

# Learning Process Behavior with EDY: an Experimental Analysis

Ugo GALASSI<sup>a</sup>

<sup>a</sup> *Dipartimento di Informatica, Università Amedeo Avogadro  
Via Bellini 25G, Alessandria, Italy  
galassi@mfn.unipmn.it*

**Abstract.** This paper presents an extensive evaluation, on artificial datasets, of EDY, an unsupervised algorithm for automatically synthesizing a Structured Hidden Markov Model (S-HMM) from a database of sequences. The goal of EDY is capturing the stochastic process by which the observed data was generated. The S-HMM is a sub-class of Hidden Markov Model that exhibits a quasi-linear computational complexity and is well suited to real-time problems of process/user profiling. The datasets used for the evaluation are available on the web<sup>1</sup>. They are a proposal benchmark for the deep-testing and comparing of tools developed for analysis of temporal (spatial) sequences in which the objective is to reconstruct the generative model from which the sequences originated.

**Keywords.** Hidden Markov Model, Sequence Analysis, Data Mining

## 1. Introduction

Since many years, temporal sequences have been the subject of investigation in many fields, such as signal processing, pattern recognition, and network monitoring. In real world, most systems are dynamic and are naturally described by temporal features, whose values change significantly during an observation period. In the general case, a temporal sequence is the observable manifestation (the trace) of a process, or of a set of processes, which evolve during time. This is, for instance, the case of sequential signals [1,2] coming from sensors or logs of system usage [3,4].

Stochastic *generative models* have been largely proposed as a suitable tool for analyzing temporal sequences. Probability theory offers a framework for modeling the evolution of processes characterized by inherent randomness or operating in environments too complex for a precise analysis. In this framework the statistical distributions governing the evolution of a system can be estimated from a learning set of traces describing its past history. The statistical approach tends to produce a model capturing only the regularities occurring in the sequences (referred to as *motifs*, in the following), while infrequent patterns are considered as noise (we will call them *gaps*). This approach allows to infer a model from a relatively small set of strings.

Hidden Markov Models (HMMs) [5] are a natural choice when the modeling process is discrete and the system depends only on the current state of the system itself. HMMs describe the underlying dynamic processes that govern system behavior and they are perhaps the simplest models of a random evolution without long-term memory.

Despite the advantages of this framework, developing the structure of an HMM is not trivial and most works proposed in the past only deal with the problem of inferring the distribution of probabilities governing that structure. In [6] has been presented EDY, an algorithm which explicitly addresses this problem. EDY is an unsupervised algorithm that models complex profiles from traces by means of Structured Hidden Markov Models (S-HMMs) [7]. A S-HMM benefits from interesting compositional properties, which allow for its incremental construction. It is a variant of Hidden Markov Models which inherits the major advantages of Hierarchical Hidden Markov Models [8] but exhibits a tractable complexity. In [7] has been shown how computing the maximum likelihood path on a sentence 1000 characters long with a model of 800 states will take less than 2 seconds on a Intel CoreDuo @ 2,16Ghz.

Aim of this paper is to provide an extensive evaluation of EDY using artificial traces. In order to understand why we use artificial traces to test EDY performances, we must put in evidence that goal of EDY is to reconstruct the generative model of a process, and not a classification model as it is done by most existing learning algorithms as, for instance, SVMs [9]. In Machine Learning literature a large number of datasets, related to temporal or spatial learning, have been proposed. Many of them are also available on the WEB. Typically they originate from real-world processes, e.g. traces of the activity of a process, DNA sequences, etc. They can be used for classification analysis. But it is difficult to have a precise knowledge of the generative process which originated the sequences, making these datasets unuseful for the kind of structural analysis that is our goal.

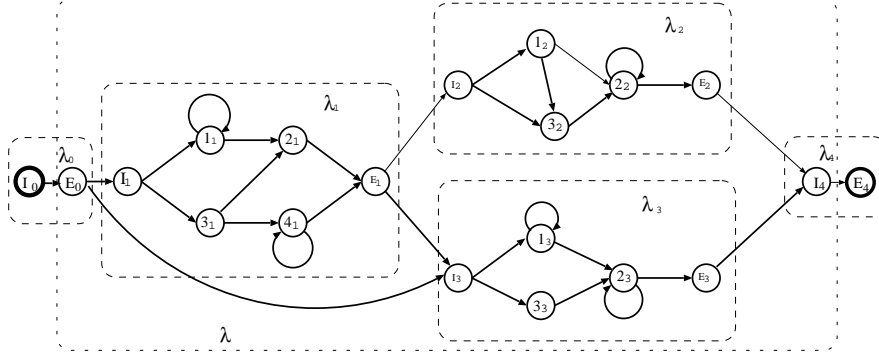
Artificial data are a suitable tool to evaluate learning algorithms, because they can be constructed on purpose to put in evidence both strong and weak points. In order to fully analyze the real potentiality of the algorithm we need to test it on sequences in which the patterns hidden inside are known. We have designed different kind of datasets of growing difficulty, aimed at testing different aspect of the algorithm. These datasets are available, for the Machine Learning community, on <http://www.edygroup.di.unipmn.it>.

