Embedded Bernoulli Mixture HMMs for Handwritten Word Recognition*

Adrià Giménez and Alfons Juan
DSIC/ITI,Univ. Politècnica de València, E-46022 València (Spain)
{agimenez,ajuan}@dsic.upv.es

Abstract

Hidden Markov Models (HMMs) are now widely used in off-line handwritten word recognition. As in speech recognition, they are usually built from shared, embedded HMMs at symbol level, in which state-conditional probability density functions are modelled with Gaussian mixtures. In contrast to speech recognition, however, it is unclear which kind of real-valued features should be used and, indeed, very different features sets are in use today. In this paper, we propose to by-pass feature extraction and directly fed columns of raw, binary image pixels into embedded Bernoulli mixture HMMs, that is, embedded HMMs in which the emission probabilities are modelled with Bernoulli mixtures. The idea is to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. Empirical results are reported in which similar results are obtained with both Bernoulli and Gaussian mixtures, though Bernoulli mixtures are much simpler.

1. Introduction

Hidden Markov Models (HMMs) are now widely used in off-line handwritten recognition [3, 4]. Given a word image, it is first transformed into a sequence of fixed-dimension feature vectors, and then fed into a word HMM-based classifier to decide on its most probable word.

In principle, each word can be modelled by its own HMM, with no parameters in common with the HMMs associated with other classes. However, this approach becomes impractical for large vocabularies due to lack of training data for infrequent words, which results in poorly estimated HMM parameters and degraded classifier performance. Following the usual approach in speech recognition [6], from where the HMM methodology was imported, word-conditional HMMs are instead built from shared, *embedded* HMMs at symbol (subword) level. In this way, each training word image contributes to the estimation of its constituent symbol HMMs, all symbol HMMs are reliably estimated, and infrequent words are better modelled.

HMMs at symbol level are usually simple in terms of number of states and topology; e.g., 6 states and a linear topology in which each state can only be reached from its preceding state or itself (loop). On the other hand, state-conditional probability (density) functions depend on the type of output that has to be emitted. In the common case of real-valued feature vectors, Gaussian mixtures are generally preferred since, as with finite mixture models in general, their complexity can be adjusted to the available training data by simply varying the number of mixture components. Another good reason for their use is the availability of reliable software from the speech recognition community [8].

After decades of research in speech recognition, the use of certain real-valued speech features and embedded Gaussian mixture HMMs is a de-facto standard [6]. However, in the case of handwritten word (text) recognition, there is no such a standard and, indeed, very different sets of features are in use today [3, 4]. In this paper, we propose to by-pass feature extraction and directly fed columns of raw, binary image pixels into *embedded Bernoulli mixture HMMs,* that is, embedded HMMs in which the emission probabilities are modelled with Bernoulli mixtures. The basic idea is to ensure that no discriminative information is filtered out during feature extraction, which in some sense is integrated into the recognition model. It must be noted that we have already obtained comparatively good results with both, conventional (not embedded) Bernoulli HMMs (for small vocabularies) [2] and embedded (single) Bernoulli HMMs [1]. However, the more general case of embedded Bernoulli mixture HMMs is first considered in this work.

The paper is organised as follows. In Sections 2 and 3, we review plain Bernoulli mixtures and Bernoulli mixture HMMs. Embedded Bernoulli mixture HMMs are presented in Section 4, while their estimation by maximum likelihood is described in Section 5. In Section 6, empirical results are reported. Concluding remarks are discussed in Section 7.

*Work supported by the EC (FEDER) and the Spanish MEC under the MIPRCV "Consolider Ingenio 2010" research programme (CSD2007-00018), the iTransDoc research project (TIN2006-15694-CO2-01), and the FPU grant AP2005-1840.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.66

2. Bernoulli mixture

Let $\mathbf{o}$ be a D-dimensional feature vector. A finite mixture is a probability (density) function of the form:

$$p_{\mathbf{\Theta}}(\mathbf{o}) = \sum_{k=1}^{K} \pi_k \, p_{\mathbf{\Theta}'}(\mathbf{o} \mid k) \ , \tag{1}$$

where K is the number of mixture components, π_k is the kth component coefficient, and $p_{\mathbf{\Theta}'}(\mathbf{o} \mid k)$ is the kth component-conditional probability (density) function. The mixture is controlled by a parameter vector $\mathbf{\Theta}$ comprising the mixture coefficients and a parameter vector for the components, $\mathbf{\Theta}'$. It can be seen as a generative model that first selects the kth component with probability π_k and then generates $\mathbf{o}$ in accordance with $p_{\mathbf{\Theta}'}(\mathbf{o} \mid k)$.

