IDIAP RESEARCH REPORT

Transforming the feature vectors to improve HMM based cursive word recognition systems

Alessandro Vinciarelli [a] Samy Bengio [a]

IDIAP–RR 02-32

August 2002

submitted for publication

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a] IDIAP, {vincia,bengio}@idiap.ch

IDIAP Research Report 02-32

Transforming the feature vectors to improve HMM based cursive word recognition systems

Alessandro Vinciarelli Samy Bengio

August 2002

submitted for publication

Abstract. Although many Offline Cursive Word Recognition systems are based on HMMs, no attention was ever paid, to our knowledge, to the fact that the feature vectors are typically not in the most suitable form for modeling. They are most of the time correlated and embedded in a space of dimension higher than their Intrinsic Dimension. This leads to several problems and has a negative influence on the performance.
By applying some transforms it is possible to solve, or at least to attenuate, such problems resulting in data easier to model and in systems with higher recognition rate. In this work, we used Principal Component Analysis (linear and nonlinear) and Independent Component Analysis. A reduction of the error rate by up to 30.3% (over single writer data) and 16.2% (over multiple writer samples) is shown to be achieved.

1 Introduction

Hidden Markov Models are widely applied in Offline Cursive Word Recognition [1][2][3]. The systems using such approach (see e.g. [4][5][6][7][8][9][10]) convert the handwritten data into vector sequences by splitting them into fragments and by extracting from each one of these a feature vector. The sequence of observations so obtained is modeled with the HMMs that are probability density functions over the space of the vector sequences [11].
Although such an approach is common, no systematic attention was ever paid, to our knowledge, to the possibility of improving the performance of the HMMs by giving the feature vectors a form that is more suitable for the modeling. The raw vectors are typically affected by two fundamental problems: they are embedded in a space that has a dimension d higher than their *Intrinsic Dimension* (ID) and they are correlated (see section 2 for a description of both points). Such problems can be solved (or at least attenuated) by applying transforms that can compress and decorrelate the data.
In this work, we used Karhunen-Loeve Expansion, Nonlinear Principal Component Analysis (based on autoassociative neural networks) and Independent Component Analysis. The first two techniques are a linear and nonlinear version respectively of Principal Component Analysis. The last one is a transform that tries to model the data as a linear combination of statistically independent stochastic variables. Depending on the transform, the attention will be focused on compression or decorrelation, however all the transforms are shown to improve significantly the recognition rate of a baseline system using raw feature vectors. A maximum reduction of the error rate by 30.3% and by 16.2% was achieved over single and multiple writer data respectively.
The remaining of the paper is organized as follows: section 2 describes the disadvantages of using raw data, section 3 presents the data transforms applied, section 4 gives a brief overview of the system used, section 5 describes experiments and results obtained, section 6 draws some conclusions and final remarks.

2 Raw Data Disadvantages

This section explains why compression and decorrelation (that can be obtained through the transforms presented in this work) of the feature vectors can improve the performance of the HMMs.
The ID of a data set is the minimum number of free variables needed to represent the data without information loss. In other terms, a data set $\Omega \subset \mathbb{R}^d$ is said to have an ID equal to m if its elements lie entirely within an m-dimensional subspace of $\mathbb{R}^d$ (where $m < d$) [12]. The use of more dimensions than necessary leads to two main problems [13]: *Curse of Dimensionality* and increase of the number of parameters in the HMMs.
Curse of Dimensionality is a phenomenon that can be explained through a simple example [14][15]. Consider a space partitioned into regularly arranged cells: by increasing the dimension, the number of cells increases exponentially. A modeling problem consists essentially of estimating the distribution of the data and this can be done reliably only if the space occupied by the data is well sampled. If the number of cells increases exponentially, the number of data points necessary for a good sampling increases itself exponentially. The use of too many dimensions can make a given amount of training data less effective or even insufficient for the modeling. A reduction of the dimension is then necessary.
Another aspect of the problem is the number of parameters in the HMMs. This is related to the dimension of the observation vectors. By reducing their dimension, the number of parameters can be significantly reduced. This is important because the amount of available training material limits the number of parameters that can be reliably trained.
The second important problem is the correlatedness. The data is said correlated when its covariance matrix:

$$C_{\mathbf{x}} = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)] \quad (1)$$

(where $\mu = E[\mathbf{x}]$) is full. When continuous density HMMs are used, the emission probabilities are modeled as Gaussian mixtures and their covariance matrices should be correspondently full. Because a full covariance matrix has too many parameters (d^2 if d is the dimension of the vectors), it is common to use diagonal matrices (having only d nonzero elements). This still allows one to model correlated data, but requires more Gaussians than when the data is uncorrelated. By decorrelating the data, it is possible to reduce the number of necessary Gaussians leading, for a fixed amount of training material, to a more effective training of their parameters.