In the following sections we will provide a brief overview of S-HMM and EDY algorithm in order to help the reader to better understand the analysis provided in the rest of paper. For a deeper analysis or for an application of the framework to real-world applications, an interested reader can consult [6,7].

## 2. The Structured Hidden Markov Model

The basic assumption underlying an S-HMM is that a sequence  $O = \{O_1, O_2, O_3, \dots, O_t\}$  of observations could be segmented into a set of subsequences  $O^{(1)}, O^{(2)}, \dots, O^{(N)}$ , each one generated by a sub-process with only weak interactions with its neighbors [10]. S-HMMs are represented as directed graphs, structured into sub-graphs (*blocks*), each one modeling a specific kind of sub-sequences. A block consists of a set of states, only two of which (the *initial* and the *end* state) are allowed to be connected to other blocks. As an S-HMM is itself a block, a nesting mechanism is immediate to define.

For providing a formal definition of S-HMM we need to recall that a generic HMM is a stochastic automaton characterized by a set of states  $S$ , an alphabet  $V$ , and a triple  $\lambda = \langle A, B, \Pi \rangle$ , being:



**Figure 1.** Example of Structured Hidden Markov Model composed of three interconnected blocks, plus two null blocks,  $\lambda_0$  and  $\lambda_4$ , providing the start and end states. Distribution  $A$  is non-null only for explicitly represented arcs.

- $A : S \times S \rightarrow [0, 1]$  a probability distribution,  $a_{ij}$ , governing the transition from state  $S_i$  to state  $S_j$ ;
- $B : S \times V \rightarrow [0, 1]$  a probability distribution,  $b_i(V_k)$ , governing the emission of symbols in each state  $S_i \in S$ ;
- $\Pi : S \rightarrow [0, 1]$  a distribution assigning to each state  $S_i \in S$  the probability of being the start state.

A state  $S_i$  will be said a *silent* state if  $\forall V_k \in V : b_i(V_k) = 0$ , i.e.,  $S_i$  does not emit any symbol. When entering a silent state, the time counter must not be incremented.

**Definition 1** A basic block of an S-HMM is a 4-tuple  $\lambda = \langle A, B, I, E \rangle$ , where  $I, E \in Q$  are silent states such that:  $\pi(I) = 1$ ,  $\forall S_i \in S : a_{iI} = 0$ , and  $\forall S_i \in S : a_{Ei} = 0$ .

In other words,  $I$  and  $E$  are the input and the output states, respectively. Therefore, a composite block can be defined by connecting, through a transition network, the input and output states of a set of blocks.

**Definition 2** Given an ordered set of blocks  $\Lambda = \{\lambda_i | 1 \leq i \leq N\}$ , a composite block is a 4-tuple  $\lambda = \langle A_I, A_E, I, E \rangle$ , where:

- $A_I : \mathbf{E} \times \mathbf{I} \rightarrow [0, 1]$ ,  $A_E : \mathbf{I} \times \mathbf{E} \rightarrow [0, 1]$  are probability distributions governing the transitions from the output states  $\mathbf{E}$  to the input states  $\mathbf{I}$ , and from the input states  $\mathbf{I}$  to the output states  $\mathbf{E}$  of the component blocks  $\Lambda$ , respectively.
- For all pairs  $\langle E_i, I_j \rangle$  the transition probability  $a_{E_i I_j} = 0$  if  $j \leq i$ .
- $I \equiv I_1$  and  $E \equiv E_N$  are the input and output states of the composite block, respectively.

According to Definition 2 the components of a composite block can be either basic blocks or, in turn, composite blocks. In other words, composite blocks can be arbitrarily nested. As a special case, a block can degenerate to the *null block*, which consists of the start and end states only, connected by an edge with probability  $a_{IE} = 1$ . An example of S-HMM structured into three blocks  $\lambda_1$ ,  $\lambda_2$ ,  $\lambda_3$ , and two null blocks  $\lambda_0$ ,  $\lambda_4$ , providing the start and the end states, is described in Figure 1.

### 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  $\lambda$  of a sparse pattern is constructed incrementally, starting from a learning set  $LS$  of sequences, going through a learning cycle in which a model is progressively extended and refined, by repeatedly incorporating new motifs and gaps. The rationale behind this architecture is that regularities due to the presence of motifs may be difficult (or impossible) to distinguish from randomness when considered in isolation, but may become evident in the context established by a partial model. Therefore the algorithm tries to discover first the motifs, which are evident in absence of any a priori information. Then using such motifs it builds up a first model which is augmented cycle after cycle by adding new motifs as long as they become detectable.

