

Learning Profiles based on Hierarchical Hidden Markov Model

Ugo Galassi¹, Attilio Giordana¹, Lorenza Saitta¹, and Maco Botta²

¹Dipartimento di Informatica, Università Amedeo Avogadro
Spalto Marengo 33, Alessandria, Italy

²Dipartimento di Informatica, Università di Torino,
C.so Svizzera 185, 10149 Torino, Italy

Abstract. This paper presents a method for automatically constructing a sophisticated user/process profile from traces of user/process behavior. User profile is encoded by means of a Hierarchical Hidden Markov Model (HHMM). The HHMM is a well formalized tool suitable to model complex patterns in long temporal or spatial sequences. The method described here is based on a recent algorithm, which is able to synthesize the HHMM structure from a set of logs of the user activity. The algorithm follows a bottom-up strategy, in which elementary facts in the sequences (motives) are progressively grouped, thus building the abstraction hierarchy of a HHMM, layer after layer. The induction strategy has been designed in order to deal with events characterized by a sparse structure, where gaps filled by irrelevant facts can be intermixed with the relevant ones. The method is firstly evaluated on artificial data. Then a user identification task, from real traces, is considered. A preliminary experimentation with several different users produced encouraging results.

1 Introduction

Building profiles for processes and for interactive users, is an important task in intrusion detection. This paper presents the results obtained with a recent induction algorithm [2], which is based on Hierarchical Hidden Markov Model [5]. The algorithm discovers typical "motives"¹ of a process behavior, and correlates them into a hierarchical model. Motives can be interleaved with possibly long gaps where no regular behavior is detectable. We assume that motives could be affected by noise due to non-deterministic causes. Noise is modeled as insertion, deletion and substitution errors according to a common practice followed in Pattern Recognition. An approach to deal with such kind of noisy patterns, which reported impressive records of successes in speech recognition [10] and DNA analysis [4], is the one based on Hidden Markov Model (HMM) [11]). However, applying HMM does not reduce to simply running a learning algorithm but it requires a considerable effort to individuate a suitable structure for the HMM. A formal framework to design and train complex HMMs

¹ A motif is a subsequence of consecutive elementary events typical of a process.

is represented by the Hierarchical Hidden Markov Model (HHMM) [5]. The problem of estimation HMM and HHMM parameters has been widely investigated while little has been done in order to learn their structure. A few proposals can be found in the literature in order to learn the structure of HMM. A novelty, in this sense, is represented by a recent paper by Botta et Al. [2], which proposes a method for automatically inferring from sequences, and possibly domain knowledge, both the structure and the parameters of complex HHMMs.

In this paper, the learning algorithm proposed in [2] is briefly overviewed and then is experimentally evaluated on three *profiling* case studies. The first two cases are built on a suite of artificial traces automatically generated by a set of given HHMMs. The challenge for the algorithm is to reconstruct the original model from the traces. It will be shown that the algorithm is able to learn HHMMs very similar to the original ones, in presence of noise and distractors.

The third case study refers to the problem of constructing a discriminative model for a user typing on a keyboard [1, 3, 8]. The results reported with a set of 20 different users are encouraging.

2 The Hierarchical Hidden Markov Model

A Hierarchical Hidden Markov Model is a generalization of the Hidden Markov Model, which is a stochastic finite state automaton [11] defined by a tuple $\langle S, O, A, B, \pi \rangle$, where:

- S is a set of states, and O is a set of atomic events (observations),
- A is a probability distribution governing the transitions from one state to another. Specifically, any member $a_{i,j}$ of A defines the probability of the transition from state s_i to state s_j , given s_i .
- B is a probability distribution governing the emission of observable events depending on the state. Specifically, an item $b_{i,j}$ belonging to B defines the probability of producing event O_j when the automaton is in state s_i .
- π is a distribution on S defining, for every $q_i \in S$, the probability that s_i is the initial state of the automaton.

A difficulty, related to a HMM defined in this way, is that, when the set of states S grows large, the number of parameters to estimate (A and B) rapidly becomes intractable.

A second difficulty is that the probability of a sequence being generated by a given HMM decreases exponentially with its length. Then, complex and sparse events become difficult to discover.

The HHMM proposed by Fine, Singer and Tishby [5] is an answer to both problems. On one hand, the number of parameters to estimate is strongly reduced by assigning a null probability to many transitions in distribution A , and to many observations in distribution B . On the other hand, it allows a possibly long chain of elementary events to be abstracted into a single event, which can be handled as a single item. This is obtained by exploiting the regular languages property

of being closed under substitution, which allows a large finite state automaton to be transformed into a hierarchy of simpler ones.

