Electronic Letters on Computer Vision and Image Analysis 5(2):30-46, 2005

Learning Model Structure from Data: An Application to On-Line Handwriting

Henri Binsztok, Thierry Artières

LIP6 / Université Paris VI
8, rue du Capitaine Scott
75015 Paris - France
Henri.Binsztok@lip6.fr, Thierry.Artieres@lip6.fr

Abstract

We present a learning strategy for Hidden Markov Models that may be used to cluster handwriting sequences or to learn a character model by identifying its main writing styles. Our approach aims at learning both the structure and parameters of a Hidden Markov Model (HMM) from the data. A byproduct of this learning strategy is the ability to cluster signals and identify allograph. We provide experimental results on artificial data that demonstrate the possibility to learn from data HMM parameters and topology. For a given topology, our approach outperforms in some cases that we identify standard Maximum Likelihood learning scheme. We also apply our unsupervised learning scheme on on-line handwritten signals for allograph clustering as well as for learning HMM models for handwritten digit recognition.

Keywords

HMM Structure Learning, Allograph Clustering, Unsupervised Learning, Online Handwriting

1. Introduction

This paper deals with on-line handwriting signals clustering and Hidden Markov Models (HMM) structure learning. These two problems may be closely related and are of interest in the field of on-line handwriting processing and recognition. Clustering on-line signals is useful for determining allograph automatically, identifying writing styles, discovering new handwritten shapes, etc. HMM structure learning may help to automatically handle allograph when designing an on-line handwriting recognition system. The standard way to learn HMM model is indeed only semi-automatic and requires manual tuning, especially for the HMM topology. Learning HMM models involves learning the structure (topology) and the parameters of the model. Usually, learning consists in first choosing a structure and then in automatically learning the model parameters from training data. Learning parameters is generally achieved with Maximum Likelihood optimization (EM algorithm). Learning of model structure is then implicitly performed manually through successive trials. A fully automatic method would open new perspectives and allow designing easily new recognition engines for any kind of language, characters or drawings.

Fundamentally, we seek to develop learning algorithms for Markovian systems and focus on the learning of mixture models for typical writing styles; it is then very close to model-based clustering. Such techniques

Correspondence to: Thierry.Artieres@lip6.fr

Recommended for acceptance by J.M. Ogier, T. Paquet, G. Sanchez
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

were studied in speech recognition. [LB93] proposed an algorithm that uses probabilistic grammatical inference techniques, which specifically addresses speech variability. A few techniques have been proposed for related tasks within the Handwriting Recognition community, e.g. automatic identification of writing styles, writer identification. For example, [NHP03] proposed a probabilistic approach to define clusters: For each handwritten character, an approach is used to learn the probabilities that a character belongs to a given cluster. The use of HMM for clustering handwritten characters was tackled by [PC00], but their approach depends on initialization so that some supervised information is needed to achieve good performance. Also, [VS97] proposed an interesting hierarchical approach. Besides, more generic approaches have been proposed for sequence clustering, for example [Smy97] presented an algorithm to cluster sequences into a predefined number of clusters, along with a preliminary method to find the numbers of clusters through cross-validation using a Monte Carlo measure. This theoretical approach relies on iterative reestimation of parameters via an instance of the EM algorithm, which requires careful initialization. Furthermore, the structure of the model is limited to a mixture model of fixed-length left-right HMM, which may not model correctly sequences of varying length in the data.

Our goal is to define a learning algorithm for HMM that meets two main requirements. First, the resulting model should describe well the training data. Second, the model should allow identifying sets of similar sequences corresponding to allograph or writing styles. However, the methods discussed above are most often adapted to the task and too restrictive to meet such requirements. Besides, there has been some more generic works dealing with HMM topology learning. Most of these approaches suggest starting by building a complex initial model covering all training data then to simplify it iteratively [Omo92, Bra99, SO93]. In [Bra99], the simplification is based on entropic prior probabilities of the transitions between states, and some transition probabilities converge towards 0, thus simplifying the structure of the model. In [SO93], pair of states from the initial HMM are merged iteratively as long as the loss of likelihood is not too significant. Both approaches, being generic, meet the first requirement but not the second.

We chose to build upon the work from [SO93] and to adapt this method to our goals by restricting the HMM to belong to the class of mixtures of left-right HMMs. As in [SO93] we focus in our work on learning discrete HMM. The learning consists in two steps. In a first step, a global HMM is built from training data, using a procedure to build a left-right HMM from each training sequence. We propose in this step an original procedure for initializing emission probability distribution from the data and discuss its interest with respect to the Maximum Likelihood strategy used in [SO93]. This initial global HMM is then iteratively simplified by removing one left-right HMM in the mixture at a time. This ensures that at any step, the global HMM belongs to the class of mixtures of left-right HMM, which in particular allows performing clustering. This study is an extension of our previous work [BAG04] with new original contributions related mainly to the iterative simplification algorithm and to extended results on different databases.

We first present our unsupervised learning algorithm. First, we detail the building of the initial HMM (section 2). Then, we describe the iterative simplification algorithm applied to this initial model (section 3). The application of our algorithm to cluster sequences and to learn character models in a recognition engine is explained in section 4. The remaining of the paper is dedicated to experiments. We present experimental databases in section 5 and evaluate the emission probability distribution estimation in section 5. The two next sections present experimental results on the two databases for clustering (section 7) and classification (section 8).

2. Building an initial HMM from training data

The main idea for building an initial global HMM covering all training data relies on the build of a left-right HMM from one training sequence. We first detail this idea, then we discuss how to build the global HMM.

Let $D = \{x_1,...,x_n\}$ be a set of training sequences (e.g. a number of handwriting signals corresponding to a character). Each training sequence x_i, whose length is noted l_i, is a sequence of symbols $x_i = \left(s_1^i, s_2^i,..., s_{l_i}^i\right)$ where each symbol s_j^i belongs to a finite *alphabet* Σ.

2.1. Building a left-right HMM from a training sequence

We detail first the structure of a left-right HMM built from a training sequence. Then we discuss its parameters, i.e. emission probability distributions and transition probabilities. We aim at building, from an original training sequence, a HMM that models well (i.e. gives high likelihood to) sequences that are close to the original sequence and that models other sequences badly.

2.1.1. HMM structure

The HMM built from a training sequence $x = (s_1,..., s_l)$ of length l is a left-right HMM with l states, one for each symbol in x. According to this procedure, there exists a natural correspondence between any state and a particular symbol in Σ. This step is illustrated in Figure 1 where a training sequence of length 3 is used to build a three-states left-right HMM.