A Bernoulli mixture model is a particular case of (1) in which each component k has a D-dimensional Bernoulli probability function governed by its own vector of parameters or *prototype* $\mathbf{p}_k = (p_{k1}, \dots, p_{kD})^t \in [0,1]^D$,

$$p_{\mathbf{\Theta}'}(\mathbf{o} \mid k) = \prod_{d=1}^{D} p_{kd}^{o_d} (1 - p_{kd})^{1 - o_d} \ , \tag{2}$$

where p_{kd} is the probability for bit d to be 1. Note that this equation is just the product of independent, unidimensional Bernoulli probability functions. Therefore, for a fixed k, it can not capture any kind of dependencies or correlations between individual bits.

Consider the example given in Figure 1. Three binary images (**a, b** and **c**) are shown as being generated from a Bernoulli prototype depicted as a grey image (black=1, white=0, grey=0.5). The prototype has been obtained by averaging images **a** and **c,** and it is the best approximate solution to assign a high, equal probability to these images. However, as individual pixel probabilities are not conditioned to other pixel values, there are $2^6 = 64$ different binary images (including **a, b** and **c**) into which the whole probability mass is uniformly distributed. It is then not possible, using a single Bernoulli prototype, to assign a probability of 0.5 to **a** and **c,** and null probability to any other image such as **b.** Nevertheless, this limitation can be easily overcome by using a Bernoulli mixture and allowing a different prototype to each different image shape. That is, in our example, a two-component mixture of equal coefficients, and prototypes **a** and **b,** does the job.

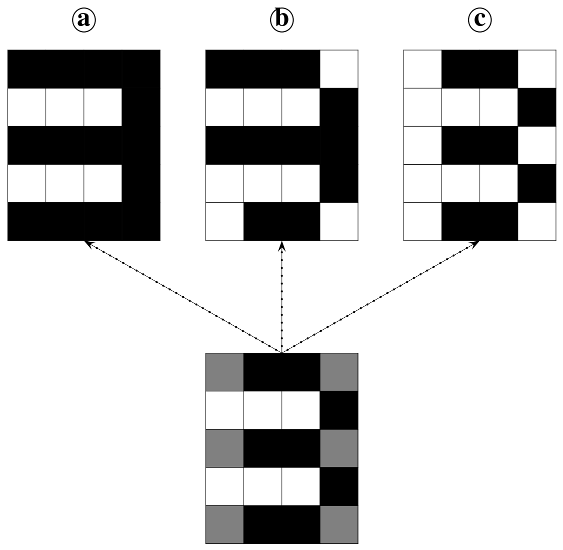

Figure 1. Three binary images (a, b and c) are shown as being generated from a Bernoulli prototype depicted as a grey image (black=1, white=0, grey=0.5).

3. Bernoulli mixture HMM

Let $O = (\mathbf{o}_1, \dots, \mathbf{o}_T)$ be a sequence of feature vectors. An HMM is a probability (density) function of the form:

$$p_{\mathbf{\Theta}}(O) = \sum_{I, q_1, \dots, q_T, F} a_{I q_1} \left[\prod_{t=1}^{T-1} a_{q_t q_{t+1}} \right] a_{q_T F} \prod_{t=1}^{T} b_{q_t}(o_t) \ , \tag{3}$$

where I and F are the special states for *start* and *stop* respectively, a_{ij} is the state-transition probability between states i and j, b_j is the observation probability (density) function in state j, and q_t denotes the state at time t.

A Bernoulli mixture HMM is an HMM in which the probability of observing $\mathbf{o}_t$, when $q_t = j$, is given by a Bernoulli mixture probability function for the state j:

$$b_j(\mathbf{o}_t) = \sum_{k=1}^{K} \pi_{jk} \prod_{d=1}^{D} p_{jkd}^{o_{td}} (1 - p_{jkd})^{1 - o_{td}} \ , \tag{4}$$

where π_j are the priors of the jth state mixture components, and $\mathbf{p}_{jk}$ is the kth component prototype in state j.