3 Data Transforms

For the reasons explained in the previous section, it is desirable to have observation vectors with dimension as close as possible to their ID and uncorrelated. Such conditions can be achieved by projecting (either in a linear or a nonlinear way) onto an appropriate subspace of dimension m in the original data space of dimension d ($m \leq d$).
The literature presents several methods to perform such task (see [13] for a deep analysis of the problem). In this work, we applied Principal Component Analysis (PCA) and Independent Component Analysis (ICA). They have two important advantages: the first one is that they are unsupervised. In this way, no labeling of the data (an operation that can be heavily time consuming in handwriting recognition) is required. The second one is that they are based on very general assumptions about the data and they can thus be applied to many different problems [13][15].
In both cases, the transforms are characterized by a set of parameters that can be estimated using a set T of training vectors. This is composed of all the feature vectors extracted from the words used to train the Hidden Markov Models.
In the next three subsections, the applied algorithms are explained in detail. Subsection 3.1 describes the Karhunen-Loeve Expansion (a linear version of the PCA), subsection 3.2 presents a nonlinear version of the PCA based on autoassociative neural networks and subsection 3.3 explains ICA.

3.1 Karhunen-Loeve Expansion

The KLE is based on the following linear transform:

$$\mathbf{y} = A\mathbf{x} \tag{2}$$

where A is a $d \times d$ matrix, $\mathbf{y}$ is the transformed vector and $\mathbf{x}$ is the original vector assumed to have $E[\mathbf{x}] = \mathbf{0}$. The elements of A are estimated so that, when only the first m components of $\mathbf{y}$ are retained, the Mean Squared Error (MSE):

$$E(m) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i^m - A_m\mathbf{x}_i)^2 \tag{3}$$

is minimized ($\mathbf{y}_i^m$ is the vector of the first m components of $\mathbf{y}_i$, A_m is the matrix composed of the first m columns of A and N is the total number of vectors in the training set). It can be demonstrated that the resulting matrix A is the one having as columns the eigenvectors of the covariance matrix $\Sigma = E[\mathbf{x}\mathbf{x}^T]$ ordered following the respective eigenvalues (from the biggest to the smallest). A component y_j of $\mathbf{y}$ corresponds to the projection of $\mathbf{x}$ along the j^{th} eigenvector of the covariance matrix. The y_j's are called *Principal Components* (hence the name PCA). Each eigenvalue σ_j^2 accounts for the data variance along the direction individuated by the corresponding eigenvector. For this reason, the subspace spanned by the first m columns of A accounts for more variance than any other subspace spanned by a different set of m orthogonal axes [15]. This is an important property. In the hypothesis that the variance of the data is given by information useful for a certain task (and not by noise), the subspace of the first m eigenvectors is the one containing more information than any other with the same dimension.
The MSE, when retaining only m Principal Components (see equation 3), can also be estimated as

follows:

$$E(m) = \sum_{j=m+1}^{d} \sigma_j^2. \tag{4}$$

If the last eigenvectors are sufficiently small, the error in (4) is negligible and the original data can be compressed without a significant loss of information. This allows one to reduce the dimensionality of the vectors with the advantages described in section 2. Although the KLE is not a good estimator of the ID (in the sense that the m for which the error in (4) is negligible is typically higher than the actual ID of the data), a significant compression can often be achieved.
When $m = d$, the error in equation (3) is null and the transform corresponds to a change of reference frame. The resulting data is uncorrelated and, although no compression is achieved, the performance of the HMMs can still be improved. In fact, the transformed data has diagonal covariance matrix and can be modeled more effectively than data having full covariance matrix given a certain number of Gaussians (see section 2).
Some systems presented in the literature (e.g. [16][17]) used KLE to compress the vectors, but no clear explanation of the criteria used to set the number of retained components was given. Moreover, no comparison was made with respect to a baseline system.
The fundamental limit of the KLE is the linearity of the transform in equation (2). When the data is distributed over nonlinear subspaces, the projection onto a linear subspace can be a rough approximation. Possible solutions are to use a nonlinear transform, or to map the data into a space (eventually with higher dimension) where the nonlinear surfaces are hyperplans and then to apply a KLE in such a new space. This last approach is used with the autoassociative neural networks described in the next section.