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. The basic step for extending  $\lambda$  encompasses the following actions:

1. Find a not yet analyzed basic block  $\lambda_k$  encoding a gap.
2. For each sequence  $O \in LS$  determine the most likely subsequence encoded by  $\lambda_k$ , using the viterbi algorithm [5]. If such a subsequence exists, i.e. the path passes through  $\lambda_k$ , insert the observed subsequence into a temporary learning set  $LS_k$ .
3. Search for regularities (motifs) occurring in the sequences accumulated in  $LS_k$ . If no new motifs are discovered, then exit, otherwise build a composite block  $\lambda_k^1$  containing the new motifs and replace  $\lambda_k$  with  $\lambda_k^1$ .
4. Locally train  $\lambda_k^1$  using the Baum-Welch algorithm, and validate the final model according to some test. If the test is successful, then continue, otherwise restore model  $\lambda_k$  and exit.

The above procedure is iterated until no *gap* block remains unexplored. At every step the algorithm can use a set of refinement operators in order to refine *motif* and *gap* blocks. An overview of these operators is presented in [6]

### 4. Artificial Datasets

We test EDY with traces generated by means of *known* S-HMMs constructed by means of a semi-automatic procedure. The task for EDY is then to reconstruct, as closely as possible, the original model starting from the traces. As the target model is a S-HMM, the task should be solvable with very good approximation. Then, the only sources of inaccuracy can be due to weaknesses of the algorithm strategies or to a dataset size insufficient to detect all existing regularities.

Two groups of artificial benchmarks have been constructed: (1) the *sequential* datasets and (2) the *structured* datasets. Each group aims at testing a different aspect of the algorithm. The first group investigates how the behavior of the algorithm is affected by the size of the alphabet encoding the sequences, and by the length of the motifs hidden in the sequences. The second group aims at checking the ability of the algorithm at learning models structured as graphs of motifs, i.e., the ability at learning disjunctive expressions.

**Table 1.** Average value of parameter  $\eta$  and of the sequence length  $l$  in datasets A3, A4, and B3.

	N0		N1		N2		N3	
	$\bar{\eta}$	$\bar{l}$	$\bar{\eta}$	$\bar{l}$	$\bar{\eta}$	$\bar{l}$	$\bar{\eta}$	$\bar{l}$
A3	0.078	755	0.186	414	0.235	388	0.256	385
A4	0.082	1081	0.219	524	0.281	485	0.307	480
B3	0.073	547	0.156	330	0.194	318	0.209	318
C3	0.076	773	0.184	414	0.230	390	0.248	389
D3	0.158	328	0.231	291	0.265	300	0.280	306

#### 4.1. "Sequential" Datasets

This benchmark includes 960 learning problems generated from S-HMMs belonging to 3 different groups (A3, A4 and B3), characterized by a growing complexity from A3 to B3. All S-HMMs have been constructed according to a two level hierarchy. All models of the benchmark generate a chain of motifs separated by gaps of varying length plus an initial and final random gap. The main difference between these models is constituted by the number of motifs composing the chain. Model A3 contains six motifs and model A4 is formed by nine motifs. The last group, B3, also generates a sequence of motifs but it is composed by a chain with forward jumps that allow generating sequences with a varying number of motifs (from two to six).

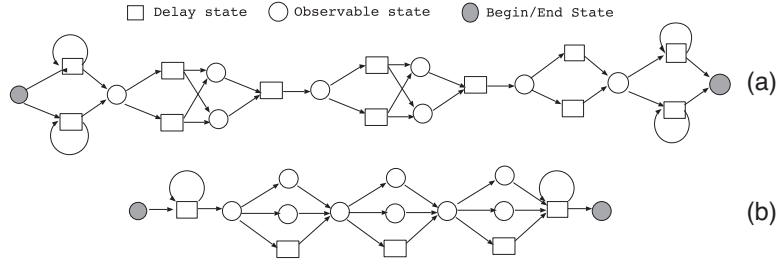
Using a semi-automated procedure, 48 template S-HMMs (16 for each group) have been constructed; they differ in the nominal length of the motifs (5, 8, 11, 15 symbols) and in the cardinality of the alphabet (4, 7, 14, 25 symbols). From each template four different S-HMMs have been obtained, with an increasing level of perturbation on the probability distributions governing the transitions from state to state and the observations generated inside states. Considering the average edit distance  $\delta_E$  between the maximum likelihood sequences generated by the model without perturbations, and the maximum likelihood sequences generated by the perturbed models the following average values have been obtained for the four classes of distributions:

Class: N0 N1 N2 N3  
 $\delta_E$ : 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. For every setting of the above parameters we have generated five different models  $\lambda_n^\Xi$  (where  $n$  ranges from 1 to 5 and  $\Xi$  corresponds to a specific combination of parameters). Finally, for every model  $\lambda_n^\Xi$ , a learning set  $LS_{\lambda_n^\Xi}$  and a test set  $TS_{\lambda_n^\Xi}$ , each containing 100 sequences, have been generated.