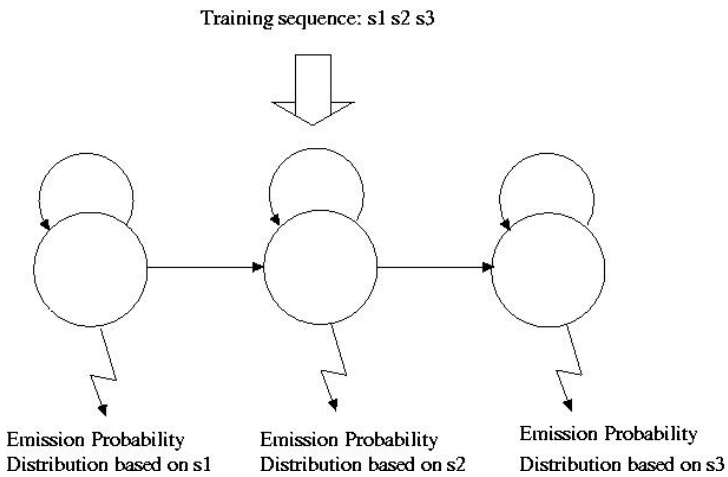

Figure 1: Building a left-right HMM from a training sequence.

As we detail next, the emission probability distribution in a state of such a HMM is determined from the associated symbol in Σ. This ensures that the HMM will score well only sequences that are close to the original sequence.

2.1.2. Parameters

Parameters of an HMM are transition probabilities, emission probabilities and initial state probabilities. Initial state probabilities are completely determined by our choice of left-right HMM (there is only one initial state). Besides, transition probabilities are well known to be a bad approximation of duration in states and we did not learn these here. We seek to use explicit duration models in the future. In the present work, transition probabilities are uniform: 0.5 for both the transition from a state to itself and to the following state.

We explain now how emission probability distributions associated to the states of this HMM are defined; it is based on the correspondence between states and symbols in Σ discussed in previous section.

An optimal solution, from the Maximum Likelihood point of view (i.e. leading to the maximum likelihood of the training sequence), would be to define emission probability of a symbol *s* in a state equal to *1* if the state corresponds to symbol *s* and *0* otherwise. There are a few other smoother solutions. Basically, we want that the emission probability distribution in a state that corresponds to a symbol *s* gives a high probability to symbols that are similar to *s*. [SO93] suggests learning emission probability distributions with a standard Maximum Likelihood criterion using an EM algorithm. However, this strategy did not appear relevant to us since training is delicate insofar as it requires to find a good initialization. Assume we have 1000 training sequences, each of length 10, with the alphabet size |Σ| equal to 50. We therefore have 10 000 symbols to estimate 1000x10 emission probability distributions. If we choose to estimate all probability

distributions without sharing parameters, we would have to estimate 10 000 probability distributions, each defined on Σ with 50 parameters. This is practically impossible to achieve with only 10 000 observed symbols in the training set. The solutions proposed in the literature rely on prior information about parameters and particularly about probability density functions [SO93]. The solution we propose may also be viewed as a use of prior information about these parameters, but this prior knowledge is in our case gained from the training data.

Recall that, according to the procedure used to build an initial global HMM, each state of this HMM is associated to a symbol in Σ. We chose to share emission probability distributions between all states that correspond to a same symbol in Σ so that there are only |Σ| probability distributions to estimate. For instance, if the first state and the last state of a left-right HMM correspond to the same stroke *s*, both states will share the same emission probability distribution. In the above context our strategy requires estimating only 2500 parameters (50 distributions, each one defined with 50 parameters with an alphabet of size 50) from the same number of observations, 10000. In addition, we will show later in our experiments that a Maximum Likelihood Estimation scheme is not necessarily an optimal method for clustering.

These |Σ| emission probability distributions are estimated by countings from *D*, and are based on a similarity measure between symbols in Σ. It is a heuristic method and it is not warranted to be a good approximation of emission probability distributions in all cases. However, it allows to capture efficiently, at least qualitatively, the similarity between symbols and has shown interesting experimental behavior.

We consider as similar two strokes which appear in the same context: Let *x* be any sequence of strokes ($x \in \Sigma^*$), and let $P_x(s)$ be the probability of seeing stroke *s* after sequence *x*. An estimate for $P_x(s)$ may be computed by counting on *D*:

$$P_x(s) = \frac{w(xs)}{w(x)}$$

where *w*(*x*) represents the number of occurrences of the subsequence *x* in *D*. We may then characterize a stroke *s* by a *profile* defined as the following distribution, where Σ^* is the set of strings on Σ:

$$P_s = \left\{P_x(s), x \in \Sigma^*\right\}$$

The idea is that two symbols with similar profiles, i.e. appearing with the same frequency in the same contexts (sequence of symbols in Σ) should be very similar. This distribution may be approximated on *D* by:

$$P_s = \left\{P_x(s), x \in sub_c(D)\right\}$$

where *sub*(*D*) stands for all subsequences of length *c* (the context length) in the training sequences in *D*.

We then define the similarity κ between two strokes $(s_1, s_2) \in \Sigma^2$ by the correlation between the *profiles* P_{s_1} and P_{s_2}:

$$\kappa(s_{1,} s_2) = corr(P_{s1}, P_{s_2})$$

Finally, the emission probability distribution, b_s, in a state corresponding to a symbol *s* is computed by normalizing the above similarities:

$$b_s(s') = \frac{\kappa(s, s')}{\sum_{u \in \Sigma} \kappa(s, u)}, \forall s' \in \Sigma$$

2.2. Building the initial global HMM

Once every training sequence x_i in D has been transformed into a left-right HMM λ_i, a global initial model M_0 is defined as a mixture of all these left-right HMM with uniform priors. This model implements a probabilistic model on symbol sequences:

$$P(x \mid M_0) = \sum_{i=1}^{n} w_i P(x \mid \lambda_i)$$

where x is an observed sequence, λ_i the ith left-right HMM built from x_i and for each i, $w_i = \frac{1}{n}$.

This HMM gives high likelihood to all training sequences in D and gives low likelihood to any sequence that is far from every sequence in D. To sum up ideas for the building of the global HMM, Figure 2 illustrates the procedure for a set of 3 training sequences $D=\{abba, aab, bacca\}$ with an alphabet $\Sigma=\{a,b,c\}$. It is a mixture of three left-right HMMs, each one corresponding to a training sequence. In this construction, each state of the HMM is naturally associated to a symbol in Σ. Probability density functions (p.d.f.) in all states are defined according to this association; for instance states associated to symbol a use a p.d.f. pa.

