \quad (1)$$

can be considered as an estimate of the reliability of accepting the hypothesis that model λ generates a motif rather than a random string in the sequences of S . Notice that λ represents the model for the whole subsequence s , in which a motif instance m , modeled by μ , has been hypothesized. The model λ will be a sequence of a gap model, a motif (i.e., μ) and another gap model (in the case of a pair of candidate motifs λ 's structure is changed accordingly).

In order to understand the meaning of $\rho(\lambda, S)$, we have to remember that the subsequence m (or variations thereof) must be found in most sequences of S to become a candidate motif. Then, the probability value computed by the *forward-backward* algorithm may differ from one sequence s from another. Then, the mean value $E[.]$ appearing in the formula refers to the average w.r.t. to sequences in S .

The value $\rho(\lambda, S)$ must be significantly greater than 1, if the candidate motif is not random. A Student's test (with $p > 0.99$) is made in order to compare $E[P(s|\lambda)]$ and $E[P(r|\lambda)]$. The $\rho(\lambda, \mathcal{LS})$ value can also be used to compare competing motif models.

4 Learning Algorithm

Before describing in details the learning algorithm, we need to describe *sequence tagging* and *sequence abstraction*, which are the basic procedures of the learning strategy.

Sequence tagging. Let λ_t denote the current version of the S-HMM, constructed by EDY after t iterations from a learning set \mathcal{LS} . Sequence tagging is accomplished by using the Viterbi algorithm to find, in each sequence $s \in \mathcal{LS}$, the most likely instances of λ_t . From these instances it is easy to determine the regions where most likely the motifs and gaps described by λ_t occur. Such regions are tagged with the *id* of the corresponding motif and gap models. In the following $\mathcal{LS}(\lambda_t)$ will denote the set of learning sequences tagged using λ_t .

Sequence abstraction. After sequence tagging has been done, an abstract description $s'(\lambda_t)$ can be generated, for each sequence $s \in \mathcal{LS}$, by replacing the tagged regions with the

corresponding motif or tag *id*. In the following, $\mathcal{LS}'(\lambda_t)$ will denote the set of all sequences abstracted using λ_t .

The learning algorithm iteratively performs a cycle in which two operators can be applied: the *Extend* and the *Refine* operators. Let **HALT** denotes the variable that controls the overall cycle execution. The abstract scheme of the learning algorithm is the following one:

```

EDY( $\lambda$ )
STABLE = False, HALT = False
while  $\neg$  HALT do
  while  $\neg$  STABLE do
     $\lambda_{new} = Refine(\lambda)$ 
    if  $\mathcal{LS}(\lambda_{new}) \simeq \mathcal{LS}(\lambda)$ 
      then STABLE = True
    endif
     $\lambda = \lambda_{new}$ 
  endwhile
  Apply Extend( $\lambda$ )
  if Extend( $\lambda$ ) fails
    then HALT = True
  else  $\lambda_{new} = Extend(\lambda)$ 
        $\lambda = \lambda_{new}$ 
  endif
endwhile

```

In the next subsection the functioning of the algorithm will be briefly illustrated.

4.1 Model Extension

Given the current model $\lambda = \lambda_t$, the algorithm applies the *Refine* operator until (approximately) no difference exists, in the tagged sequences in $\mathcal{LS}(\lambda_t)$, between two consecutive cycles. When this happens, EDY tries to extend the current model, by adding some motif discovered inside a gap. However, a candidate motif is not substituted to the gap, but both are kept in parallel, waiting for the *Refine* operator to decide. Notice that at most one candidate motif is added in an extension step, with the only exception in the first cycle, where a more complex initial model, containing two motifs, may be constructed.

