

# Experimental Evaluation of Hierarchical Hidden Markov Models

Attilio Giordana, Ugo Galassi, and Lorenza Saitta

Dipartimento di Informatica, Università del Piemonte Orientale  
Via Bellini 25g, Alessandria, Italy

**Abstract.** Building profiles for processes and for interactive users is a important task in intrusion detection. This paper presents the results obtained with a Hierarchical Hidden Markov Model. The algorithm discovers typical "motives" 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, modeled as insertion, deletion and substitution errors. In this paper the learning algorithm is briefly recalled and then it is experimentally evaluated on three *profiling* case studies. The first case is 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 second and third case studies refer to the problem of constructing a discriminant model for a user typing on a keyboard.

## 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"<sup>1</sup> 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. A recent paper by Botta et al. [2] 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 it is experimentally evaluated on three *profiling* case studies. The first case is 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.

---

<sup>1</sup> A motif is a subsequence of consecutive elementary events typical of a process.

The second and third case study refers to the problem of constructing a discriminant model for a user typing on a keyboard [1, 3, 7]. The results reported with a set of 20 different users are very 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 [9] 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  $s_i$ .
- $\pi$  is a distribution on  $S$  defining, for every  $s_i \in S$ , 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 et al. [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, the model 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.

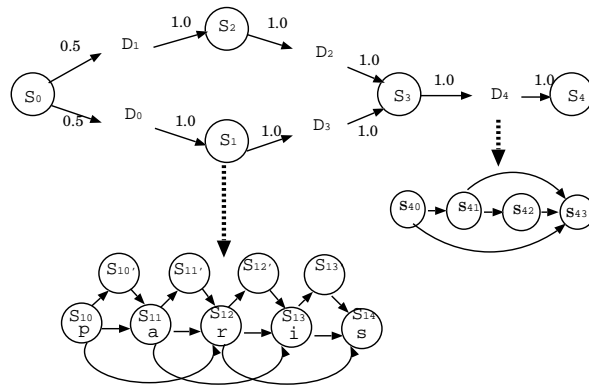
An example of HHMM is given in Figure 1.

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

The research efforts about HHMMs mostly concentrate on the algorithms for estimating the probabilities governing the emission and the transition from state to state. Fine et al. [5] extend the classical Baum-Welch algorithm to the HHMM. In a more recent work, Murphy and Paskin [8] derive a linear (approximated) algorithm by mapping a HHMM into a Dynamic Bayesian Network.

## 3 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,



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

i.e., short chains of consecutive symbols that appear frequently in the learning traces, and building an HMM for each one of them. This step is accomplished by means of classical methods used in DNA analysis [4, 6]. As models of the motives are constructed independently from one another, it may happen that models for spurious motives are constructed. At the same time, it may happen that relevant motives are 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 learning task easier and decreases its 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. After building the HHMM structure in this way, it can be refined using standard training algorithms [5, 8]. However, two other refinement techniques are possible.

The first technique concerns the recovery of motives lost in the first learning phase because not having 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 changing the global structure. Then, the model of a gap can be transformed into the model of a motif later on, when further data become available.

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. 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 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.

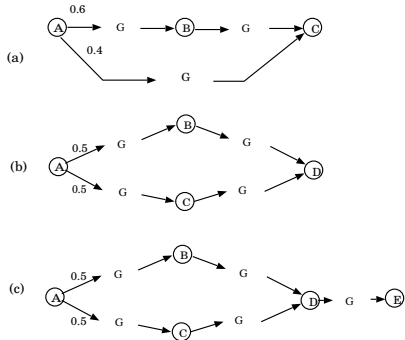
Three target HHMMs, each one constructed according to a two level hierarchy, have been used to generate a set of 72 learning tasks (24 for every model). Every learning task uses a set of 330 traces. 90% of the sequences contain an instance of a target HHMM that should be discovered by the learning program, whereas the 10% of the sequences contain 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 2. 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 basis of the *Bayes classification error* between two (or more) HHMMs. Formally, given two HHMMs,  $\lambda_1$  and  $\lambda_2$ , and the set  $L$  of all possible traces that can be generated by  $\lambda_1$  or  $\lambda_2$ , the Bayes classification error  $C(\lambda_1, \lambda_2)$  is defined as:

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

being  $p(\lambda_1|x)$  and  $p(\lambda_2|x)$  the probability that, given a trace  $x$ , it has been generated by  $\lambda_1$  or  $\lambda_2$ , 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  $\lambda_1$  and  $\lambda_2$  are identical. In general, for  $N$  models, the upper-bound is given by  $(1 - 1/N)$ .