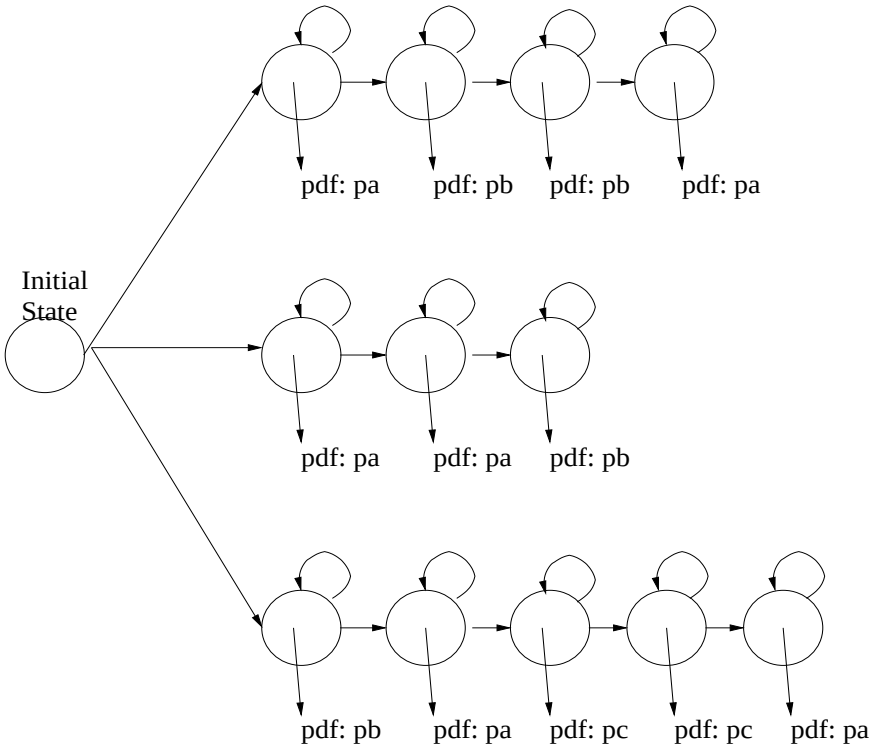

Figure 2: illustration of the building of a global HMM from a training set of three training sequences.

A major difference between our work and previous works lies in these p.d.f., pa, pb and pc. In [SO93], the global HMM that is built maximizes the likelihood of training data. It has the same topology, but p.d.f. are somehow Dirac functions. This means that the p.d.f. associated to symbol a, pa, would be [1 0 0], pb would be [0 1 0]. That means the only possible observation in a state associated to symbol a would be a, the only one possible observation in a state associated to symbol b would be b etc. In our case, p.d.f. are estimated from the data through the computation of a similarity measure between symbols in Σ. This allows, as we will describe next, to design an iterative simplification procedure for the initial global HMM based on a simple likelihood criteria.

3. Iterative simplification algorithm

The general idea of the algorithm is to iteratively merge the two *closest* left-right HMM in the global model, *M*, so that, at the end only typical left-right HMM remain, each one may be viewed as a model of an allograph. However, in order to keep a limited set of emission p.d.f., hence a limited number of parameters, we do not actually merge left-right HMMs but we rather remove less significant left-right HMMs. The principle of the algorithm is then to *select* the best models from the initial mixture model. The iterative simplification algorithm relies on a maximum likelihood criterion and is summed up below:

1. For each sequence of the database, build the corresponding left-right HMM.
2. Build the initial global HMM model $M=M_0$ as detailed in §**3**. Using *n* training data sequences, *M* is a mixture of *n* left-right HMM.
   k=0.
3. Loop:
   At the kth loop, model M_k is a mixture of (*n*-*k*) left-right HMM.
   (a) Build (*n*-*k*) alternate models for M_{k+1} by removing one of the (*n*-*k*) left-right components of M_k.
   (b) Select the alternate model that maximizes the likelihood of the all training data in *D*.

Several stop criteria may be used to determine when to stop simplification. In the context of clustering, this corresponds to strategies for determining the good number of clusters. Unfortunately, it does not exist satisfying methods to determine automatically such an optimal number of clusters; it remains an open problem. In the present implementation, the stop criterion is satisfied when a given number of left-right HMMs is obtained. However we will show experimentally that the likelihood decreases sharply when a right number of clusters is reached. This suggests that standard strategies can provide effective hints to determine automatically a correct number of left-right HMMs.

4. Using the approach for clustering and for classification

Our algorithm leads to a model *M* that is a mixture of a limited number of left-right HMM, which are the most significant to model the whole training data. Such an approach may be used for two different tasks.

First, it may be used for clustering sequences, when viewing each element of the mixture (a left-right HMM) as a model of a given cluster. This may be of interest to identify writing styles, for example to cluster writers according to the way they write some characters.

Consider that *M* is a mixture of *N* left-right HMM, with $N<<n$:

$$P(x \mid M_0) = \sum_{i=1}^{N} w_i P(x \mid \lambda_i)$$

Then, for a given sequence *x*, posterior probabilities of clusters given *x* may be computed with:

$$P(i^{th} cluster / x) = \frac{w_i P(x \mid \lambda_i)}{\sum_{i=1}^{N} w_i P(x \mid \lambda_i)}$$

This allows to assign any sequence *x* to a particular cluster using a Bayes rule, i.e. a maximum posterior probability rule.

Second, the approach may be used to learn character models. For example, we will provide experimental results for the task of digit classification. In these experiments, the algorithm is run independently on the training data for each character, leading to a HMM whose topology and parameters are fully learned from training data. This is an interesting procedure to design, with less manual tuning, a new recognition engine for a particular set of symbols or characters.

5. Experimental databases

We apply our approach to two sets of data. In a first series of experiments we use artifical data generated by a set of HMMs. These experiments, being based on generated data, allow to control the task complexity and thus allow a deep investigation of the behavior of our method. In a second series of experiments, we used real on-line handwritten signals from the Unipen database [GSP94+]. We present now these databases.

5.1. Artificial data

Our artificial data are generated by HMMs, which have already been used in [LK00] to investigate HMM topology learning strategies. Note that we will not compare our approach with their results since these results were only visual.