In the following we will briefly overview the heuristic procedure used for generating hypotheses for motifs in gaps, and for building an S-HMM, to be validated according to the formula (1). In order to find motifs, EDY searches for regularities exploiting techniques developed in Molecular Biology [Durbin *et al.*, 1998]. Two basic notions are used: *edit distance* and *alignment* among strings. Informally, the edit distance is measured as the number of corrections necessary to make two strings identical. The alignment of two strings s_1, s_2 is a pair of strings s'_1, s'_2 obtained by s_1, s_2 by inserting a proper number of spaces such that identical symbols are put into correspondence to a maximum extent. An alignment is said *local* when it is restricted to a subsequence of s_1, s_2 , and it is said *multiple* when it involves more than two strings.

Therefore, motifs are discovered and modeled according to the following steps:

1. For every pair of sequences (s_1, s_2) in \mathcal{LS} , or pairs of subsequences where a gap has been found, EDY finds all local, statistically significant *alignments* between them, and collects the aligned subsequences into a set **A**. Subsequences in **A** are the candidate motif instances.
2. Subsequences in **A** are then grouped, forming three kinds of clusters: (a) clusters of highly frequent (disregarding the position) subsequences, bearing a strong

similarity among them; (b) clusters of similar subsequences that occur at an almost constant distance from one of the boundaries of the sequence; (c) pairs of clusters of similar subsequences that frequently occur at a regular distance from one another. Levenstein’s distance [Levenstein, 1966] is used between sequences.

3. Every cluster C_i from the previous step is used to construct a corresponding LR-HMM μ_i , using the algorithm described in [Durbin *et al.*, 1998]. The algorithm first constructs a multiple alignment among all subsequences in C_i , and then it builds a model μ_i from the aligned subsequences.
4. Gap models are then constructed, on the basis of their length distribution.
5. From every motif model μ_i and the models of the adjacent gaps a partial S-HMM λ_i is constructed and evaluated, as explained in Section ???. Among all the discovered motif models, the one which obtains the best evaluation is selected for actually extending the model.

4.2 Model Refinement

As the mechanism exploited by the model extension procedure is rather primitive, it is not able to reliably collect all and only a motif instances. Then, the model refinement procedure reconstructs motif and gap models at every step, until convergence on stable models is achieved.

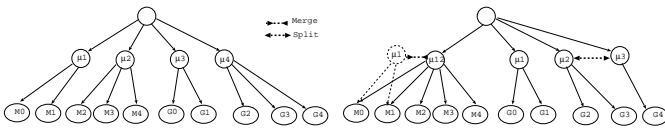


Figure 3: Example of cluster hierarchy. Leaves corresponds to the states of the level γ , whereas second level nodes correspond to models μ of motifs and gaps

The starting point is the tagged dataset $\mathcal{LS}(\lambda_t)$ constructed before calling the refinement operator. All sequence segments corresponding to motif and gap instances, which have been detected by the Viterbi algorithm, are collected into a two level hierarchical clustering. The left part of Figure 3 provides an example of the clusters for the S-HMM in Figure 1. The clusters associated to the leaves correspond to states at the level γ of the model. Each leaf contains all the subsequences which have been emitted by the model μ (motif or gap) when the S-HMM was in the corresponding state. However, emissions in different states can be generated by the same motif/gap model. Then, the clusters at the second level group together the leaves whose elements are generated by the same model μ , but in different states. The root node of the tree is a dummy node, representing the whole set of segments. During the refinement process, second level clusters can be split or merged (see right part of Figure 3), thus increasing or decreasing the set of existing motif/gap models. Given a distance measure between instances (the edit distance in the specific case), two clusters of motif/gap instances are merged if the distance between their centers is

not greater than their average intra-cluster distance. Alternatively, a cluster, whose children have an intra-cluster distance much smaller than the inter-cluster distance, may be split.

The specific operators, which are applied in a refinement step, are briefly described in the following.

Boundary refinement - This operator is meant to correct possible segmentation errors performed during the initial learning phase. Before trying to refine a motif model, the algorithm for searching local alignments is run on the new set of instances, but allowing the alignments to possibly extend into the adjoining gap regions for one or two positions. Instances of the motif can thus be extended (or reduced) if the original segmentation is found inaccurate. However, this operator is only applied a few times when a new motif is constructed, because, in the long term, it may cause instability.