The length of each sequence ranges from 800 to 1500 depending on the models. It is worth noticing that, considering the quite short motif length, the coding part is much smaller than the non coding part making the task of discovering it quite difficult. Table 1 reports the average percentage  $\eta$  of the coding part over the total length of the sequences for all the datasets.

Globally 960 different datasets, 320 for each kind of structure have been generated.



**Figure 2.** The structured datasets: (a) template model for C3 datasets is composed by a sequence of constant and alternative motifs, separated by gaps; (b) the template model for D3 datasets is a complex model with alternative motif (that could also be optional), alternated with gaps

#### 4.2. "Structured" Datasets

The procedure used to construct the learning problems of this benchmark is identical to the one used in the previous case. Here, target S-HMMs have a graph like structure at the abstract level. In this way, each S-HMM encodes disjunctive regular expressions.

Two groups of S-HMMs have been defined. The first group, C3, is structured as two crossing chains of motifs separated by gaps. Some motifs always occur in all sequences, whereas others can occur randomly in one or another position, alternatively. The high-level structure of C3 is described in figure 2(a). The second group, D3, is similar to C3, with the difference that motifs may occur consecutively without any separation gap. The template of this model is described in figure 2(b).

According to the procedure described in section 4.1, 32 template S-HMMs (16 for each group) have been generated, which 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 level of noise affecting their emissions (N0, N1, N2, N3).

For every setting of the above parameters five different models  $\lambda_n^\Xi$  have been generated, each one used for generating a learning problem. The length of generated sequences ranges from 800 to 1500 depending on models, and also in this case the portion of string containing motifs is a small fraction of the entire string (see Table 1). A set of 640 different datasets, 320 for each kind of structure has been generated.

### 5. Comparing HMMs

The target for EDY is to reconstruct the original model from a sample of the sequences it generates. Therefore, EDY's performances can be evaluated by comparing the original model to the learned one. We choose to compare HMMs on the base of the sequence distribution they generate. Under this assumption two models can be considered *equivalent* when the corresponding distributions are identical.

Several measures can be found in the literature [5,11], which can be used for this purpose. We selected the one proposed by Rabiner in [5]. Given two models  $\lambda_O$  and  $\lambda_D$  and a string  $s$  generated by model  $\lambda_O$  (denoted with  $s_O$ ), the distance between  $\lambda_D$  and  $\lambda_O$  with respect to  $s_O$  is measured as:

$$d(s_O, \lambda_D, \lambda_O) = \frac{1}{T} [\log(p(s_O|\lambda_D)) - \log(p(s_O|\lambda_O))] \quad (1)$$

being  $T$  the length of string  $s_D$ . We measured the distance between  $\lambda_O$  and  $\lambda_D$  by computing the average value and the standard deviation of the distance measure (1) on all sequences belonging to the distributions to compare. Therefore two models are equivalent when the average distance  $\bar{d}(\lambda_D, \lambda_O) = 0$  and the standard deviation  $\sigma(d(\lambda_D, \lambda_O)) = 0$ .

Distance (1) holds for any kind of HMM and then also for a S-HMM and can be used for assessing the validity of learned models in all the applications which can be set as classification tasks or as prediction tasks. In other kinds of application, the model is used in order to provide an explanation for the patterns observed in a sequence. Usually, this is done by applying Viterbi algorithm in order to find the maximum likelihood path on the model, which can generate the observed string. In this case, the structure of the model becomes really important, because the explanation will be given in terms of a sequence of transitions on the model. Then we introduced a distance measure aimed at testing EDY's ability at discovering the macro structure of the generative model. More specifically, the measure  $\rho(s_O, \lambda_D, \lambda_O)$  accounts for EDY's ability at segmenting a string  $s$  into motifs and gaps corresponding to the ones defined by the original model.