**Fig. 2. HHMM used for evaluation on artificial data**

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 role of Bayes classification error in the evaluation procedure is twofold. First, it is used 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, the closer to 0.5 the classification error (between it and the original model), the more accurate a learned model is considered.

Second, it is used 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 with the similarity among the motives belonging to the model and the spurious ones. Moreover, the difficulty grows also when the motives belonging to a same model become similar each other; in this case, in fact, 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 errors among them.

The results obtained under three different conditions of difficulty are summarized in Table 1. The similarity between the six kinds of generated 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.

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).

**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 |

## 5 User Profiling

User profiling is widely used to detect intrusions in computer networks or in telephony networks. The possibility of automatically building a profile for *users* or for *network services* reflecting their temporal behavior would offer a significant help to the deployment of adaptive Intrusion Detection Systems (IDSs).

The experiments described in the following investigate the possibility of automatically constructing a user profile from the logs of its activity. The task that has been selected consists in learning to identify a user from her typing style on a keyboard. The basic assumption is that every user has a different way of typing, which becomes particularly evident when she types words which are specifically important for her, such as her own name, or words referring to her job. This application has not been selected with the goal of challenging the results previously obtained [7, 1, 3], but because it is highly representative of the class of tasks we tackle, and the data are easy to acquire. In other words, if the methodology described so far succeeds in building up a HHMM for this kind of user profiling, it is likely that it will succeed in other cases as well. Two experiments, described in the following subsections, have been performed.

### 5.1 Key Phrase Typing Model

In the first experiment, the goal was to construct a model for a user typing a key phrase, discriminant enough to recognize the user among others. A selected sentence of 22 syllables has been typed many times on the same keyboard, while a transparent program recorded, for every typed key, the duration of each stroke and the delay between two consecutive strokes. Then, every repetition of the sentence generated a sequence, where every key stroke corresponded to an atomic event; the delay between two strokes was represented as a gap, whose length was set to the corresponding duration. Four volunteers provided 140 sequences each, and, for every one of them, a model has been built up using 100 traces (for each user) as learning set. The four learned models have been tested against the remaining 160 traces. For each model  $\lambda$  and for each trace  $s$ , the probability of  $\lambda$  generating  $s$  has been computed using the forward-backward algorithm [9]. Then,  $s$  has been assigned to the model with the highest probability. The results reported only one commission error and two rejection errors (no decision taken), when a trace was not recognized by any one of the models. The models were organized on two levels. The first one contained from 10 to 12 episodes separated

by gaps. Even if the recognition rate is high, it does not seem realistic to use the acquired models to build up a deployable authentication system. In fact, a user profile based on a key phrase only is too restricted. The positive result is that a Markov model of a user typing on a keyboard seems to be appropriate.

