Offline Cursive Word Recognition using Continuous Density Hidden Markov Models trained with PCA or ICA Features

A.Vinciarelli and S.Bengio
IDIAP - Institut Dalle Molle d'Intelligence Artificielle Perceptive
Rue du Simplon 4, CP592 - 1920 Martigny, Switzerland
{vincia,bengio}@idiap.ch

Abstract

This work presents an Offline Cursive Word Recognition System dealing with single writer samples. The system is based on a continuous density Hiddden Markov Model trained using either the raw data, or data transformed using Principal Component Analysis or Independent Component Analysis. Both techniques significantly improved the recognition rate of the system.
Preprocessing, normalization and feature extraction are described as well as the training technique adopted. Several experiments were performed using a publicly available database. The accuracy obtained is the highest presented in the literature over the same data.

1. Introduction

This work presents a system for offline single writer Cursive Word Recognition (CWR). The system is based on a sliding window approach: a window shifts column by column across the image and, at each step, isolates a frame. A feature vector is extracted from each frame and the sequence of frames so obtained is modeled with Continuous Density Hidden Markov Models (HMMs). The use of the sliding window approach has the important advantage of avoiding the need of an independent segmentation, a difficult and error prone process.
In order to reduce the number of parameters in the HMMs, we use diagonal covariance matrices in the emission probabilities. This corresponds to the unrealistic assumption of having decorrelated feature vectors. For this reason, we applied Principal Component Analysis (PCA) and Independent Component Analysis (ICA) to decorrelate the data. This allowed a significant improvement of the recognition rate.
Several experiments were performed on a publicly available database. The results we obtained are, to our knowledge, the best among those presented in the literature over the same data.
The paper is organized as follows: section 2 presents preprocessing, normalization and feature extraction, section 3 introduces the Hidden Markov Models, section 4 describes PCA and ICA, section 5 shows the results obtained and the final section 6 draws some conclusions.

2. Preprocessing, Normalization and Feature Extraction

The aim of preprocessing is the removal of all the elements in the word image that are not useful for the recognition. The operations performed at this stage depend on the data. In our case, a binarization (performed with the Otsu algorithm [1]) is sufficient.
The normalization is supposed to remove slant (the angle between the vertical direction and the direction of the strokes supposed to be vertical in an ideal model of handwriting) and slope (the angle between the horizontal direction and the direction on which the word is aligned). The slope removal technique estimates the distribution of the horizontal density values to identify the core region of the sample. The stroke minima closest to the lower limit of the estimated core region are used to fit the baseline of the word. The image is finally rotated until the baseline is horizontal (and the image is desloped).
The slant removal technique estimates the *deslantedness* of the word with a measure based on the horizontal projection of the sample. Several shear transforms are performed (corresponding to the angles of a reasonable interval) and the shear transformed image giving the highest value of deslantedness is assumed as the deslanted one. For a full description of the normalization technique we used, see [7].
The slope and slant removal methods applied are adaptive and do not use any parameters to be set empirically. This avoids the need of tuning a different parameter set for each writer and makes the system flexible with respect to a

change of data.
The sliding window blindly isolates the patterns from which the feature vectors are extracted. For this reason a feature extraction based on local averaging rather than on exact reconstruction of the patterns is selected. The window is 16 pixels wide and as high as the isolated pattern. At the resolution of our images (300 dpi), this corresponds to $\sim$1.5 mm, a dimension comparable with the width of the strokes produced by a common pen. The isolated frame is partitioned into cells regularly arranged in a 4×4 grid. The number n_i of foreground pixels is computed in each cell i. The feature vector collects the values $f_i = \frac{n_i}{\sum_j n_j}$.