3.2 Nonlinear Principal Component Analysis

Many pattern recognition problems can be solved with Neural Networks that (even if based on different underlying principles) are found to be implicitly equivalent or similar to statistical methods usually applied to solve them [18][19][20]. The main advantage is that different problems can be solved using a single theoretic framework without the need of a deep domain specific knowledge [21].
A KLE can be obtained using an autoassociative (i.e. having as output the input pattern) Multi Layer Perceptron [14] with linear activation functions in the neurons. The training criterion must be the minimization of the MSE. The hidden layer of such a network encodes the KLE of the input data [22][23]. In order to discard the less informative Principal Components, it is necessary to estimate the variance along the different components of the transformed vectors. Another possibility is to train several networks with different size of the central layer.
By changing the architecture of the network it is possible to perform a *Nonlinear Principal Component Analysis* (NLPCA). By nonlinear it is meant that the data can be projected onto nonlinear subspaces (e.g. curve surfaces). This is an advantage when the original data is distributed over nonlinear structures and the projection onto linear subspaces (like in KLE) might lead to a high approximation error [24]. This allows one to overcome the main limit of the KLE (see previous subsection). NLPCA was applied to many domains [13], but it was never used, to our knowledge, in Offline Cursive Handwriting Recognition.
The network performing NLPCA has five layers (see figure 1). The activation functions are linear in the input, output and central layers and nonlinear in the others [14][15]. This is often called a *bottleneck network* because it forces the data to pass through the central layer that has a number of neurons less or equal to the dimension of the space the input vector is embedded in. The network is autoassociative and the training criterion is the minimization of the Mean Squared Error (see subsection 3.1).
The first nonlinear layer (see figure 1) performs a mapping of the original data into a space (called *feature space*) where nonlinear structures are transformed into linear ones. The central layer projects the mapped data onto a linear subspace of the feature space. The second nonlinear layer performs

the inverse mapping of the first one and the final layer outputs the input pattern.
The NLPCA is encoded by the central layer. The subspace spanned by its output accounts, given the number of its neurons, for the biggest amount of variance of the input data. The information loss is then minimized when the original data is compressed [25]. Unlike in KLE, the uncorrelatedness of the transformed data is not guaranteed [13]. This can create problems when using continuous density HMMs (see section 2).
A further problem is given by the fact that the mapping of the data onto the subspace spanned by the Principal Components is continuous. This creates problems when the original data is distributed over a surface that is discontinuous or self-intersecting. In correspondence of the discontinuities or of the point of self-intersection the results are unreliable [25].

3.3 Independent Component Analysis

The Independent Component Analysis can be defined using a statistical latent variable model. The components of the data vectors $\mathbf{x}$ can be thought of as linear combinations of m stochastic variables s_j that cannot be observed directly:

$$x_i = w_{i1}s_1 + w_{i2}s_2 + \ldots + w_{im}s_m \tag{5}$$

where the w_{ij} (with $i = 1, \ldots, d$ and $j =, 1, \ldots, m$) are some real coefficients. This is the basic ICA model, the w_{ij} coefficients are not known and must be estimated (under assumptions as general as possible) using only the data vectors in T.
This corresponds to estimate the parameters of a linear transform:

$$\mathbf{x} = W\mathbf{s} \tag{6}$$

making the problem similar to the KLE one (see subsection 3.1). There are only two hypotheses that must be assumed to make sure that the w_{ij} can be estimated. The first one is that the s_j are statistically independent, the second one is that their distribution is not Gaussian [26][27][28]. Two variables s_i and s_j are said statistically independent when $p(s_i, s_j) = p(s_i)p(s_j)$. The nongaussianity is necessary because, when the s_i are Gaussians and statistically independent, the transform W can be identified only up to an orthogonal transform and this ambiguity must be avoided.
The above assumptions are very general and, for this reason, ICA can be applied to a wide spectrum of problems. On the other hand, the generality causes some ambiguities that cannot be avoided. It is not possible to determine the variance of the Independent Components. The reason is that any scalar multiplier in one of the sources s_i can be canceled by dividing the corresponding column in W by the same factor [27].
From a practical point of view, a measure of nongaussianity, the *negentropy*, is the basis of the criterion used to find the Independent Components [29][30]. The negentropy of a variable $\mathbf{y}$ is defined as follows:

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \tag{7}$$

where $\mathbf{y}_{gauss}$ is a Gaussian variable with the same covariance matrix and mean as $\mathbf{y}$ and H is the entropy:

$$H(\mathbf{y}) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}. \tag{8}$$

An important advantage of $J(\mathbf{y})$ is that, if the y_i are forced to be uncorrelated (which is in our case a desirable condition), it is related to their mutual information:

$$I(y_1, y_2, \ldots, y_d) = J(\mathbf{y}) - \sum_i J(y_i). \tag{9}$$

The mutual information is the information theoretic measure of the independence of the variables. The Independent Components can then be obtained by finding the directions of maximum negentropy.