## 5.2 Text Typing Model

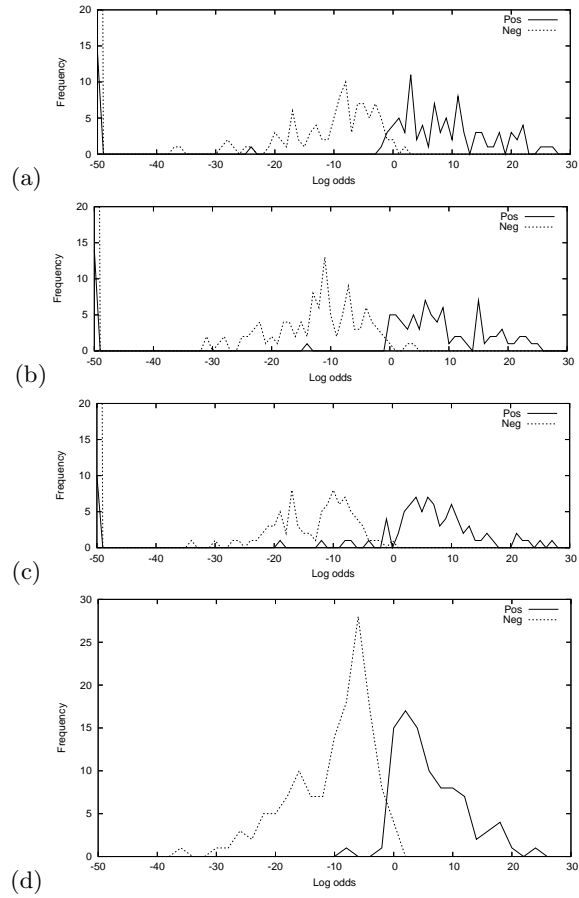
The second experiment addressed the more general problem of modeling a user during a text editing activity. A corpus of several paragraphs, selected from newspapers and books, has been collected. The total number of words was 2280, and the number of typed keys 14273. Again, four users typed the entire corpus in several different sessions, without any constraint, in order not to modify their natural typing style.

In this kind of application, a user model should be centered not on the specific words she types, but on the user typing style, which, in turns, depends on the position of the keys on the keyboard. Therefore, a standard keyboard subdivision into regions, used in dactylography, has been considered, and, on this basis, keys have been grouped into 10 classes. In this way, transition from one region of the keyboard to another should be emphasized. Afterwards, the sequences generated during a typing session have been rewritten by replacing every character with the name of the class it has been assigned to. Moreover, only the gap duration between strokes has been considered, disregarding the length of the key strokes themselves. Finally, long sequences deriving from an editing session have been segmented into shorter sequences, setting the breakpoint in correspondence of long gaps. The idea is that typical delays due to the user typing style cannot go beyond a given limit. Longer delays are imputable to different reasons, such as thinking or change of the focus of attention. In this way a set of about 1350 subsequences has been obtained. For every user, a subset of 220 subsequences has been extracted in order to learn the corresponding model. The remaining ones have been used for testing. As in the previous case, the probability of generating each one of the sequences in the test set has been computed for every model.

The results are summarized in Figure 3, where the distribution of the scoring rate on the test sequences is reported for every model. The scoring rate is measured in *log odds*<sup>2</sup>. The continuous line, labelled "Pos", represents the distribution of the scores assigned to the correct model (user), whereas the other one, labelled "Neg", represents the distribution of the scores assigned to all other models (users), considered together. The sequences on the extreme left have been rejected. It is evident from the figure that sequences belonging to the model are well separated from the other ones. Referring to the data in the test set, a monitoring system using the simple rule that, in a set of three consecutive sequences generated by a user at least two must have a score higher than '0', would give a perfect discrimination of the legal user without rising false alarms.

---

<sup>2</sup> The logarithm of the ratio between the probability that the observed sequence is generated by the model and the probability that it is generated by a random process.



**Fig. 3.** User profiling statistics. Graphs (a), (b), (c) and (d) refer each one to a different user profile. The continuous line "Pos" reports the scoring, measured in log odds, for the sequences belonging to the profile (correct user). The dotted line "Neg" refers to the sequences not belonging to the profile (other users).

It is worth noting that the results have been obtained as a first shot, without requiring any tuning of the algorithm. This means that the method is robust and easy to apply to this kind of problems.

## 6 Conclusion

We have proposed a method for automatically synthesizing from traces complex profiles based on HHMMs. 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, 7], but to test how a general-purpose algorithm performed on a non trivial task for which it was not directly customized. Even if this case study is just a preliminary investigation, the results are promising. In fact, the considered case is highly representative of many other similar problems found in intrusion detection systems. On the one hand, the results show that HHMM is a suitable tool for building profiles for users, or services, in this area. On the other hand, it appears that the methodology is effective, robust and easy to apply.

## 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. D. Gussfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, 1997.
7. R. Joyce and G. Gupta. User authorization based on keystroke latencies. *Communications of the ACM*, 33(2):168–176, 1990.
8. K. Murphy and M. Paskin. Linear time inference in hierarchical hmms. In *Advances in Neural Information Processing Systems (NIPS-01)*, volume 14, 2001.
9. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.