Let  $s_{\lambda_D}, s_{\lambda_O}$  denote the segmentations obtained from  $\lambda_D$  (the discovered model) and  $\lambda_O$  (the original model), respectively;  $s_{\lambda_D}, s_{\lambda_O}$  are aligned by putting into correspondence the pair of segments tagged as motifs, which show the greatest similarity between them. Finally,  $\rho(s_O, \lambda_D, \lambda_O)$  is computed as the ratio:

$$\rho(s_O, \lambda_D, \lambda_O) = \frac{\mathcal{A}(s_{\lambda_D}, s_{\lambda_O})}{\mathcal{L}(s_{\lambda_O})} \quad (2)$$

In (2)  $\mathcal{A}(s_{\lambda_D}, s_{\lambda_O})$  is the sum of the the edit distance computed for all segment pairs, which have been aligned. Possible segments on both sides, which have not found any correspondence are accounted by computing the edit distance from the *null string*. The denominator  $\mathcal{L}(s_{\lambda_O})$ , is the sum of the length of all motifs occurring in the segmentation  $s_{\lambda_O}$ .

A second measure  $\epsilon(s, \lambda_D, \lambda_O)$  has also been introduced, which simply accounts for EDY's ability at correctly distinguishing the meaningful information in the learning set from the non meaningful one inside the gaps. The algorithm for computing  $\epsilon$  is obtained as a simplification of the previous one. Again segmentation  $s_{\lambda_D}, s_{\lambda_O}$  are computed. Then, two substrings  $m_{\lambda_D}, m_{\lambda_O}$  are extracted from  $s_{\lambda_D}, s_{\lambda_O}$ , by collecting all motifs defined by  $\lambda_D, \lambda_O$ , respectively. Finally,  $\epsilon(s_O, \lambda_D, \lambda_O)$  is computed as the ratio:

$$\epsilon(s_O, \lambda_D, \lambda_O) = \frac{\mathcal{D}(m_{\lambda_D}, m_{\lambda_O})}{\mathcal{L}(s_{\lambda_O})} \quad (3)$$

being  $\mathcal{D}(m_{\lambda_D}, m_{\lambda_O})$  the edit distance between  $m_{\lambda_D}$  and  $m_{\lambda_O}$ .

## 6. Discovering Sequential S-HMMs

In this section we report the results obtained running EDY on datasets A3, A4, and B3 we described in Section 4.1. For every learning problem, corresponding to a specific  $\lambda_O$ , a learning set of 100 sequences has been generated. Then measure (1) has been evaluated using 2000 sequences different from the ones in the learning set.

**Table 2.** Values for the distance measure  $\bar{d}(\lambda_O, \lambda_D)$  in dependency of the alphabet cardinality (AC) and the motif length (ML), obtained for datasets A3, A4 and B3

A3	AC				ML			
	4	7	14	25	5	8	11	15
$\bar{d}$	-0.011	-0.023	-0.021	-0.017	-0.012	-0.015	-0.021	-0.025
$\sigma_d$	0.017	0.036	0.042	0.033	0.020	0.026	0.036	0.046
N0								
$\bar{d}$	-0.044	-0.023	-0.028	-0.032	-0.023	-0.032	-0.028	-0.043
$\sigma_d$	0.013	0.026	0.033	0.033	0.019	0.027	0.029	0.030
N1								
$\bar{d}$	-0.050	-0.028	-0.030	-0.025	-0.023	-0.028	-0.037	-0.045
$\sigma_d$	0.014	0.038	0.031	0.023	0.015	0.016	0.030	0.043
N2								
$\bar{d}$	-0.048	-0.022	-1.362	-0.028	-0.022	-0.033	-0.032	-1.373
$\sigma_d$	0.014	0.026	4.829	0.032	0.016	0.024	0.023	4.838
N3								
A4	AC				ML			
	4	7	14	25	5	8	11	15
$\bar{d}$	-0.022	-0.020	-0.019	-0.029	-0.020	-0.018	-0.022	-0.030
$\sigma_d$	0.012	0.026	0.030	0.044	0.016	0.024	0.030	0.043
N0								
$\bar{d}$	-0.066	-0.052	-0.043	-0.051	-0.041	-0.054	-0.054	-0.062
$\sigma_d$	0.012	0.024	0.032	0.053	0.017	0.029	0.031	0.043
N1								
$\bar{d}$	-0.090	-0.046	-0.036	-0.071	-0.032	-0.058	-0.058	-0.095
$\sigma_d$	0.015	0.018	0.039	0.075	0.018	0.022	0.025	0.081
N2								
$\bar{d}$	-0.110	-0.053	-0.079	-0.049	-0.044	-0.055	-0.067	-0.125
$\sigma_d$	0.017	0.016	0.058	0.055	0.019	0.019	0.027	0.082
N3								
B3	AC				ML			
	4	7	14	25	5	8	11	15
$\bar{d}$	-0.019	-0.018	-0.016	-0.019	-0.011	-0.015	-0.021	-0.026
$\sigma_d$	0.014	0.022	0.028	0.034	0.012	0.022	0.029	0.036
N0								
$\bar{d}$	-0.041	-0.026	-0.027	-0.030	-0.021	-0.026	-0.034	-0.043
$\sigma_d$	0.027	0.029	0.030	0.034	0.020	0.028	0.032	0.039
N1								
$\bar{d}$	-0.061	-0.029	-0.033	-0.035	-0.022	-0.033	-0.044	-0.060
$\sigma_d$	0.035	0.033	0.034	0.039	0.021	0.029	0.038	0.053
N2								
$\bar{d}$	-0.056	-0.023	-0.025	-0.192	-0.020	-0.029	-0.036	-0.210
$\sigma_d$	0.037	0.022	0.034	2.273	0.020	0.025	0.033	2.287
N3								