3. Hidden Markov Models

Hidden Markov Models are probability density functions over sequences of vectors (for a good introduction see [4]). The sequences are assumed to be produced by a system characterized by a state (belonging to a finite set of possible states $\mathbf{Q} = \{Q_i : i = 1, 2, \ldots, N\}$) that changes at discrete time steps. The evolution of the system can be represented by the sequence of states $\{q_0, q_1, \ldots, q_T\}$. At each time step t, an observation vector $\mathbf{o}$ is emitted with probability $b_{q_t}(\mathbf{o}) = p(\mathbf{o}|q_t)$.
In the case of handwriting, the states are the letters (or parts of them) composing the words (that cannot be observed directly, thus the name *Hidden*) and the observations are the vectors extracted from the word image.
The state variable at time t can be though of as a stochastic variable. Two symplifying assumptions are made about its distribution. The first one (called *first order assumption*) is that the transition probability $a_{q_t q_{t-1}}$ from state q_{t-1} to state q_t depends only on q_{t-1}. The second one (called *stationarity assumption*) is that $a_{q_t q_{t-1}}$ do not depend on t. The two assumptions can be expressed as follows: $a_{q_t q_{t-1}} = p(q_t|q_{t-1})$. This allows one to describe the dynamics of the system with a transition matrix $A = \{a_{ij}\}$. The state variable at time $t = 0$ cannot be expressed in terms of transition probabilities, hence an initial state probability distribution must be provided: $\pi = \{\pi_i = p(q_0 = i) : i = 1, 2, \ldots, N\}$.
If we consider the set of the emission probabilities $B = \{b_{Q_1}(\mathbf{o}), b_{Q_2}(\mathbf{o}), \ldots, b_{Q_N}(\mathbf{o})\}$, a Hidden Markov Model is represented by the triple $\lambda = (A, B, \pi)$.
When the observations are continuous, the HMMs are said Continuous Density Hidden Markov Models and the b_i's are often represented as Mixtures of Gaussians. In order to reduce the number of parameters, the Gaussians have diagonal covariance matrix.
The parameters λ (transition probabilities, initial state probabilities, means, covariances and weights of the Gaussians) are typically adjusted to maximize the probability of generating the data at disposition (the training set): $\lambda^* = \arg\max_\lambda \prod_i p(\mathbf{O}_i|\lambda)$. This is called Maximum Likelihood training and it is performed with the Baum-Welch Algorithm, a specific case of the Expectation-Maximization technique.
Once a model for each word in the lexicon is trained, the likelihood of an observation sequence can be estimated using EM. In practice, to make faster the system, we approximated it by applying the Viterbi Algorithm. This finds the alignment with the highest likelihood value which is a good approximation of the probability of the data being generated by the model. The word corresponding to the model giving the highest probability is assumed as transcription.

4. Principal Components Analysis and Independent Component Analysis

Dy decorrelating the data it is possible to use diagonal covariance matrices in the mixtures of Gaussians (see section 1). In the most simple case, this can be done through a linear transform: $\mathbf{y} = W\mathbf{x}$, where $\mathbf{x}$ is the original vector, $\mathbf{y}$ the transformed one and W is a $k \times d$ matrix. When $k = d$ the transform simply corresponds to a change of reference frame, when $k < d$, the transform projects the data onto a subset of the original d-dimensional space. In the second case, the transform determines an information loss. The value of k must be selected as a trade-off between the disadvantage due to information loss and the advantage due to dimensionality reduction.
The criterion used to estimate the elements of W determines the properties of $\mathbf{y}$. We used two different criteria leading to the extraction of the Principal Components (using PCA) and of the Independent Components (using ICA).
When performing PCA [2], the rows of W are the eigenvectors of the covariance matrix of the original data (assumed to have 0 mean, condition that can always be easily achieved by substracting the mean estimated over the training set). The principal component z_j of a vector $\mathbf{x}$ corresponds to its projection along the direction of the j^{th} eigenvector of the data covariance matrix. The eigenvectors are ordered so that, if σ_i^2 is the eigenvalue related to the i^{th} eigenvector, then $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_d^2$. The eigenvalue accounts for the data variance along the direction of the corresponding eigenvector. The first eigenvector indicates the direction of highest variance in the data, the second one the direction orthogonal to the first one with the highest variance and so on. Often the last eigenvectors account for very small variance and the corresponding principal components can be eliminated. This allows a reduction of the dimensionality making the HMMs faster and better trainable (the number of parameters is reduced as it depends on the size of the observation vector).
A linear transform $\mathbf{y} = W\mathbf{x}$ can be used also to obtain the Independent Components [2]. In order to estimate the W

elements, it is enough to assume that the y_i are not Gaussian and that they are statistically independent ($p(y_i, y_j) = p(y_i)p(y_j)$ for $i \neq j$). The nongaussianity is necessary because, if the y_i are Gaussian, the matrix W can be identified only up to an orthogonal transform. Such ambiguity must be avoided. Moreover, the nongaussianity is the criterion used to find in practice the directions of the Independent Components.