We used in this study the same HMMs as in [LK00]. There are four discrete HMMs operating over an alphabet Σ of 16 symbols noted 'A' to 'P'. Each HMM is a 5 states left-right model: 4 emitting states and a final state. Self-transition probabilities equal 0.9 so that the expected length of sequences is 40. Each state has a high probability *a* of emitting 4 symbols (either 'A' to 'D', 'E' to 'H', 'I' to 'L' or 'M' to 'P') and a low probability *b* of emitting the 12 others symbols. Figure **3** represents the states of the HMMs. One may tune the complexity of the learning task by varying parameters of the generation process, namely *a* and *b* (note that *a* and *b* are linked since $4a+12b=1$). Of course, recovering the generative HMM models from generated data is easier as *a* increases.

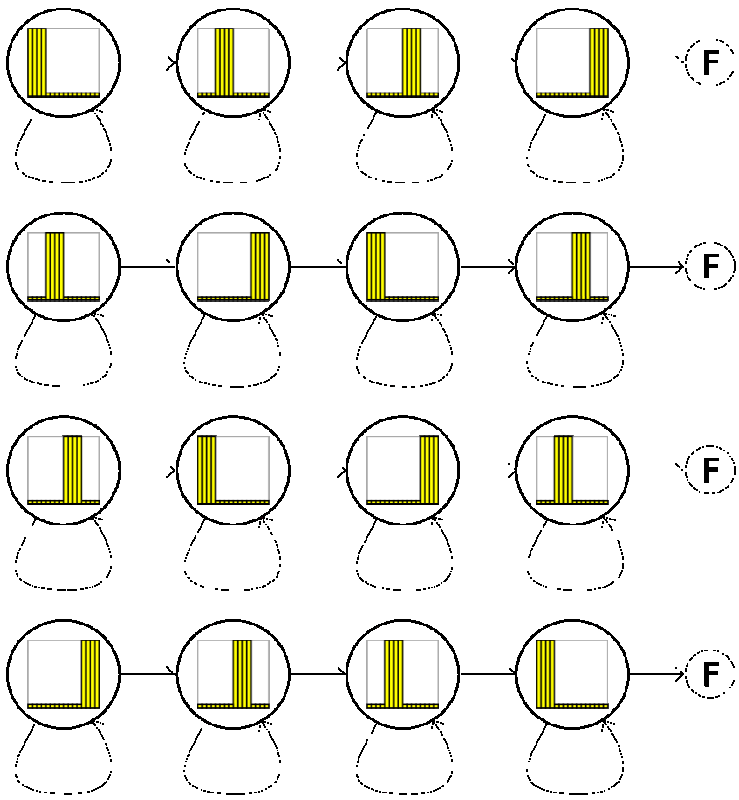

Figure 3: The four generating HMMs for the artificial datasets. F represents the final state; the emission p.d.f. shown correspond to the *easy* dataset.

Two datasets of 1000 sequences were generated from this set of 4 HMMs. The first set is labelled *easy*, with *a = 0.22*. The second set is labelled *hard*, with *a = 0.15*. Table 1 shows statistical details for these two datasets.

	Average Length	Min. Length	Max. Length	Bayes Error	Noise Ratio	Parameters
Set easy	40.23	7	122	0.7%	13.6%	a = 0.22 b = 0.01
Set hard	42.34	6	109	13.4%	66.7%	a = 0.150 b = 0.034

Table 1: Statistical details about the two artifical datasets

5.2. On-line handwritten signals

We carried out our experiments on on-line handwritten digits written by about 100 writers, extracted from the Unipen database [GSP94+]. The rough on-line signal is a temporal sequence of pen coordinates and is first preprocessed as in [AG02] using a kind of direction coding. A handwritten signal is represented as a sequence of symbols that are *strokes*; each stroke is characterized by a direction and a curvature. The strokes belong to a finite dictionary Σ of 36 elementary *strokes*, including 12 straight lines in directions uniformly distributed between 0 and 360°, 12 convex curves and 12 concave curves. This set of elementary strokes is illustrated in Figure 4.

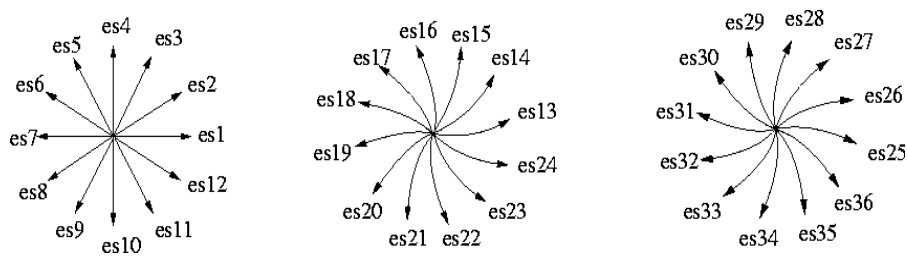

Figure 4: Set Σ of 36 fixed elementary strokes used to represent handwriting signals - from left to right: 12 straight lines (named es_1 to es_{12}), 12 convex strokes (named es_{13} to es_{24}) and 12 concave strokes (named es_{25} to es_{36}).

At the end of the preprocessing step, an on-line handwritten signal is represented as a sequence of symbols belonging to the alphabet Σ. This representation is computed through dynamic programming [AG02]. Such a sequence of strokes represents the shape of the signal and may be efficiently used for recognition.

We used several subsets of the database: 1000 samples of digits ’0’ and ’9’, 1000 samples of all ten digits, and about 6000 samples of all ten digits for classification.

6. Probability density function estimation

We investigate here the quality of our method for estimating emission probability distributions.

6.1. Artificial data

As the topology of the generating HMMs suggests, there is a close similarity between the first four symbols ‘A’ to ‘D’, etc. Therefore, the artificial datasets are useful to test the validity of our estimation model.

Figure 5 shows the estimated emission probability distributions for the *easy* dataset in matrix form. The dimensions of the matrix are 16x16. The jth column of the matrix corresponds to the emission probability distribution in a state associated to the jth symbol. The pixel at the intersection of the ith row and j^{th} column is

the probability of observing the ith symbol in a state corresponding to the jth symbol in Σ ; Gray levels are proportional to probabilities (white: close to 1, black: close to 0).

We see that our estimation model captures well the information at the symbol level with the *easy* dataset.

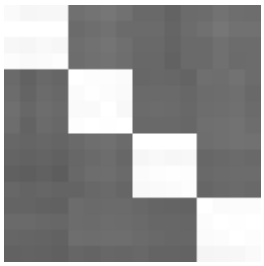

Figure 5: Similarities between symbols inferred from the *easy* dataset, with context length 1. The jth column of the matrix corresponds to the emission probability distribution in a state associated to the jth symbol.