The results obtained for measure  $d$  are reported in table 2. As the results depends on two parameters: the alphabet cardinality, and the motif length, the tables have been compressed by marginalizing on the motif length and on the alphabet cardinality, respectively. In the tables two parameters are reported:  $\bar{d}$  and the standard deviation of  $d(s_O, \lambda_D, \lambda_O)$ . It appears that the probability distribution generated by the discovered model closely resembles to the original one. However, we observe that the value of  $\bar{d}$  is always negative that means that the probability assigned by  $\lambda_D$  is systematically slightly larger than the one assigned by  $\lambda_O$ . This is due to the fact, that  $\lambda_D$  has been learned from a sequence sample quite small comparing to the entire set  $\lambda_O$  can generate. Implicitly, the learning algorithm tends to assign small, or null, probability to the sequences not occurring in the learning set, and, consequently, the other ones will have a probability higher than the original one.

**Table 3.** Values for the distance measures  $\rho$  and  $\epsilon$  in dependency of the alphabet cardinality and motif length, obtained for datasets A3, A4, and B3

		Alphabet Cardinality							
		4		7		14		25	
DS	N	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$
A3	0	0.4053	0.3346	0.1578	0.0776	0.1452	0.0650	0.1391	0.0509
	1	0.5184	0.3899	0.2434	0.0819	0.1472	0.0380	0.0938	0.0186
	2	0.5335	0.3702	0.2014	0.0615	0.1798	0.0397	0.1332	0.0174
	3	0.3593	0.0592	0.0928	0.0502	0.0288	0.0291	0.0244	0.0217
A4	0	0.1017	0.0372	0.1248	0.0632	0.0419	0.0341	0.0503	0.0350
	1	0.5875	0.5147	0.2707	0.1788	0.1363	0.0894	0.1267	0.0561
	2	0.6308	0.5333	0.2954	0.1327	0.1318	0.0426	0.1379	0.0474
	3	0.6566	0.5361	0.3102	0.1299	0.2243	0.1054	0.1434	0.0276
B3	0	0.2460	0.1914	0.2118	0.1253	0.0852	0.0735	0.0737	0.0583
	1	0.6199	0.4835	0.2026	0.1492	0.2113	0.1017	0.1833	0.0784
	2	0.6259	0.5521	0.3175	0.1491	0.2179	0.0812	0.2257	0.0586
	3	0.5397	0.4222	0.3239	0.1205	0.1737	0.0543	0.2153	0.0382

		Motif Length							
		4		7		14		25	
DS	N	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$
A3	0	0.2870	0.2703	0.1951	0.1762	0.0577	0.0449	0.3076	0.0366
	1	0.2466	0.2301	0.1617	0.1505	0.1220	0.0508	0.4724	0.0972
	2	0.2348	0.1839	0.1678	0.1531	0.1558	0.1120	0.4894	0.0398
	3	0.0554	0.0447	0.0480	0.0392	0.1211	0.0415	0.2049	0.0312
A4	0	0.0856	0.0723	0.0609	0.0496	0.0611	0.0277	0.1071	0.0287
	1	0.3528	0.3225	0.2328	0.2165	0.2254	0.1746	0.3101	0.1253
	2	0.2634	0.2275	0.2585	0.2378	0.2661	0.1614	0.4079	0.1293
	3	0.3182	0.2762	0.1963	0.1659	0.2649	0.1662	0.5552	0.1907
B3	0	0.1127	0.1023	0.1486	0.1370	0.1721	0.1168	0.1500	0.0701
	1	0.3715	0.3158	0.2138	0.1817	0.2133	0.1757	0.4185	0.1396
	2	0.3104	0.2614	0.2388	0.2185	0.3236	0.1948	0.5143	0.1663
	3	0.1824	0.1276	0.2006	0.1580	0.2637	0.1538	0.5166	0.1222

The results of the analysis made with respect to measure  $\rho$  and  $\epsilon$  are reported in Table 3. For the estimate of measure  $\rho$ , and  $\epsilon$  a test set of 100 sequences, has shown to be enough. Also in this case marginalization has been applied on the complementary parameter, in order to have a more compact representation.