A good measure of the nongaussianity is the negentropy: $J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y})$, where $\mathbf{y}_{gauss}$ is a gaussian variable with the same covariance matrix and mean as $\mathbf{y}$ and H is the entropy: $H(\mathbf{y}) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}$. A very important property of the negentropy is that, if we let the y_i being uncorrelated (condition desirable in our case), it is related to the mutual information, the information theoretic measure of the statistical independence: $I(y_1, y_2, \ldots, y_d) = J(\mathbf{y}) - \sum_i J(y_i)$. This allows one to look for the direction of the independent components as the directions of maximum negentropy. The number of Independent Components can be less than the number of components in the original vector leading to a dimensionality reduction.

With both PCA and ICA, it is possible to transform the original vectors into uncorrelated data with lower dimensionality. Both aspects are very important to reduce the number of parameters in the HMMs. This makes possible to train them more reliably given the same amount of training data.

5. Experiments and Results

The experiments were performed over a publicly available database[1] composed of 4053 words written by a single person. The words belong to the transcription of a text extracted from the LOB corpus, a collection of texts supposed to be representative of the average english. The dataset was split into three subsets with a random process: training set (2362 words), validation set (675 words) and test set (1016 set).

A different HMM is created for each letter. This makes the system flexible with respect to a change of lexicon because it allows to build the word models as concatenations of letter models. In this way, it is sufficient to have in the training set samples of the letters composing the words to be modeled rather than samples of the words themselves. Moreover, the number of parameters is kept lower because the word models share the parameters belonging to the same letters. This allows a better training given the same amount of training data.

The Baum-Welch algorithm (see section 3) is not appleded directly to the letter models, but to their concatenations corresponding to the words in the training set. This is called *embedded training* and has two important advantages: the first one is that the letters are modeled when being part of a word (that is the actual situation of the letters in the cursive handwriting), the second one is that it is not necessary to segment the words into letters to perform the training.

For simplicity, the number of states S and Gaussians G in the mixtures is the same for every letter model. The optimal values of S and G are selected through cross-validation: all the systems corresponding to couples (S, G) falling in a range determined by the amount of training data are trained and tested. The system giving the highest recognition rate on the validation set is retained as optimal. Such system is retrained over the union of training and validation set and is tested over the test set giving a final measure of the recognition rate. This procedure was applied using as data not only the raw feature vectors, but also alternatively the Principal Components and the Independent components extracted from them with PCA and ICA respectively. The results obtained are shown in details in the next section.

5.1. Recognition Results

The first system was obtained using the raw feature vectors. Systems with $5 \leq S \leq 15$ and $1 \leq G \leq 14$ were trained over the training set and tested over the validation set. The best accuracy was obtained by the system with $S = 11$ and $G = 12$. It was then retrained over the set composed of the training and of the validation set and tested over the test set. The recognition rate obtained (the lexicon size is 1374) is $\mathbf{92}.4\%$.

For PCA data, systems with $5 \leq S \leq 11$, $6 \leq G \leq 15$ and $13 \leq P \leq 16$ (P is the number of retained Principal Components) were trained and tested. The best result was obtained by a system with $S = 9$, $G = 13$ and $P = 16$ that gives an accuracy (after having been retrained on the union of validation and training set) of $\mathbf{94}.\mathbf{7}\%$ over the test set (corresponding to a reduction of 30.3% of the error rate with respect to the system using the raw data).

Finally, for ICA data, the explored range of parameters was as follows: $7 \leq S \leq 13$, $5 \leq G \leq 15$ and $13 \leq I \leq 16$. The best system ($S = 11$, $G = 14$, $I = 14$) was trained again over the union of the training and validation set and tested over the test set giving an accuracy of $\mathbf{93}.\mathbf{6}\%$ (corresponding to a reduction of 15.8% of the error rate with respect to the system using the raw data).