Consider the left part of Figure 2, where a Bernoulli mixture HMM for number 3 is shown, together with a binary image generated from it. It is a two-state model with a single prototype attached to state 1, and a two-component mixture assigned to state 2. In contrast to the example in Figure 1, prototypes do not account for whole digit realisations, but only for single columns. This column-by-column emission of feature vectors attempts to better model horizontal distortions at symbol level and, indeed, it is the usual approach in both speech and handwriting recognition when continuous-density (Gaussian mixture) HMMs are used. The reader can easily check that, by direct application of Eq. (3), the probability of generating the binary image is 0.02835.

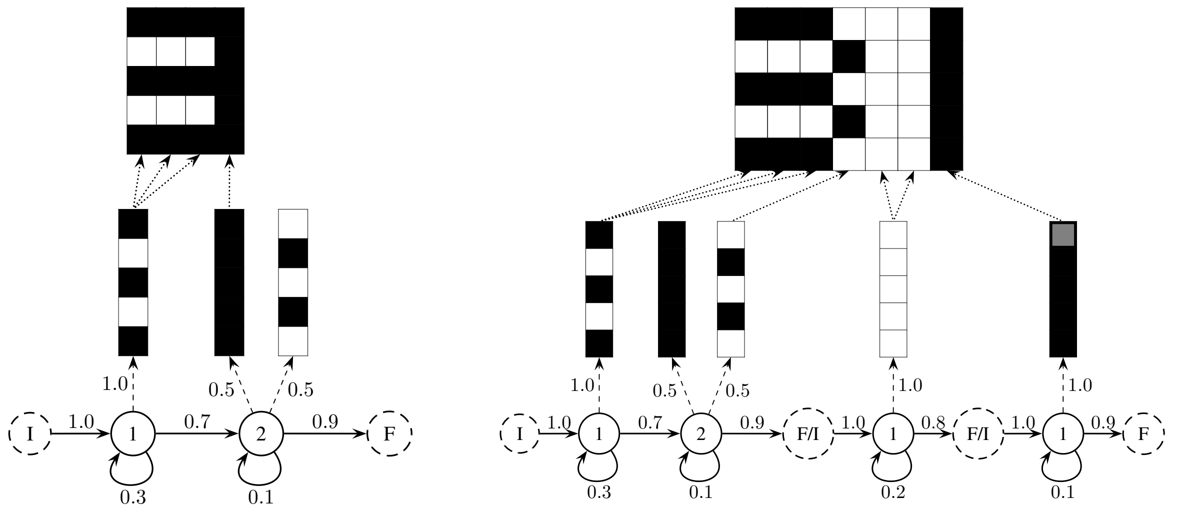

Figure 2. Example of embedded Bernoulli mixture HMMs for the numbers 3 (left) and 31 (right), and binary images generated from them. Note that a shared Bernoulli mixture HMM for digit 3 is used.

4. Embedded Bernoulli mixture HMM

Let C be the number of different characters (symbols) from which words are formed, and assume that each character c is modelled with a different HMM of parameter vector λ_c. Let $\mathbf{\Lambda} = \{\lambda_1, \dots, \lambda_C\}$, and let $O = (\mathbf{o}_1, \dots, \mathbf{o}_T)$ be a sequence of feature vectors generated from a word formed by a sequence of symbols $S = (s_1, \dots, s_L)$, with $L \leq T$. The probability (density) of O can be calculated, using embedded HMMs for its symbols, as:

$$p_{\mathbf{\Lambda}}(O \mid S) = \sum_{i_1,\dots,i_{L+1}} \prod_{l=1}^{L} p_{\lambda_{s_l}}(\mathbf{o}_{i_l}, \dots, \mathbf{o}_{i_{l+1}-1}) \,, \quad (5)$$

where the sum is carried out over all possible segmentations of O into L segments, that is, all sequences of indices $i_1, \dots, i_{L+1}$ such that $1 = i_1 < \dots < i_L < i_{L+1} = T+1$; and $p_{\lambda_{s_l}}(\mathbf{o}_{i_l}, \dots, \mathbf{o}_{i_{l+1}-1})$ refers to the probability (density) of the lth segment, as given by (3) using the HMM associated with symbol s_l.