As one may expect, the algorithm always finds an error-free model when motifs are not affected by noise (gaps are always filled with random noise). In presence of noise, it appear that both  $\rho(\lambda_O, \lambda_D)$  and  $\epsilon(\lambda_O, \lambda_D)$  increase when the alphabet cardinality and the motif length decrease. 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, in some cases, the accuracy decreases again when motifs become

**Table 4.** Values for the distance measure  $\bar{d}(\lambda_O, \lambda_D)$  in dependency of the alphabet cardinality (AC) and the motif length (ML), obtained for datasets C3 and D3

C3		AC				ML			
		4	7	14	25	5	8	11	15
N0	$\bar{d}$	-0.018	-0.024	-0.028	-0.028	-0.014	-0.024	-0.027	-0.032
	$\sigma_d$	0.015	0.030	0.045	0.052	0.022	0.033	0.039	0.049
N1	$\bar{d}$	-0.049	-0.035	-0.033	-0.043	-0.030	-0.033	-0.044	-0.053
	$\sigma_d$	0.016	0.032	0.039	0.048	0.020	0.034	0.038	0.042
N2	$\bar{d}$	-0.044	-0.036	-0.037	-0.040	-0.031	-0.032	-0.044	-0.049
	$\sigma_d$	0.015	0.038	0.035	0.045	0.021	0.025	0.036	0.050
N3	$\bar{d}$	-0.067	-0.036	-0.039	-0.036	-0.039	-0.033	-0.037	-0.069
	$\sigma_d$	0.017	0.031	0.046	0.029	0.024	0.026	0.039	0.035
D3		AC				ML			
		4	7	14	25	5	8	11	15
N0	$\bar{d}$	-0.067	-0.076	-0.074	-0.086	-0.051	-0.070	-0.081	-0.102
	$\sigma_d$	0.029	0.048	0.087	0.116	0.043	0.060	0.078	0.097
N1	$\bar{d}$	-0.084	-0.088	-0.093	-0.083	-0.064	-0.093	-0.082	-0.108
	$\sigma_d$	0.027	0.052	0.051	0.063	0.032	0.045	0.042	0.073
N2	$\bar{d}$	-0.080	-0.073	-0.085	-0.081	-0.073	-0.073	-0.092	-0.081
	$\sigma_d$	0.025	0.038	0.045	0.052	0.035	0.034	0.043	0.049
N3	$\bar{d}$	-0.088	-0.086	-0.094	-0.107	-0.076	-0.084	-0.098	-0.117
	$\sigma_d$	0.028	0.050	0.051	0.061	0.039	0.041	0.054	0.056

longer than 11 symbols. An explanation for this behaviour will be discussed in section 8. Nevertheless, it is interesting that the error rate remains comparable to the level of noise of the dataset. 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.

## 7. Discovering graph structured patterns

Aims of this case study is to check the ability of the algorithm at reconstructing patterns described by disjunctive expressions. We used the *structured* datasets (C3 and D3) in order to perform this analysis. As discussed in section 4.2 this group of datasets is very similar to the *sequential* datasets but characterized by a more complex graph structure.

The results are described by means of a set of tables, reporting the values obtained for measure  $d$  (Table 4), and for measures  $\rho$  and  $\epsilon$  (Table 5).

Also in this case, we obtain performances, which, even if worst that in the previous one, are good for what concerns  $\bar{d}$  and  $\epsilon$  measures. Instead, parameter  $\rho$  shows that the learned structure tends to be significantly different from the original one.

It is important to evidence how the performances improve when the cardinality of alphabet grow up. As stated in previous section when the cardinality of alphabet grows apparent regularities due to randomness are more rare. The accuracy in some cases, decreases again when motif length increases. This phenomenon is particularly evident when

**Table 5.** Values for the distance measures  $\rho$  and  $\epsilon$  in dependency of the alphabet cardinality and motif length, obtained for datasets C3 and D3

		Alphabet Cardinality							
		4		7		14		25	
DS	N	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$
C3	0	0.3694	0.2608	0.1923	0.1168	0.1573	0.0768	0.1510	0.0711
	1	0.6157	0.3738	0.2542	0.1060	0.2303	0.0663	0.1447	0.0400
	2	0.5626	0.2794	0.3698	0.1069	0.2026	0.0469	0.1812	0.0482
	3	0.4471	0.2737	0.1125	0.0596	0.0483	0.0303	0.0322	0.0193
D3	0	0.6692	0.2434	0.4604	0.1832	0.2300	0.0888	0.2118	0.0779
	1	0.7789	0.3645	0.5491	0.2252	0.3980	0.1446	0.3102	0.0925
	2	0.6618	0.3236	0.5388	0.1403	0.4034	0.1164	0.3653	0.0774
	3	0.7231	0.3625	0.5370	0.1775	0.3789	0.1266	0.4161	0.1072