The best system was obtained by training over the Principal Components. Its performance over the data set used is significantly higher than the accuracies claimed (over the same data) in [5, 3] (92.8% and 85.0% respectively) and slightly better than the 94.6% recognition rate presented in [6]. On the other hand, this last system is much more complex than our. The words are first segmented into primitives using a sliding window approach. A Neural Network is then used

[1] The data can be downloaded at the following ftp address: `ftp://ftp.eng.cam.ac.uk/pub/data`.

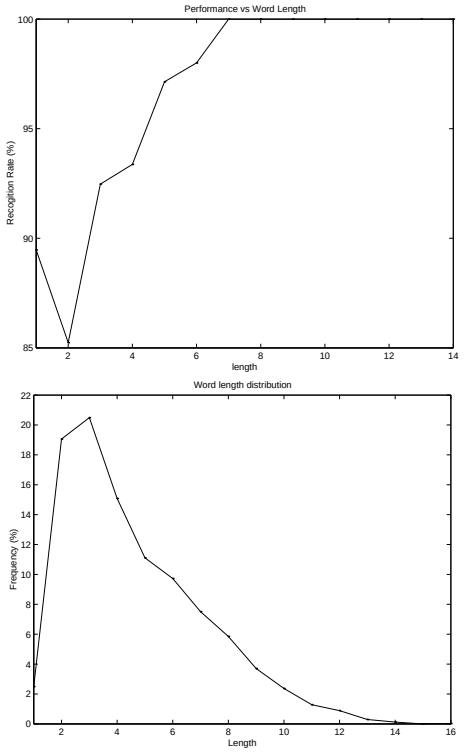

Figure 1. The upper plot shows the performance as a function of the word length. The lower plot shows the length distribution in the database.

to classifiy the primitives and their aggregates and, through a Dynamic Programming algorithm, the best segmentation into letters is found.

Figure 1 shows the recognition rate as a function of the word lenght. The system achieves a 100% accuracy for samples longer than six letters. The low recognition rate for short words has several causes. First, when there are few letters, the misalignemnt of a single letter model with the corresponding letter in the word can have a strong influence. When there are more letters, an eventual misalignement can be equilibrated by other letters. A second problem is that the normalization scheme is conceived to work on long words (more than 3 letters) and produces sometimes bad results over short samples. For these reasons, the performance of the system can be better on data sets having a length distribution with more weigth on the longer samples.

6 Conclusions

This work presented a system for the offline recognition of cursive words written by a single person. All the processing steps were briefly described. Moreover, the application of Principal Component Analysis and Independent Component Analysis was investigated. Several experiments were performed on a publicly available database. The recognition accuracy achieved with the approach proposed here is, to our knowledge, the highest among the results over the same data presented in the literature. The analysis of the recognition as a function of the word length shows that the system achieves a 100% recognition rate for samples longer than six letters. This suggests that the performance of our system in tasks involving words with high average length can be very good.

Both PCA and ICA had a positive effect on the recognition rate, PCA in particular reduced the error rate, with respect to the use of raw data, by 30.3%. A further improvement can probably be obtained by using *nonlinear* or *kernel* PCA. Such techniques often work better than the linear transform we used to perform PCA.

The use of data dependent heuristics was avoided in order to make the system flexible with respect to a change of writer. Any ad-hoc algorithm for the specific style of the writer was avoided.

The prior information about the word frequency and distribution can be useful to improve the recognition of short words. These are typically articles, conjunctions and propositions that appear often in the sentences. For this reason, a possible future direction to follow is the application of language models that take into account this kind of information.

Acknowledgements This work was done under the grant no. 21-55733.98 issued by Swiss National Science Foundation.

References

[1] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison Wesley, USA, 1992.

[2] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley and Sons, 2001.

[3] U. Marti and H. Bunke. Towards general cursive script recognition. In *Proc. of IWFHR*, pages 379–388, Korea, 1998.

[4] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann, 1989.

[5] A. W. Senior and A. J. Robinson. An off-line cursive handwriting recognition system. *IEEE Trans. on Patt. An. and Mac. Int.*, 20(3):309–321, March 1998.

[6] Y. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, and S. Knerr. An offline cursive handwritten word recognition system. In *Proceedings of IEEE Region 10 Conference*, 2001.

[7] A. Vinciarelli and J. Lüttin. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22:1043–1050, 2001.