Consider now the right part of Figure 2. An embedded Bernoulli mixture HMM for number 31 is shown, which is the result of concatenating Bernoulli mixture HMMs for digit 3, blank space and digit 1, in that order. Note that the HMMs for blank space and digit 1 are simpler than that for digit 3. Also note that the HMM for digit 3 is shared between the two embedded HMMs shown in the Figure. The binary image of the number 31 shown above can only be generated from the segmentation represented as arrows connecting prototypes to image columns. This is due to the fact that all but the rightmost prototype are $0-1$ column prototypes that can only emit themselves as binary columns. It is straightforward to check that, according to (5), the probability of generating this image of number 31 is 0.020412.

As with conventional HMMs, the exact probability (density) of an observation can be efficiently computing by dynamic programming. For each time t, symbol s_l and state j from the HMM for symbol s_l, define $\alpha_{lt}(j)$ as:

$$\alpha_{lt}(j) = p_{\mathbf{\Lambda}}(O_1^t, q_t = (l,j) \mid S) \,, \quad (6)$$

that is, the probability (density) of generating O up to its tth element and ending at state j from the HMM for symbol s_l. This definition includes (5) as the particular case in which $t = T$, $l = L$ and $j = F_{s_L}$; that is,

$$p_{\mathbf{\Lambda}}(O \mid S) = \alpha_{LT(F_{s_L})} \,. \quad (7)$$

To compute $\alpha_{LT(F_{s_L})}$, we must first take into account that, for each position l in S except for the first, the initial state of the HMM for s_l is joined with final state of its preceding HMM, i.e.

$$\alpha_{lt}(I_{s_l}) = \alpha_{l-1t}(F_{s_{l-1}}) \qquad \begin{matrix} 1 < l \leq L \\ 1 \leq t \leq T \end{matrix} \,. \quad (8)$$

Having (8) in mind, we can proceed at symbol level as with conventional HMMs. In the case of final states, we have:

$$\alpha_{lt}(F_{s_l}) = \sum_{j=1}^{M_{s_l}} \alpha_{lt}(j) a_{s_l j F_{s_l}} \qquad \begin{matrix} 1 \leq l \leq L \\ 1 \leq t \leq T \end{matrix} \,, \quad (9)$$

while, for regular states, $1 \leq j \leq M_{s_l}$, we have:

$$\alpha_{lt}(j) = \left[\sum_{i \in \{I_{s_l}, 1, \dots, M_{s_l}\}} \alpha_{lt-1}(i) a_{s_l ij} \right] b_{s_l j}(\mathbf{o}_t) \,, \quad (10)$$

with $1 \leq l \leq L$ and $1 < t \leq T$. The base case is for $t = 1$:

$$\alpha_{l1}(i) = \begin{cases} a_{s_1 I_{s_1} i}\, b_{s_1 i}(\mathbf{o}_1) & l = 1, 1 \leq i \leq M_{s_1} \\ 0 & \text{otherwise} \end{cases} . \quad (11)$$

5. Maximum likelihood estimation

Maximum likelihood estimation of the parameters governing an embedded Bernoulli mixture HMM does not differ significantly from the conventional Gaussian case, and it can be carried out using the well-known EM (Baum-Welch) re-estimation formulae [6, 8]. Let $(O_1, S_1), \ldots, (O_N, S_N)$, be a collection of N training samples in which the nth observation has length T_n, $O_n = (\mathbf{o}_{n1}, \ldots, \mathbf{o}_{nT_n})$, and was generated from a sequence of L_n symbols ($L_n \leq T_n$), $S_n = (s_{n1}, \ldots, s_{nL_n})$. At iteration r, the E step requires the computation, for each training sample n, of their corresponding forward and backward probabilities (see (6) and [6, 8]), as well as the expected value for its tth feature vector to be generated from kth component of the state j in the HMM for symbol s_l,

$$z^{(r)}_{nltk}(j) = \frac{\pi^{(r)}_{s_{nl}jk} \prod_{d=1}^{D} {p^{(r)}_{s_{nl}jkd}}^{o_{ntd}} \left(1 - p^{(r)}_{s_{nl}jkd}\right)^{1-o_{ntd}}}{b^{(r)}_{s_{nl}j}(\mathbf{o}_{nt})} ,$$

for each t, k, j and l.