Model diversification - If μ is a model associated to two different states M_j, M_k of level γ , and the two associated instance clusters C_j and C_k significantly differ, then μ is split into μ_j and μ_k , which are trained on C_j and C_k , respectively.

Model unification - When two models μ_j and μ_k have children that cannot be distinguished among themselves according to the distance criterion, the models can be merge into a single one, μ , whose parameters can be estimated from the cluster obtained as union of μ_j and μ_k ’s children. The procedure for merging gap models is analogous, but based on a different criterion. More precisely, considering two clusters C_j and C_k of gap instances, the histograms h_j and h_k of the corresponding gap lengths are constructed. Histograms are compared among each other, and ”similar” ones are merged. This operator is only activated optionally, as it may slow down convergence to a stable hierarchy.

Parameter refinement - As the instances of a model may be currently different from those used to initially learn it, the model’s parameters are re-estimated from the new set of instances.

Gap model refinement - This operator is similar to the preceding one, except that the parameters to be estimated are those appearing in the distribution of the gap lengths.

Hierarchy Revision. The algorithm for the construction/reconstruction of the level γ of the S-HMM is very similar to the one that constructs the motif models. The difference is that it works on the abstracted sequences belonging to \mathcal{LS}' .

As the above algorithm is computationally inexpensive, it is repeated at every refinement step, in order to propagate to the upper level changes in the structure at the lower level.

5 Validation on Artificial Data

The algorithm has been validated using artificial sequence sets, where known patterns have been hidden. The challenge for the algorithm was to reconstruct the original model from the data. Two groups of target S-HMMs have been constructed and used to generate a large number of sequence datasets. The S-HMMs in the first group contain three motifs separated by two gaps, plus an initial and a final random gap. The S-HMMs in the second group have a similar, but more complex, structure. They encode a sequence of six motifs separated by 5 gaps.

Using a semi-automated procedure, 768 models (384 for each group) have been constructed; they differ in the nominal length of the motifs (5, 8, 11, 15 symbols), in the cardinality of the alphabet (4, 7, 14, 25 symbols) and in the probability distribution controlling transitions from state to state and symbol emission inside states. More specifically, four classes of normal distributions (N0, N1, N2, N3) of increasing variance have been considered. For every setting of the above parameters three different models have been generated. They differ one from another for a small perturbation in the center locations of the probability distributions. Finally, for every model, a learning set and a test set, each containing 100 sequences, have been generated.

The sequence length ranges from 800 to 1500. It is worth noticing that, considering the quite short motif length, the coding part is much smaller than the non coding part appearing in the gaps.

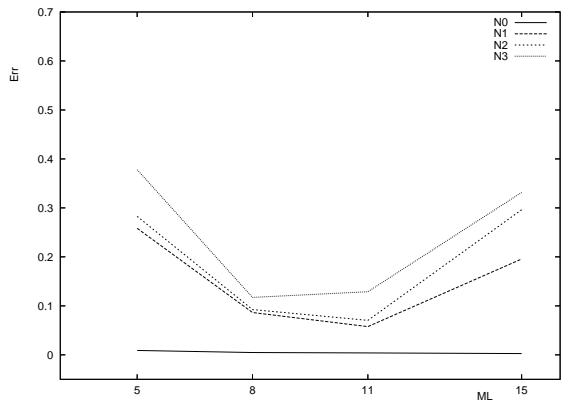


Figure 4: EDY’s performances on the sequences generated by models in Group 2 (6 motifs). The plot reports the error $Err = Err(\lambda_D)$ on the test set versus the motif length $ML \in \{5, 8, 11, 15\}$.

The perturbation effect on the sequences, due to the increase of the standard deviation in the probability distribution, has been evaluated as the average edit distance δ_E between the motif instances occurring in a dataset and the maximum likelihood instance, computed from the generative model by the Viterbi algorithm. The following average values have been obtained for the four distributions:

Class:	N0	N1	N2	N3
δ_E :	0.0	0.11	0.19	0.28

Notice that also the gap length spread is strongly affected by the increase in the distribution spread, even if it is not accounted for in the measures reported above.