Figure 6 represents the estimation of emission laws associated to symbols from the *hard* dataset. We have used three different context lengths *c* (see section 3.1.2). The number of training sequences being limited the estimation of emission probabilities naturally tend to 1 for $\kappa(s_{1,}s_1)$ and 0 for $\kappa(s_{1,}s_2), s_2 \neq s_1$ when *c* increases. Therefore, a context length *c=1* provides best estimation results in our experiments since it does not introduce artefacts.

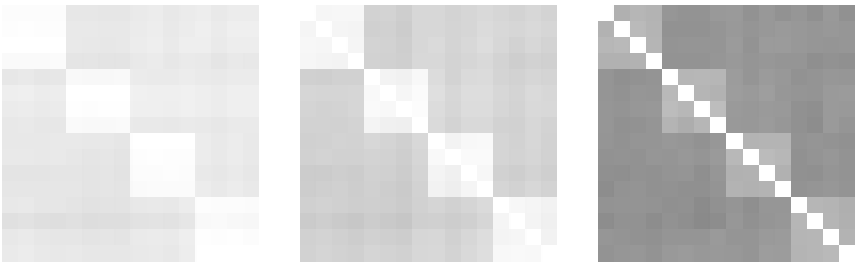

Figure 6: Similarities between symbols of the alphabet inferred from the *hard* dataset, with different context lengths (from left to right: 1, 2, 3).

6.2. Handwritten signals

Figure 7 represents two sets of emission probabilities distributions over alphabet Σ of 36 elementary strokes in the same matrix form as above. The dimensions of the matrixes are then 36x36. The pixel at the intersection of the ith row and j^{th} column is the probability of observing the ith stroke in a state corresponding to the jth stroke in Σ.

The left matrix has been tuned manually according to prior knowledge [AG02] while the right matrix is estimated with the method presented in section §2.1.2. As may be seen, there are strong correlations between these two matrices, which shows that our estimation method allows capturing efficiently, from the training database *D*, the similarity between symbols.

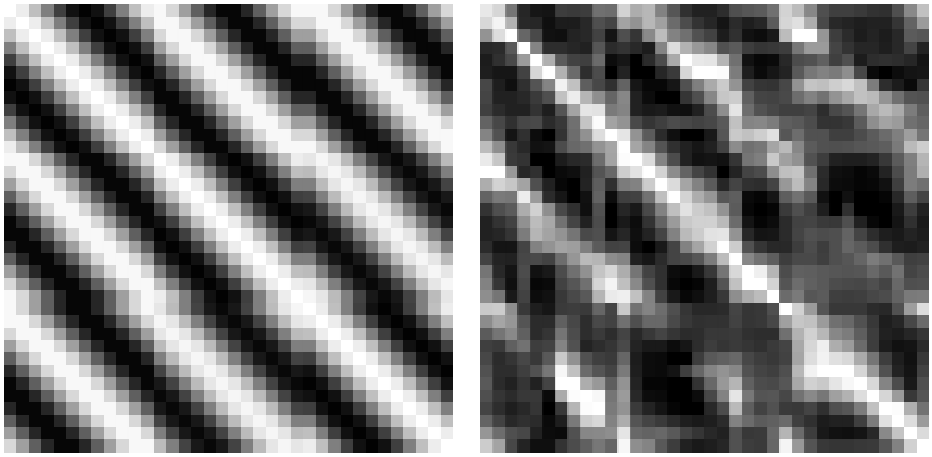

Figure 7: 36x36 matrices representing probability distributions of states associated to strokes of Σ. The matrix on the left has been tuned manually using prior knowledge and the matrix on the right has been learned from the data using the procedure in §2.1.2.

7. Clustering experiments

We present here experimental results for the sequences clustering task. We first discuss the evaluation criteria. And then we present a benchmark method, with which we compare our approach. Finally, we present experiments on artifical data and on handwritten signals.

7.1. Evaluation criteria

Evaluating unsupervised methods (e.g. clustering) is still an open problem. This is not a problem for artificial data experiments since we do have an ideal labeling information for these data (we know which HMM generated each sequence). This is more problematic for the handwritten signals since we do not have any label information about allographs. Then, for these data, we chose to perform clustering experiments on databases including signals of various but close digits (e.g. ’0’ and ’9’). This allows an objective evaluation of clustering using the available label information.

Hence, we evaluated clustering results in the following way: After learning of a mixture of left-right HMM from the data, all sequences in the database are clustered using these left-right HMMs as cluster models. We then use criteria relying on a labeling of samples with class information (e.g. digits) to evaluate the clustering results.

A few criteria may be used to evaluate clustering [SKK00]. Among these, we chose the *precision* measure that is also used in classification. In the following, we name *clusters* the result of our clustering and *classes* the labeling of the data. For a cluster j, P_{ij} is the probability that an element of the cluster j belongs to class i. This probability is estimated by counting: Let n_j be the number of sequences in cluster j and n the total number of sequences in

the data. We note $\max_i$ the maximum of all possible values for *i*. Then:

$$precision = \sum_j \frac{n_j}{n} \max_i P_{ij}$$

7.2. Benchmark method

In order to give more insights of our approach, labelled **BAG** in the figures, we provide some comparative results using a standard learning scheme for HMM parameters, based on the CEM algorithm (stochastic EM). It is a variant of the EM algorithm that may outperform EM in unsupervised learning, especially when dealing with too few data to estimate the likelihood correctly [CD88].

For each number of clusters *K*, we learn a HMM, whose topology is a mixture of left-right HMMs. We use this HMM to perform clustering, using a Maximum Likelihood estimation scheme. To use such a learning strategy, one has to define first the topology of the model and then to initialize parameters (emission probability distributions). We have investigated two ways to do this. The first one, named **CEM1**, is based on a *k*-means like algorithm and a distance between sequences [REFF]. The second approach, named **CEM2**, uses the global HMM obtained with our method after a number of iterations (*n-K*). It may be seen as an upper-bound of Maximum Likelihood estimation performance since this initialization is, as we will show experimentally, already a good solution.

7.3. Experiments on artificial data

First, we investigate the clustering performance of our approach and compare this to CEM reestimation (CEM2). This favours the CEM approach, since the main problem for the latter is to find a correct initialization. On the *easy* dataset, as may be seen in Figure 8, our approach outperforms CEM and its performance is close to the Bayes error of classification, though the learning is totally unsupervised. With the *easy* dataset, we also see that the likelihood function shows an inflexion point for the "good" number of clusters, i.e. the number of HMMs that generated the data. This allows to easily detect the correct number of clusters.