More specifically, numbering the hierarchy levels with ordinals increasing from the highest towards the lowest level, observations generated in a state s_i^k by a stochastic automaton at level k are sequences generated by an automaton at level $k + 1$. Moreover, no direct transition may occur between the states of different automata in the hierarchy. An example of HHMM is given in Figure 1.

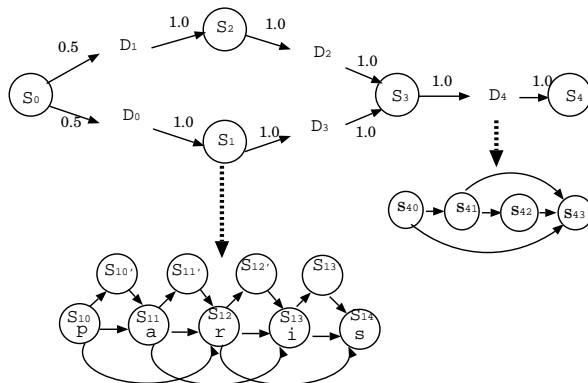


Fig. 1. Example of Hierarchical Hidden Markov Model. Circles denotes states with observable emission, whereas rectangles denote *gaps*.

The advantage of the hierarchical structure, as defined by [5], may help very much the inference of the entire structure of the automaton by part of an induction algorithm.

The research efforts about HHMM mostly concentrate on the algorithms for estimating the probabilities governing the emissions and the transition from state to state. In the seminal paper by Fine et al. [5], the classical Baum-Welch algorithm is extended to the HHMM. In a more recent work, Murphy and Paskin [9] derive a linear (approximated) algorithm by mapping a HHMM into a Dynamic Bayesian Network.

3 Learning Algorithm Overview

The basic algorithm [2] is bottom-up and constructs the HHMM hierarchy starting from the lowest level. The first step consists in searching for possible motives, i.e., short chains of consecutive symbols that appear frequently in the learning traces, and building a HMM for each one of them. This step is accomplished by means of classical methods used in DNA analysis [4, 7]. As, motif models are constructed independently one from another, it may happen that models for spurious motives be constructed. At the same time, it may happen that relevant motives be disregarded just because their frequency is not high enough. Both

kinds of errors will be fixed at a second time. The HMMs learned so far, are then used as feature constructors. Each HMM is labeled with a different *name* and the original sequences are rewritten into the new alphabet defined by the set of names given to the models. In other words, every sub-sequence in the input sequences, which can be attributed to a specific HMM, is replaced by the corresponding name.

The subsequences between two motives, not attributed to any model, are considered gaps and will be handled by means of special construct called *gap*. We will call *sequence abstraction* this rewriting process. After this basic cycle has been completed, an analogous learning procedure is repeated on the abstracted sequences. Models are now built for sequences of *episodes*, searching for long range regularities among co-occurrent motives. In this process, spurious motives not showing significant regularities can be discarded. The major difference, with respect to the first learning step, is that the models built from the abstract sequences, are now observable markov models. This makes the task easier and decreases the computational complexity. In this step, models (*gaps*) are built also for the long intervals falling between consecutive motives.

In principle, the abstraction step could be repeated again on the sequences obtained from the first abstraction step, building a third level of the hierarchy, and so on. However, up to now, we considered only problems where two hierarchical levels are sufficient.

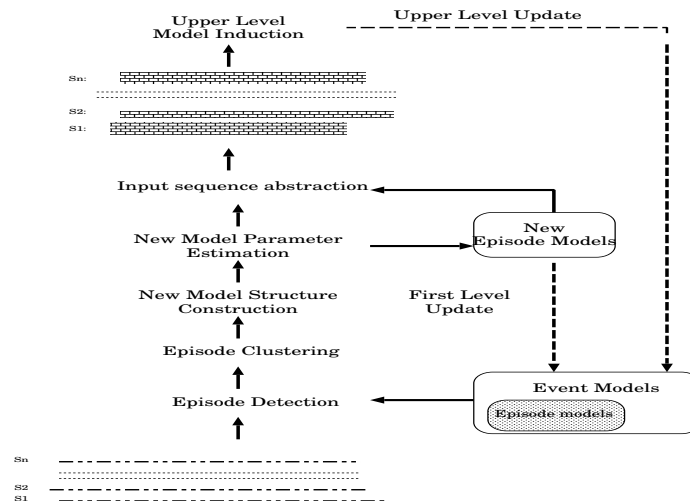


Fig. 2. Learning cycle

After building the HHMM structure in this way, it can be refined using standard training algorithms like the ones proposed in [5, 9]. However, two other refinement techniques are possible.