In the M step, the Bernoulli prototype corresponding to the kth component of the state j in the HMM for character c has to be updated as:

$$\mathbf{p}^{(r+1)}_{cjk} = \frac{1}{\gamma_{ck}(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi^{(r)}_{nltk}(j)\mathbf{o}_{nt}}{P(O_n \mid S_n, \lambda_1^C)} , \quad (12)$$

where $\gamma_{ck}(j)$ is a normalisation factor,

$$\gamma_{ck}(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi^{(r)}_{nltk}(j)}{P(O_n \mid S_n, \lambda_1^C)} , \quad (13)$$

and $\xi^{(r)}_{nltk}(j)$ the probability for the tth feature vector of the nth sample, to be generated from the kth component of the state j in the HMM for symbol s_l,

$$\xi^{(r)}_{nltk}(j) = \alpha^{(r)}_{nlt}(j) z^{(r)}_{nltk}(j) \beta^{(r)}_{nlt}(j) . \quad (14)$$

Similarly, the kth component coefficient of the state j in the HMM for character c has to be updated as:

$$\pi^{(r+1)}_{cjk} = \frac{1}{\gamma_c(j)} \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \xi^{(r)}_{nltk}(j)}{P(O_n \mid S_n, \lambda_1^C)} , \quad (15)$$

where $\gamma_c(j)$ is a normalisation factor,

$$\gamma_c(j) = \sum_n \frac{\sum_{l:s_{nl}=c} \sum_{t=1}^{T_n} \alpha^{(r)}_{nlt}(j) \beta^{(r)}_{nlt}(j)}{P(O_n \mid S_n, \lambda_1^C)} . \quad (16)$$

To avoid null probabilities in Bernoulli prototypes, they can be smoothed by linear interpolation with a flat (uniform) prototype, $\mathbf{0.5}$,

$$\tilde{\mathbf{p}} = (1 - \delta)\,\mathbf{p} + \delta\,\mathbf{0.5} , \quad (17)$$

where, for instance, $\delta = 10^{-6}$.

6. Experiments

Experiments have been carried out using the IAM database [5]. This corpus contains forms of unconstrained handwritten English text. All texts were extracted from the LOB corpus. A total of 657 writers contributed. Different datasets were obtained by using segmentation techniques and, in particular, we have used the handwritten words dataset. More precisely, we have selected those samples in this dataset that are marked as correctly segmented in the corpus, and which belong to a word with at least 10 samples.

All input gray level images were preprocessed before transforming them into sequences of feature vectors. Preprocessing consisted of three steps: gray level normalisation, deslanting, and size normalisation of ascenders and descenders. See [4] for further details.

Selected samples were randomly splitted into 30 80%-20% training-test partitions at the writer level to ensure writer-independent testing. This means about 59000 samples for training and 14000 for testing. The lexicon comprises 1117 different words and the alphabet is composed by 71 characters. The reader is referred to [3] for sample images and previous results on a very similar task.

For the Bernoulli recogniser, feature extraction has been carried out by rescaling the image to height 30 while respecting the original aspect ratio, and applying an Otsu binarisation to the resulting image. Therefore, the observation sequence is in fact a binary image of height 30. In the Gaussian case, feature vectors are of dimension 60, where the first 20 values are gray levels, and the other 40 are horizontal and vertical gray level derivatives [4]. In this case, we used the well-known HTK software [8].

Experiments have been carried out by varying number of states, $Q \in \{4, 6, 8, 10, 12\}$, varying the number of mixture components per state $K \in \{1, 4, 16, 64\}$, and comparing our Bernoulli recogniser with a conventional recogniser based on (embedded) Gaussian mixture HMMs. For $K = 1$, both recognisers were initialised by first segmenting the training set using a "neutral" model, and then using the resulting segments to perform a Viterbi initialisation. For $K > 1$, both recognisers were initialised by splitting the mixture components of the trained model with $K/4$ mixture components per state. Models have been trained with 4 EM iterations, and the recognition has been performed using the Viterbi algorithm. Figure 3 shows the results for the

Gaussian and Bernoulli recognisers, where each point is the average of 30 repetitions (30 random splits). Due to computational costs, only 15 repetitions have been carried out for $K = 64$. Vertical bars denote $\pm$ standard deviation.