A look to cluster models reveals that our approach correctly identifies the *best shortest sequences* that are typical for each model. Our analysis is that the strength of our approach is to correctly identify the most typical sequences in the data, and use them as cluster models. Furthermore, we can stress that, given the high probability of self-transition (0.9), there is a high tendancy to have in the data much longer sequences that there are number of states in the generating models. Therefore, to minimize the probability of having a misrepresentative state in the cluster models, the shorter the sequence, the more likely it is to have only "good" states. But there are also fewer short sequences present in the data.

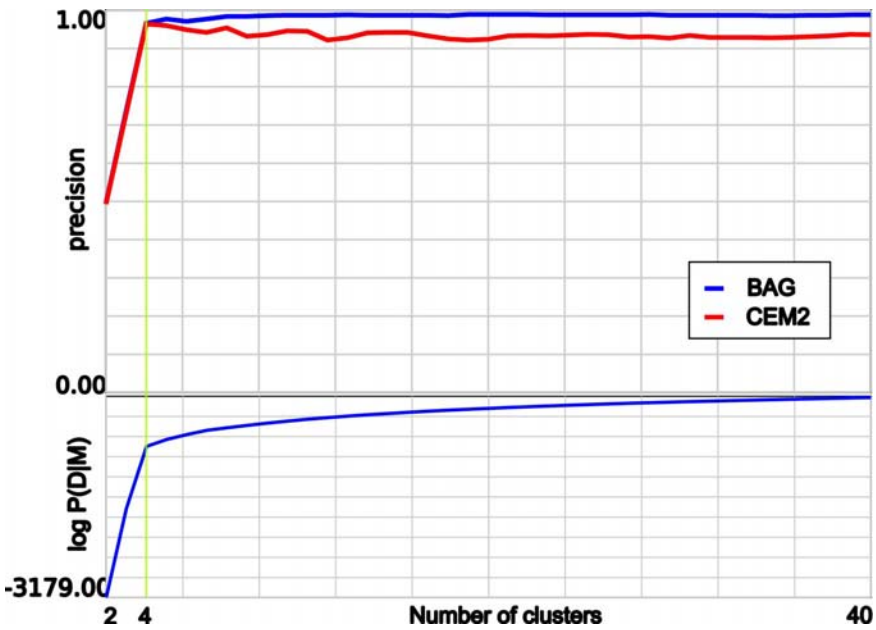

Figure 8: Above, Performance on the *easy* dataset comparing our approach (BAG) to CEM2, an EM reestimation of the model learned with our approach (above). Below, logarithm of the likelihood, showing an inflexion point for 4 clusters, which corresponds to the number of HMMs that generated the data.

The *hard* dataset provides weaker results, which is logical, given the high noise ratio. Figure 9 shows the clustering results for all three approaches (BAG, CEM1 and CEM2). There is no clear tendency between BAG and CEM1: CEM1 gives better results for a low number of clusters, our approach gives better results for a high number of clusters.

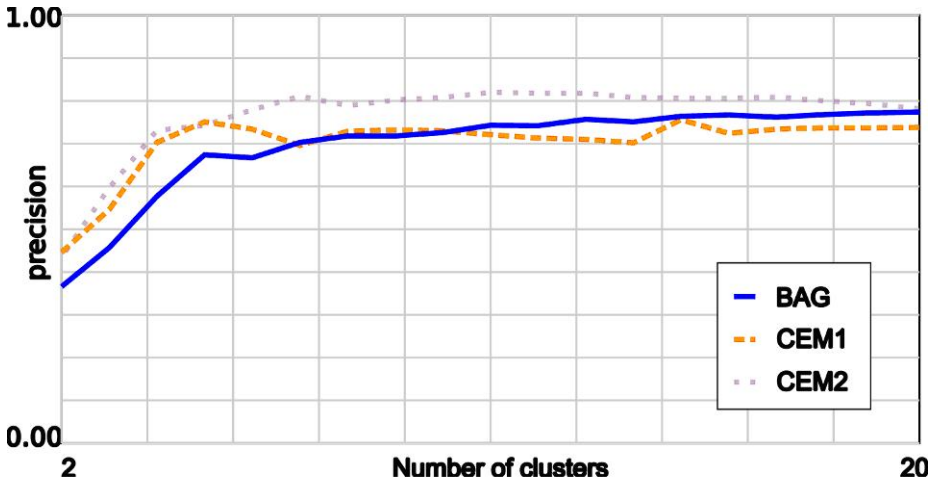

Figure 9: Performance on the *hard* dataset, comparing our approach (BAG) to the CEM1 and CEM2 clustering approaches.

For CEM2, we used our approach (BAG) as the initialization of the CEM clustering algorithm and CEM2 provides better results. We can explain this using our previous interpretation: Our approach works by selecting the most representative sequences of the model in the data. Indeed, as we could check by a deep look at the data, there is simply no single fully representative sequence of each model, since the noise ratio is very high in the *hard* dataset. Therefore, the selected cluster models contain some “bad” states, and our

approach can not modify the left-right HMMs which are part of the model, whereas the CEM reestimation does.

In the next section, we will look to our real world application – handwritten digit clustering and classification – to see how our approach compares, and whether there exists at least some good sequences in the data in a real-world application.

7.4. Experiments on on-line handwritten signals

In a first series of experiments, we used 100 samples of digits ’0’ and ’9’ whose drawings are very similar. As an illustration, the resulting clusters from one experiment using our model are drawn in Figure 10: The discovered clusters are homogeneous (including either ’0’ or ’9’ samples). The two clusters for digit ’0’ include indeed slightly different drawings since samples from the smaller set are drawn the other way round. In this figure, the drawing is generated from our model representation; therefore, characters do not display as nicely as the fine-grained original representation.

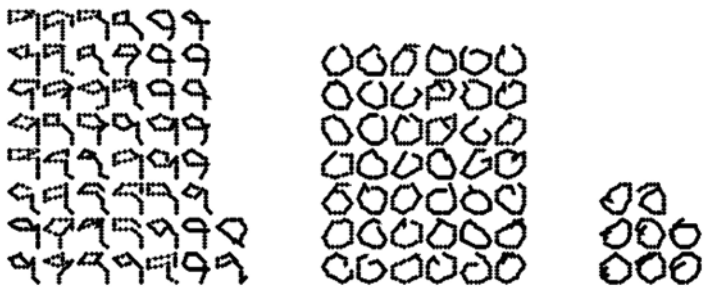

Figure 10: The three discovered clusters for a database of on-line handwriting samples of digits ’0’ and ’9’

To demonstrate the ability of our approach to discover allographs, we applied our clustering approach to 500 samples of handwritten digit ’2’. Since we do not have any allograph labeled database, it is difficult to define an interesting evaluation criteria, but we show the resulting clusters (Figure 11). One may recognize some typical allograph of this digit: drawn in ’Z’ shape, upper round, lower round, etc. We note however that the cluster limits are not always well defined and some examples could be affected to different clusters. This visual evaluation is completed by more quantitative results next.