The first technique concerns the recovery of motives that have been lost in the primary learning phase because they did not have a sufficient statistical evidence. As said above, this missed information has actually been modeled by *gaps*. A nice property of the HHMM is that sub-models in the hierarchy have a loose interaction with one another, and so their structure can be reshaped without destroying the global structure. This means that the model of a gap can be transformed into the model of a motif later on, when further data will be available. This is actually done on demand: all sub-sequences attributed to a given *gap* are collected, creating a new learning set where the learning process is repeated. If there is now evidence for a motif, a model is built up and replaced to the *gap*.

The second method consists in repeating the entire learning cycle using as learning set only the portion of the sequences where the instance of the previously learned HHMM have been found with sufficient evidence. Repeating the procedure allows more precise models to be learned for motives, because false motives will no longer participate to the learning procedure.

Figure 3 provides a scheme of the algorithm architecture. The details about the implementation can be found in [2].

4 Evaluation on Artificial Traces

A specific testing procedure has been designed in order to monitor the capability of the algorithm of discovering "known patterns" hidden in trace artificially generated by a handcrafted HHMMs. Random noise and spurious motives have been added to all sequences filling the gaps between consecutive motives, in order to make the task more difficult.

Three target HHMMs, each one constructed according to a two level hierarchy have been used to generate a set of 72 learning tasks (24 for every model). Every learning task consists of a set of 330 traces. The 90% of the sequences contain an instance of a target HHMM that should be discovered by the learning program, whereas the 10% contain sequences of spurious motives non generated by the target HHMM. The sequence length ranges from 80 to 120 elementary events.

The structure for the high level of the three models is shown in Figure 3. Every state at the high level emits a string (motif) generated by an HMM at the low level, indicated with a capital letter (A,B,C,D,E). A different HMM (F) has been used to generate spurious motives. The gaps between motives have been filled with subsequences containing random noise.

The evaluation of the obtained results has been done on the base of the *bayes classification error* between two (or more HHMMs). Formally, given two HHMMs, λ_1 and λ_2 , and the set L of all possible traces, which can be generated by λ_1 or λ_2 , the Bayes classification error $C(\lambda_1, \lambda_2)$ is defined as:

$$C(\lambda_1, \lambda_2) = \sum_{x \in L} [\min(p(\lambda_1|x), p(\lambda_2|x))]p(x) \quad (1)$$

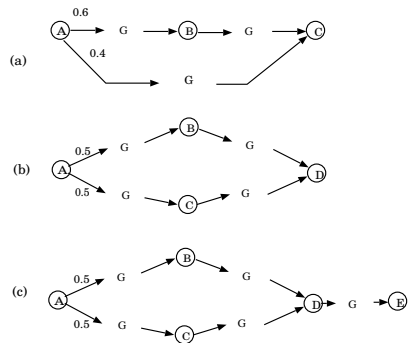


Fig. 3. HHMM used for evaluation on artificial data

being $p(\lambda_1|x)$ and $p(\lambda_2|x)$ the probability that, given a trace x , it has been generated by λ_1 or λ_2 , respectively, and $p(x)$ the a priori probability of x . We notice that the upper-bound for $C(\lambda_1, \lambda_2)$ is 0.5, when λ_1 and λ_2 are identical. In general, for N models, the upper-bound is given by the expression $1 - 1/N$.

In general, expression (1) cannot be computed because L is too large. Therefore, we adopted an approximate evaluation made using a subset of L stochastically sampled.

The bayes classification error ϵ intervenes in the evaluation procedure in two different ways. A first way is to measure the quality of the learned models. A perfect learner should learn a model identical to the one used to generate the traces. Therefore, a learned model has to be considered as much accurate as much close to 0.5 the classification error, between it and the original model, is.