In order to evaluate EDY’s accuracy, let λ_D be the model learned by the algorithm, and λ_T the target model, used to generate the data. Let moreover $\mathcal{TS}(\lambda_D)$ denote the test set \mathcal{TS} tagged with λ_D and $\mathcal{TS}(\lambda_T)$ the one tagged with λ_T . The error $Err(\lambda_D)$ of λ_D on \mathcal{TS} is measured as the average edit distance between the motif instances in $\mathcal{TS}(\lambda_D)$ and the motifs instances in $\mathcal{TS}(\lambda_T)$, divided by the length of the motif instances in $\mathcal{TS}(\lambda_T)$.

The performances obtained by EDY on the two groups of datasets are similar, even though the ones on the first group are slightly better. For the sake of brevity, only the errors on the second group (more difficult) are reported in Figures 4 and 5. As one may expect, EDY always finds an error-free model when motifs are not affected by noise (gaps are always filled with random noise).

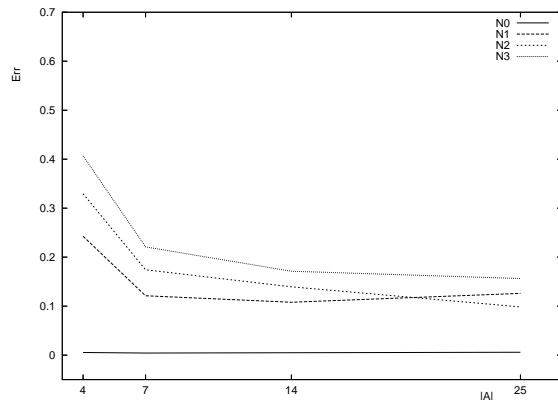


Figure 5: EDY’s performances on the sequences generated by models in Group 2 (6 motifs). The plot reports the error $Err = Err(\lambda_D)$ on the test set versus the alphabet cardinality $|\mathbf{A}| \in \{5, 7, 14, 25\}$.

In presence of noise, it appears that $Err(\lambda_D)$ increases when the alphabet cardinality and the motif length decrease, as well as when the standard deviation of the target model increases, as it is reasonable to expect. In fact, when the alphabet is small, it is more difficult to distinguish real motifs from apparent regularities due to randomness. For the same reason, short motifs are more difficult to detect. Then, the performance degradation is due, in general, to the failure of the algorithm, which searches for new motifs without finding the correct ones. However, it is surprising that the accuracy decreases again when motifs become longer than 11 symbols. A possible explanation is the following: when the average length of a motif instances increases in presence of noise, the number of alternative sequences, among which the correct instances of the motif are to be identified, increases, smoothing thus the similarity among strings and increasing confusion.

The decrease in the similarity between the target model and the discovered model, when the probability distributions have long tails, is also in agreement with what one expects. Nevertheless, it is interesting that the error rate remains comparable to the level of noise of the dataset. It is also worth noticing that the performances evaluated on the test sets and on the learning sets are almost identical, as their differences are not statistically significant.

Finally, the system always converged to a stable model in a number of steps ranging from 11 to 35. The computational complexity for solving a single problem of the second group corresponds to a cpu time ranging from 30 to 40 minutes on a Opteron.

6 Discovering Regulatory Control in Proteins

EDY has been applied to a biological task, inside a project aimed at discovering the expression regulators of proteins responsible for obesity. Currently, the experimentation in field is still made with animals, specifically with *ractus norvegicus*. A group of 49 proteins, involved in obesity and sensible to hypoxia (reduced presence of oxygen in the atmosphere), has been identified [R C. Roach, 2003]. All such proteins share a very similar behavior: the conjecture is that they could share also the genetic regulators (motifs), which control their expression in the phenotype. Goal of the experiment is to discover such regulators. The EDY algorithm is used to select candidate motifs, which will be tested later on in a bio-genetic laboratory. Therefore, the ultimate evaluation of the results produced by EDY will be available only in several months. Here we will show how EDY was able to deal with complex sequences, and we will present some of the obtained results, which can be at least partially validated.