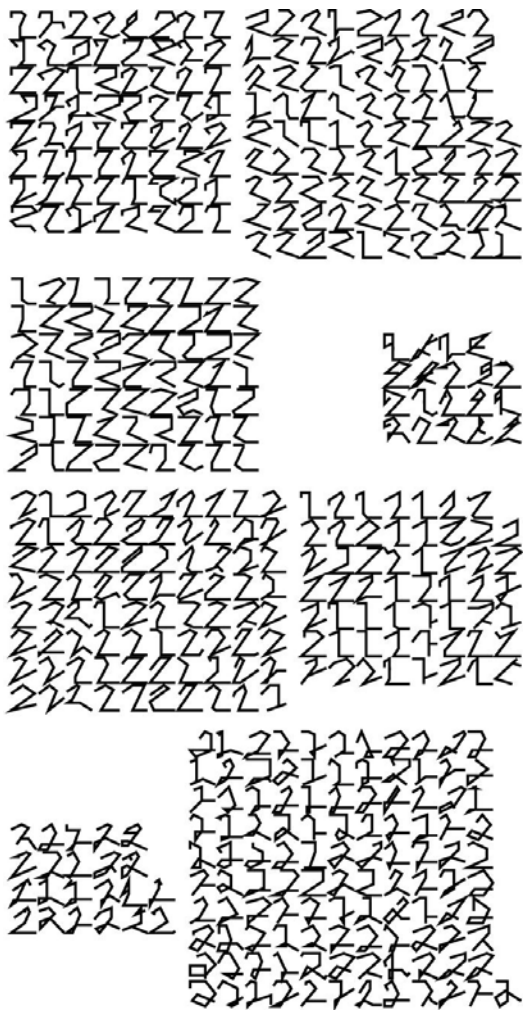

Figure 11: Visualization of allograph of handwritten digit '2' with 8 clusters.

To further validate our approach, we performed another comparison on a set of 1000 samples of the ten digits. Figure 12 compares the performance of our approach with emission probability distributions tuned by hand (cf. §6 and Figure 7) or estimated using the technique detailed in §2.1.2.

The graphs are labeled "BAG (Fix)" and "BAG (Est)". Results (using the precision measure defined in §7.1) are given as a function of the number of clusters identified (i.e. all along the iterative learning algorithm of §3, as the number of clusters decreases). In addition to these two systems we provide results obtained with the benchmark method (CEM2). Hence, at each step of the simplification algorithm, i.e. for any number of clusters, the resulting models *M* are re-estimated with the CEM algorithm. Graph "CEM2 (Fix)" use the model learned with manually tuned emission probability distributions, while "CEM2 (Est)" use the model using distributions estimated from the data.

For example, for 20 clusters, our approach leads to about 86% accuracy with tuned emission probability distributions and to 83% with estimated emission probability distributions. These two systems when re-estimated using a CEM optimization lead respectively to 80% and 74% accuracy.

As may be seen, whatever the number of clusters, CEM re-estimation lowers the performance, although it maximizes the likelihood. Note that, assuming that there are, in average, two allographs per digit, we are mostly interested here in the performance for about 20 clusters; i.e. a perfect automatic method would find the 20 clusters that would represent all allographs. The reason for the ineffectiveness of CEM reestimation scheme is not clear. However, we think that our learning strategy is naturally more adapted to discover typical cluster of sequences. The reason lies in that a left-right HMM built from a training sequence, as detailed in section §2.1 cannot handle much variability around the original training sequence. Thus it leads to compact and homogeneous clusters. At the opposite, performing CEM re-estimation may result in less

specific left-right HMM, thus in less precise clusters. These results answer the question we left open in section 6.1. Our approach depends on its ability to find typical sequences in the data. Indeed, in our real application, there are at least some characters that are well recorded and associated to a given handwritten character.

At last, we conducted an experiment using 1000 samples of the letters ’a’ and ’d’, which are often confused in online handwriting systems. Whereas the precision is only 60% for 2 clusters, it jumps to 95% for 5 clusters, which constitutes a rather acceptable approximation of the number of allograph for these two characters.

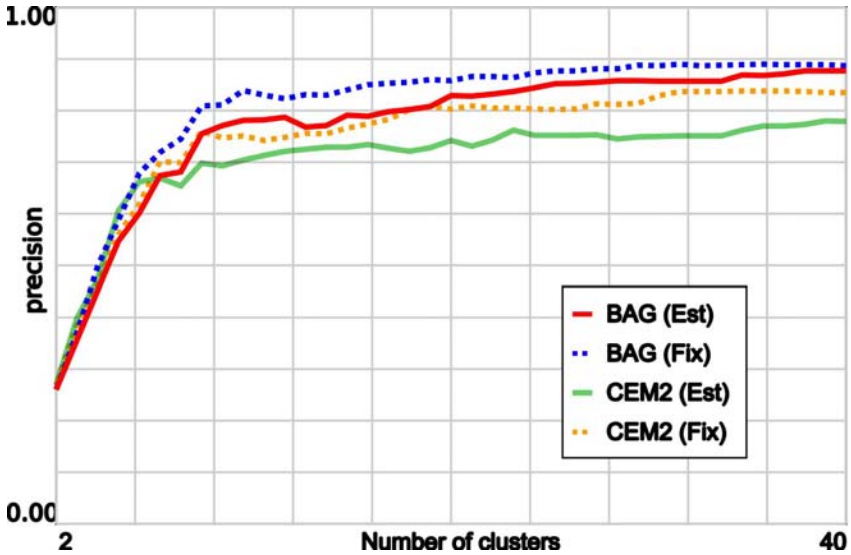

Figure 12: Clustering performance using our approach (BAG) and CEM2. The red graph BAG (Est) corresponds to the model using estimated emission probability distributions; the blue graph BAG (Fix) corresponds to the model using manually tuned emission probability distributions. Green (CEM2 (Est)) and yellow (CEM2 (Fix)) graphs correspond to the re-estimation of the two models BAG (Est) and BAG (Fix).