The second way is to estimate the difficulty of the learning task. It is reasonable to assume that the difficulty of identifying a model hidden in a set of traces grows along with the similarity among the motives belonging to the model and the spurious motives. Moreover, the difficulty grows also when the motives belonging to a same model become similar each other, because it becomes more difficult to discover the correspondence between a motif and the hidden state it has been emitted from. Therefore, the experimentation has been run using different versions of models A, B, C, D, E, F with different bayes classification error among them.

The results obtained under three different conditions of difficulty are summarized in Table 1. The similarity between the six kinds of motives has been varied from 0.2 to 0.55. For every setting, the experiment has been repeated 8 times for each one of the three models. The reported results are the average over the 8 runs. In all cases, the bayes classification error has been estimated using a set of traces obtained by collecting 500 sequences generated from each one of the models involved in the specific comparison.

Table 1. Bayes classification error between the target model and the learned model, versus the confusion among the basic motives (reported in the first line).

Motives	0.2	0.4	0.55
Model (a)	0.48	0.46	0.45
Model (b)	0.47	0.42	0.42
Model (c)	0.43	0.42	0.41

It appears that the performances suffer very little from the similarity among the motif models, and in all cases, the similarity between the original model and the learned model is very high ($C(\lambda_1, \lambda_2)$, is close to 0.5).

5 Evaluation of the generalization capabilities

The second case study has the goal of evaluating the ability of the algorithm at correctly generalizing the nominal form of motives in presence of noise. The generalization of the learned HHMM is assessed by considering the maximum likelihood sequence, it generates. In the best case this should be identical to the one generated by the original model, used to construct the dataset. For this group of experiments, HHMMs, which generate sequences of names of towns in a predefined order, have been used. Such HHMMs also model the presence of noise in the data, in form of insertion, deletion and substitution errors. The gaps between the names are filled by symbols randomly chosen in the alphabet defined by the union of the letters contained in the names. Moreover, random subsequences, up to 15 characters long, have been added at the beginning and the end of each sequence. The global length of the sequences ranges from 60 to 120 characters. The difficulty of the task has been controlled by varying the degree of noise.

One set of experiments has been designed in this framework. More specifically, a sequence of problems has been generated varying the number of words ($5 \leq w \leq 8$), the word length ($5 \leq L \leq 8$) and the noise level ($N \in \{0\%, 5\%, 10\%, 15\%\}$). For every triple $\langle w, L, N \rangle$, 10 different datasets has been generated for a total of 640 learning problems.

The most important results are summarized in Table 2. The error rate is evaluated as the edit distance (i.e. the minimum number of corrections) between the maximum likelihood sequence (maximum consensus) generated by the Viterbi algorithm [6] from the original HHMM and the one generated from the learned HHMM. When, an entire word is missed, the corresponding error is set equal to the its length. Experiments in table 2, reporting an error rate much higher than the others, have missed words. In all cases, the learning cycle has been iterated twice, as explained in Section 3. It appears that the average error rate after one refinement cycle decreases of about 50% with respect to the first learning step.

From Table 2, it appears that the model extracted from the data without noise is almost error free. Moreover, the method seems to be little sensitive with

Table 2. Performances obtained with *town names* dataset. The sequence length ranges from 60 to 140 characters. The CPU time, for solving a problem, ranges from 42 to 83 seconds on a Pentium IV 2.4Ghz.

		After the first cycle				After refinement			
w	L	Noise Level				Noise Level			
		0%	5%	10 %	15%	0%	5%	10 %	15%
5	5	0.03	0.06	0.06	0.08	0.04	0.04	0.04	0.04
5	6	0.06	0.12	0.12	0.09	0.03	0.03	0.03	0.03
5	7	0.00	0.02	0.03	0.05	0.00	0.00	0.02	0.00
5	8	0.02	0.04	0.02	0.04	0.00	0.00	0.00	0.00
6	5	0.06	0.11	0.04	0.04	0.10	0.06	0.00	0.03
6	6	0.06	0.10	0.06	0.19	0.05	0.00	0.00	0.00
6	7	0.03	0.03	0.02	0.05	0.02	0.00	0.00	0.00
6	8	0.01	0.04	0.05	0.05	0.00	0.00	0.04	0.00
7	5	0.02	0.05	0.11	0.17	0.02	0.05	0.01	0.10
7	6	0.01	0.10	0.05	0.14	0.04	0.02	0.05	0.04
7	7	0.00	0.06	0.02	0.05	0.00	0.00	0.02	0.05
7	8	0.01	0.06	0.09	0.11	0.01	0.00	0.09	0.09
8	5	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00
8	6	0.03	0.08	0.10	0.14	0.03	0.06	0.06	0.14
8	7	0.00	0.01	0.01	0.08	0.00	0.00	0.00	0.00
8	8	0.01	0.03	0.08	0.09	0.01	0.00	0.00	0.0

respect to the sequence length while the error rate roughly increases proportionally to the noise in the original model (the 15% of noise corresponds to an average error rate of about 19%).