The biologists suppose that the loci of the regions controlling protein expression should found in the sequence of 2000 nucleotides preceding the gene coding for the protein. A first attempt of running EDY on the whole dataset of 49 sequences failed. On the contrary, exploiting EDY's incremental facilities, the results reported in Figure 6 have been obtained.

The key point has been to exploit the known right boundary between the promoter and the gene as an anchor for building an S-HMM capturing the regularities found in the region upstream the gene. Then, EDY was run several times, progressively extending the explored region of the sequences from 500 nucleotides until 2000 nucleotides upstream. After the explored region exceeded the size of 1000 nucleotides, the algorithm systematically begun to generate the S-HMM reported in Figure 6.

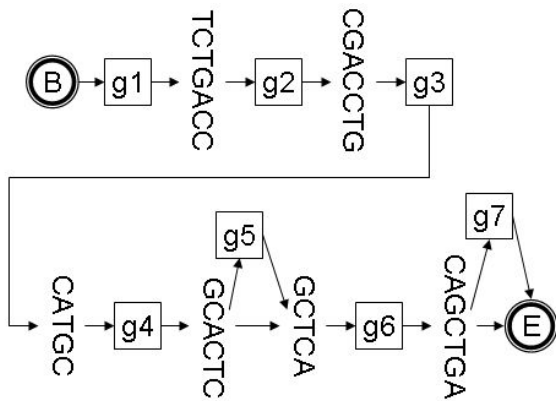


Figure 6: S-HMM generated from the hypoxia dataset exploring a region of 2000 nucleotides preceding the promoter.

Motifs 1, 2, and 4 (referring to the order defined by the graph in Figure 6) are known transcriptional factors. Moreover, the second motif is very similar to a transcriptional factor of ATF3, a gene which is known to be involved in hypoxia (but wasn't in the group used for learning the model). The other motifs are now being investigated by biologists.

Finally, it is worth noticing that the probability computed by the forward-backward algorithm for the Hypoxia model on the sequences of the genes sensible to hypoxia has been found to be 3 order of magnitude higher than on three analogous sequences corresponding to not sensitive genes. This means that no complete instances of the model exist in the second sequence group.

7 Conclusions

A method for automatically synthesizing a Structured HMM for complex and sparse patterns hidden in symbolic sequences has been proposed. The major contributions of the paper are two: the S-HMM itself, which is subclass of Hierarchical HMM [Fine *et al.*, 1998] powerful enough to model many kind of patterns found in real applications, but computationally effective so that it can be applied to very complex data. The most important aspect of S-HMM is its compositional property, which allows a model to be extended incrementally, integrating sub-models possibly obtained from different sources (other learning algorithms or experts of the domain).

The second contribution is EDY's discovery algorithm, which has been validated on a non trivial suite of artificial data and is now applied in a real application of Data Mining on DNA sequences.

References

- [Durbin *et al.*, 1998] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.
- [Fine *et al.*, 1998] S. Fine, Y Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.
- [Fisher, 1987] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.
- [Gussfield, 1997] D. Gussfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.
- [Lee and Stolfo, 1998] W. Lee and S.J Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the Seventh USENIX Security Symposium (SECURITY '98)*, San Antonio, TX, 1998.
- [Lee *et al.*, 2002] W. Lee, w. Fan, M. Miller, S.J. Stolfo, and E. Zadok. Toward cost-sensitive modeling for intrusion detection and response. *Journal of Computer Security*, 10:5–22, 2002.
- [Levenstein, 1966] V.I. Levenstein. Binary codes capable of correcting insertions and reversals. *Soviet. Phys. Dokl.*, 10:707–717, 1966.
- [R C. Roach, 2003] P H. Hacke R C. Roach, P D. Wagner. *Hypoxia: Through the Life Cycle*. Kluwer/Plenum, 2003.
- [Rabiner, 1989] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.