		Motif Length							
		4		7		14		25	
DS	N	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$	$\rho$	$\epsilon$
C3	0	0.3416	0.1995	0.1549	0.1579	0.1192	0.1117	0.2543	0.1242
	1	0.3209	0.2562	0.1506	0.2416	0.2687	0.1926	0.5047	0.1364
	2	0.3280	0.2513	0.1545	0.1624	0.2978	0.14979	0.5360	0.0943
	3	0.0658	0.0549	0.0703	0.0601	0.1572	0.0421	0.2516	0.1711
D3	0	0.4678	0.1995	0.4065	0.1579	0.3287	0.1117	0.3685	0.1242
	1	0.6045	0.2562	0.3852	0.2416	0.3852	0.1926	0.5079	0.1364
	2	0.5907	0.2513	0.4324	0.1624	0.406	0.1497	0.5401	0.0943
	3	0.6031	0.2564	0.3882	0.2013	0.4239	0.1596	0.6399	0.1565

motifs are characterized by a low alphabet cardinality and a high noise level. We observed this behavior also in the test reported in the previous section and we will provide an explanation in the final discussion in section 8.

Despite the complexity of the task the algorithm reached systematically performance comparable to the level of noise of the dataset and the system always converged to a stable model in a number of steps not greater than 40. Finally the average time for performing each one of those task was about 1 hour on an Opteron.

## 8. Conclusions

The analysis reported in this paper investigates the capabilities of EDY of reconstructing a generative model starting from a sample of sequences it generated.

The first feature emerging from the tables reported in the previous sections, is the strong influence of the alphabet cardinality on the algorithm performances. With small alphabets, the performances remarkably decay. Moreover, this effect is amplified by the presence of noise. The phenomenon depends on the probability of finding apparent regularities, due to randomness, which increases as long as the alphabet becomes small. Consequently, on the one hand, apparent motifs are confused with the real ones making harder the task for the discovery procedure. On the other hand, the statistical test we use

for selecting the motifs to include in the model, tends to reject also good motifs, because they have a high probability of occurring also in a random sequence.

The second feature, which emerges, is the divergence between  $\rho$  and  $\epsilon$  in many learning problems, which appears to depend upon both the motif length and the alphabet cardinality. The cause of this divergence is due to the fact that some of the original motifs are segmented into several smaller ones.

If motifs are short, this fragmentation happens when the cardinality of the alphabet is small. True motifs can be confused with false motifs and the discovery procedure can distinguish only one part of a motif, i.e. the one which by chance appears to be more stable in the learning set. However, a similar problem it may happen also with alphabet of large cardinality when the motifs are very long and highly perturbed. When a long motif is corrupted in the central part it may be split into two or more fragments that the refinement procedures are no more able to recover. In both cases, it may happens that in a later step, searching inside a gap the missed part is discovered and included in the model as an independent motif. The problem of fragmentation of motifs could be solved using a algorithm for merging contiguous motifs. Unfortunately, at the moment such an algorithm has not yet been implemented.

In conclusion, the motif fragmentation effect let  $\rho$  and  $\epsilon$  exhibit quite different values in several learning problems. In these cases most relevant information has been discovered, while the macro-structures of the original S-HMMs have not been preserved with good accuracy. Then, the discovered models are suitable for classification or for prediction tasks (as confirmed by  $\bar{d}$  distance measure), but they are not suitable for interpretation (segmentation) tasks.

Despite the mentioned drawbacks EDY demonstrated to be quite powerful. It is able to perform fast analysis on huge datasets and can operate totally unsupervised being of great help in such tasks in which there is not a background knowledge of the domain.

## References

- [1] K. S. Fu. *Syntactic pattern recognition and applications*. Prentice Hall, 1982.
- [2] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, November 1995.
- [3] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval Journal*, 3:127–163, 2000.
- [4] 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.
- [5] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.
- [6] Ugo Galassi, Attilio Giordana, and Lorenza Saitta. Incremental construction of structured hidden markov models. In Manuela M. Veloso, editor, *IJCAI*, pages 798–803, 2007.
- [7] Ugo Galassi, Attilio Giordana, and Lorenza Saitta. Structured hidden markov models: A general tool for modeling agent behaviors. In *Soft Computing Applications in Business*, number 230 in Studies in Fuzziness and Soft Computing, pages 273–292. Springer, 2008.
- [8] S. Fine, Y Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.
- [9] B. Schölkopf, C. Burgess, and A. Smola. *Advances in Kernel Methods*. MIP Press, 1998.
- [10] D. Bouchaffra and J. Tan. Structural hidden markov models using a relation of equivalence: Application to automotive designs. *Data Mining and Knowledge Discovery*, 12:79 – 96, 2006.
- [11] Lillian Lee. Measures of distributional similarity. In *ACL*, 1999.