6 User Identification

The task consists in learning to identify a user from the its typing style on a keyboard. The basic assumption is that every user has a different way of typing that becomes particularly evident when he types words which are specifically important for him, such its name or words referring to his job. Assimilating such words to motives characteristic of a user, it is possible to learn a discriminating HHMM on the basis of them.

Two experiments have been done. In the first, a group of 20 users have been asked to type a sentence of 21 syllables on the same keyboard. A transparent program recorded, for every typed key, the duration and the delay two consecutive strikes creating a trace for every typing session. Each user provided ten repetitions of the sentence. Then, a dataset of 200 traces has been obtained, which has been partitioned into a learning set of 140 traces and a testing set of 60 traces. According to a standard procedure in machine learning, 20 HHMMs have been learned from the learning set. Then, the 20 HHMMs have been used to classify the traces in the testing set, according to the following standard procedure. For each HHMM M and for each trace t the forward-backward algorithm

[11] is applied in order to estimate the probability for t of being generated by M . Then, t is attributed to the HHMM that has shows the highest probability. If such a HHMM is the model of the the user that has generated t , the classification is considered correct. Otherwise it is counted as a misclassification error. However, it may happen that all HHMMs show a null probability when t does not belong the language of anyone of them. This is considered a rejection error. In, the specific case, 76% of the traces have been correctly classified with a very good margin (a strong difference between the probability assigned by the correct model and the other ones). An analysis of the misclassified data has shown that they are by to the presence of serious typing errors (it is quite rare that a sequence in the dataset does not show any typing correction). As the number of repetitions for a single user is small, the algorithm was not able to learn also the error modalities for each one of them.

In the second experiment, a volunteer (not belonging to the group of 20 users in the first experiment) provided 140 repetitions of the same sentence used in the first experiment. A set of 100 of them has been used to build a model of the user. The HHMM learned by the algorithm contained 11 motif models at the low level corresponding to 11 states at the high level. It has been evaluated using the remaining 40 traces of the volunteer and the 200 traces used in the first experiment. The results have been excellent: on the 40 traces belonging to the volunteer, the log-odds of the probability of the trace assigned by the model was always very high (from +140 to +190), whereas on the 200 traces of the other users it was always very low (from -10 to +10).

The experiment shows that, increasing the size of the dataset, the algorithm has been able to learn a very accurate model.

7 Conclusions

We have proposed a method for automatically synthesizing complex profiles based on HHMMs from traces. In preliminary tests on artificial datasets, the method succeeded in reconstructing non trivial two level HHMMs, whereas the results obtained on a task of user identification are very encouraging. It is worth noticing that, in this case, the goal was not to compete with the results obtained by task specific algorithms[1, 3, 8], but to test how a general purpose algorithm performed on a non trivial task for which it was not customized. In fact, the learning algorithm has been simply run on the datasets dedicating few hours to prepare the experiment. Even if the experimentation is not extensive enough for a definitive conclusion, the results look interesting. In particular, we consider very promising the fact that the distance between the models of the different users is very large.

8 Acknowledgments

The present work has been supported by the FIRB Project: WebMinds

References

1. S. Bleha, C. Slivinsky, and B. Hussein. Computer-access security systems using keystroke dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-12(12):1217–1222, 1990.
2. M. Botta, U. Galassi, and A. Giordana. Learning complex and sparse events in long sequences. In *Proceedings of the European Conference on Artificial Intelligence, ECAI-04*, Valencia, Spain, August 2004.
3. M. Brown and S.J. Rogers. User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies*, 39:999–1014, 1993.
4. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998.
5. S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32:41–62, 1998.
6. G. D. Forney. The viterbi algorithm. *Proceedings of IEEE*, 61:268–278, 1973.
7. D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.
8. R. Joyce and G. Gupta. User authorization based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990.
9. K. Murphy and M. Paskin. Linear time inference in hierarchical hmms. In *Advances in Neural Information Processing Systems (NIPS-01)*, volume 14, 2001.
10. L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NY, 1993.
11. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.