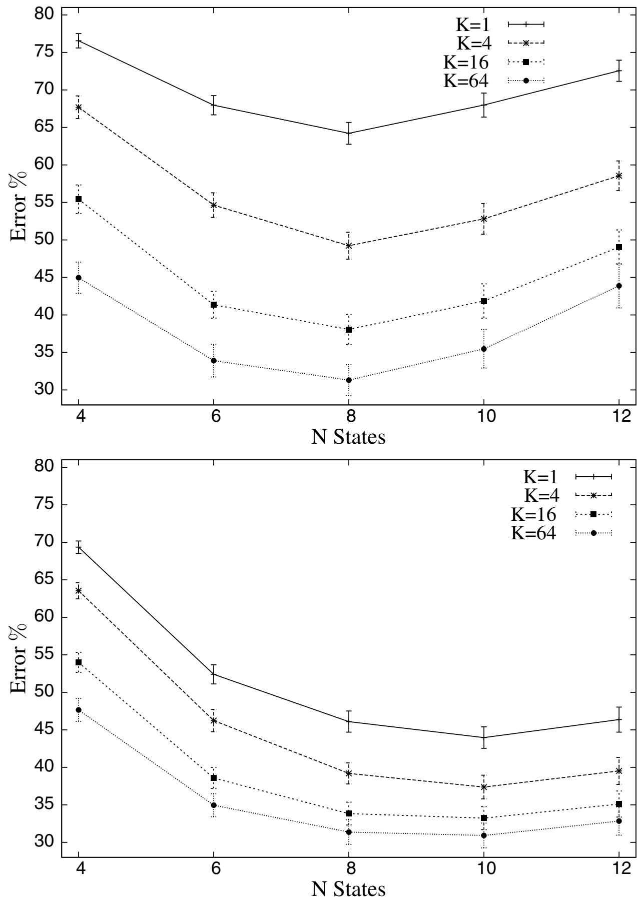

Figure 3. Classification error-rate (%) as a function of the number of states, for varying number of components (K). Top: Gaussian HMMs. Bottom: Bernoulli HMMs.

From the results in Figure 3, it becomes clear that Bernoulli mixture HMMs achieve similar or better results than those of the conventional Gaussian mixture HMMs. Moreover, Bernoulli HMMs have a more stable behaviour, as compared with Gaussian HMMs, in terms of flatter curves. However, as the value of K increases, the difference between the two recognisers decreases. The best results for them both are obtained with $K = 64$. In particular, the best result for the Bernoulli recogniser is an error rate of 30.9% (with $Q = 10$), while the Gaussian recogniser is slightly worse: 31.3% (with $Q = 8$). It must be noted that the optimal Bernoulli recogniser is much simpler than the optimal Gaussian recogniser: $1.4M$ and $4.4M$ parameters respectively. This relative simplicity may be very well be the cause of its apparently better performance.

7. Concluding remarks and future work

Embedded Bernoulli mixture HMMs have been proposed for handwritten word recognition. They have been formally described first, and then empirically compared with conventional Gaussian HMMs on a task of handwritten word classification from the IAM database. The results of the Bernoulli-based recogniser are similar or better than those of the Gaussian-based recogniser. Nevertheless, no feature extraction is required for the Bernoulli recogniser and, moreover, it is much simpler in terms of number of parameters.

For future work, we plan to use the ideas reported in [7] for explicitly modelling of invariances in Bernoulli mixtures, and to extend the experiments to general handwritten text recognition.

References

[1] A. Giménez and A. Juan. Bernoulli HMMs at Subword Level for Handwritten Word Recognition. In *Proc. of the 4th Iberian Conference on Pattern Recognition and Image Analysis*, Póvoa de Varzim (Portugal), June 2009. (submitted).

[2] A. Giménez-Pastor and A. Juan-Císcar. Bernoulli HMMs for Off-line Handwriting Recognition. In *Proc. of the 8th Int. Workshop on Pattern Recognition in Information Systems (PRIS 2008)*, pages 86–91, Barcelona (Spain), June 2008.

[3] S. Günter and H. Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

[4] M. P. i Gadea. *Aportaciones al reconocimiento automático de texto manuscrito*. PhD thesis, Dep. de Sistemes Informàtics i Computació, València, Spain, Oct 2007. Advisors: E. Vidal and A.H. Tosselli.

[5] U. Marti and H. Bunke. The IAM-database: an English sentence database for offline handwriting recognition. 5(1):39–46, 2002.

[6] L. Rabiner and B.-H. Juang. *Fundamentals of speech recognition*. Prentice-Hall, 1993.

[7] V. Romero, A. Giménez, and A. Juan. Explicit Modelling of Invariances in Bernoulli Mixtures for Binary Images. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4477 of *LNCS*, pages 539–546. Springer-Verlag, Girona (Spain), June 2007.

[8] S. Young et al. *The HTK Book*. Cambridge University Engineering Department, 1995.