Preceding results show that clustering is indeed a difficult task since for a reasonable number of clusters (20) precision does not exceed 85% whereas classification results on such handwriting signals may reach about 95% [AG02]. However, our unsupervised approach outperforms benchmark methods provided there are enough clusters while performance falls sharply when the number of clusters decreases.

8. Classification experiments

We present here experiments on learning character models for classification tasks. In this section, we use our learning algorithm to learn, for every digit, a digit model that is a mixture of left-right HMM. Experiments were performed on a bigger subset of Unipen, about 6000 samples of the ten digits (from ’0’ to ’9’) with 800 samples for training and the remaining for test. Recognition rates are displayed in Figure 13 as a function of the number of left-right HMMs in a character model. Without simplification of the initial HMMs (i.e. about 80 left-right HMM per digit) the classification accuracy reaches an asymptotic performance of 92.5%. By learning a model for each digit, we can achieve same or better performance while simplifying the models up to 7 left-right HMM per digit in average.

Note that these performance do not match state of the art recognition rates [Rat03]. The main reason is that we did not model specific parameters of handwriting recognition (i.e. duration model and pen-up moves) to keep the generality of our approach. However, in sight of these shortcomings, our results appear promising since we obtain the same level of performance than by using the approach described in [MSAG03].

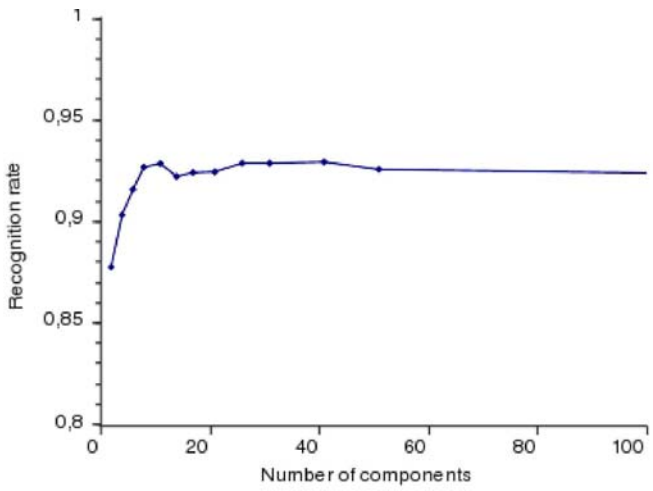

Figure 13: Recognition rate (for digit recognition) as a function of the number of components (left-right HMMs) per digit model.

9. Conclusions

We presented a model-based approach to cluster sequences that we tackled through unsupervised HMM learning. We proposed to learn, from the data, the structure and parameters of a global HMM that is a mixture of left-right HMMs. This structure seems much appropriate for sequence clustering and allograph identification. The learning consists in building from data an initial mixture model of left-right HMMs that cover all training data and then simplifying it by removing iteratively the less significant left-right HMM. This algorithm relies on an original estimation of emission probability distributions. We provide experimental results on artificial data that show that our approach is efficient for learning HMM topology. Furthermore, we obtained an unexpected and interesting result: for a fixed HMM topology our approach may outperform Maximum Likelihood re-estimation in some cases. We also applied our approach to clustering and classification of on-line handwritten digits. These results confirm the ones obtained on artificial data. Furthermore, clustering as well as classification results are promising, showing for instance that it is possible to learn complete character models from the data without any manual tuning of model topology.

References

[AG02] Thierry Artières and Patrick Gallinari. Stroke level HMMs for on-line handwriting recognition. In *8th International Workshop on Frontiers in Handwriting Recognition (IWFHR-8)*, Niagara, August 2002, pages 227-232.

[BAG04] Henri Binsztok, Thierry Artières, Patrick Gallinari: A Model-Based Approach to Sequence Clustering. *European Conference on Artificial Intelligence* (ECAI 2004). Valencia, Spain. Pages 420-424.

[Bra99] M. Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation*, 11:1155–1182, 1999.

[CD88] Gilles Celeux and Jean Diebolt. A random imputation principle : the stochastic em algorithm. *Technical report, Rapport de recherche de l'INRIA- Rocquencourt*, 1988.

[CGS00] Igor V. Cadez, Scott Gaffney, and Padhraic Smyth. A general probabilistic framework for clustering individuals and objects. In Raghu Ramakrishnan, Sal Stolfo, Roberto Bayardo, and Ismail Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pages 140–149, N. Y., August 20–23 2000. ACM Press.

[GSP+94] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and benchmarks. In *International Conference on Pattern Recognition, ICPR'94*, pages 29–33, Jerusalem, Israel, 1994. IEEE Computer Society Press.

[LB93] Philip Lockwood and Marc Blanchet. An algorithm for the dynamic inference of hidden markov models (DIHMM). In *Proceedings of ICASSP*, pages 251–254, 1993.

[LK00] M. H. Law, J. T. Kwok. Rival penalized competitive learning for model-based sequence clustering. *International Conference of Pattern Recognition, ICPR'00*, Barcelona, Spain, 2000. Pages 2195-2198.

[MSAG03] Sanparith Marukatat, Rudy Sicard, Thierry Artières, and Patrick Gallinari. A flexible recognition engine for complex on-line handwritten character recognition. In *7th International Conference on Document Analysis and Recognition (ICDAR 2003)*, Edinbourgh, Scotland, August 2003, pages 1048-1052.

[NHP03] Ali Nosary, Laurent Heutte, and Thierry Paquet. Unsupervised writer adaptation applied to handwritten text recognition. *Pattern Recognition*, 37:385–388, 2003.

[Omo92] Stephen M. Omohundro. Best-first model merging for dynamic learning and recognition. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 958–965. Morgan Kaufmann Publishers, Inc., 1992.

[PC00] M. Perrone and S. Connell. K-means clustering for hidden markov models. In *In Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 229–238, Amsterdam, Netherlands, September 2000.

[Rat03] Eugene H. Ratzlaff. Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database. In *Seventh International Conference on Document Analysis and Recognition*, Edinburgh, Scotland, August 2003, pages 623-628.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, 2000.

[Smy97] Padhraic Smyth. Clustering sequences with hidden markov models. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 648. The MIT Press, 1997.

[Spe03] Speed Terry. Statistical analysis of gene expression microarray data, Terry Speed Ed.

[SO93] Andreas Stolcke and Stephen Omohundro. Hidden Markov Model induction by bayesian model merging. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 11–18. Morgan Kaufmann, San Mateo, CA, 1993.

[VS97] L. Vuurpijl and L. Schomaker. Finding structure in diversity: A hierarchical clustering method for the categorization of allographs in handwriting. *International Conference on Document Analysis and Recognition* 1997. Pages 387-393.