This transforms the ICA problem into a numerical optimization problem. Since the negentropy is very heavy to compute, several approximations of it were proposed allowing a fast and reliable estimation of the independent components [29][31].

The ICA is helpful for both compressing and decorrelating the data. The statistical independence implies the uncorrelatedness and this is helpful to the HMM performance (see section 2). The compression can be achieved by setting $m < d$ in equation (5). A survey on the applications of ICA can be found in [27]. Although the algorithm was used in a wide variety of problems, no handwriting recognition system appears to make use of it.

4 The Recognition System

This section presents the recognition system used in the experiments (for a full description see [32]). The raw word images are first binarized with the Otsu algorithm [33], then slope (the angle between the horizontal direction and the direction of the line the word is aligned on) and slant (the angle between the vertical direction and the direction of the strokes supposed to be vertical in an ideal model of handwriting) are removed with the technique described in [34].

The system is based on a sliding window approach. A window shifts column by column from left to right and, at each position, a pattern is isolated. A feature vector is extracted from each pattern and the word is so converted into a sequence of observations. The feature extraction process consists of partitioning the pattern into 16 cells regularly arranged in a 4×4 grid and of counting the number n_i of foreground pixels in each cell. The feature vector is then obtained as follows:

$$\mathbf{f} = (\frac{n_1}{\sum_i n_i}, \frac{n_2}{\sum_i n_i}, \ldots, \frac{n_{16}}{\sum_i n_i}). \tag{10}$$

The vector sequences so obtained are modeled with Continuous Density Hidden Markov Models [11]. A different left-right HMM is created for each letter and the word models corresponding to the lexicon entries are built by concatenating the single letter models. This makes the system flexible with respect to a change of lexicon because it is not necessary to have samples of the lexicon words in the training set. The only important thing is to have in the training set samples of all the letters (a condition easy to achieve).

Each letter model is characterized by the number of states S and by the number of Gaussians G in each state. For simplicity, G and S are the same for every letter models. The training of the models is performed with the Baum-Welch algorithm [11][35] and it is embedded. This means that the Baum-Welch algorithm is applied to the word models (built by concatenating the single letter models) rather than to the single letter models. This has two advantages, the first one is that it is not necessary to segment the words into letters, the second one is that the letters are modeled as a part of the words, i.e. their actual condition in cursive handwriting.

The recognition is performed using the Viterbi algorithm [35][36]. This gives the best likelihood λ that can be obtained by following a unique state sequence in the word model with the vectors extracted from the handwritten data. The entry of the lexicon corresponding to the model giving the highest value of λ is selected as transcription of the handwritten word.

5 Experiments And Results

The experiments were performed over two different data sets. The first is publicly available [1] and it is composed of 4053 words produced by a single person. The database is split into training (2362 words), validation (675 words) and test set (1016 words). It was originally presented in [9] and will be referred to as *Cambridge* database. The second is a collection of 12199 samples (split into training, validation and test set containing 5347, 2715 and 4137 words respectively) written by around 200 persons. The

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

samples were extracted from a database of handwritten pages collected at University of Bern [37] and will be referred to as *Bern* database. The word length distribution of the two sets is shown in figure 2. In the next three subsections, the estimation of the transform parameters, the results over the Cambridge database and the results over the Bern database are respectively presented.

5.1 Transform Parameters Estimation

In the case of KLE, the raw data is transformed through the matrix A (see above). This is obtained by estimating the eigenvectors of the covariance matrix of the vectors extracted from the words used to train the HMMs.
The word recognition experiments involve a validation and a test (see next subsections). For the validation, the matrix A is obtained using the vectors extracted from the words of the training set. For the test, A is obtained using the vectors extracted from the union of training and validation set.
In the case of ICA, the approach used is the same as in the case of KLE, but it must be repeated for each number I of Independent Components. As explained above, the Independent Components cannot be ordered following some criteria, hence the only way to achieve a data compression is to estimate a different transform for each value of I.
A similar approach is followed in the case of the NLPCA with the difference that there is a parameter to set: the number N of neurons in the second and fourth layers. An optimal N is selected for each number of Principal Components (neurons in the central layer) P. For a given P, several networks (with different N) are trained over the training set and tested over the validation set. The network with the smallest N having an MSE lower than a fixed threshold is retained as optimal. The value of the threshold is fixed so that the absolute error per component is 0.01. Our features are percentages (see section 4) obtained from sets containing typically less than 1000 elements. For this reason, the third digit after the comma is very noisy and it is not worth estimating it precisely.

5.2 Experiments Over The Cambridge Database

Our system is characterized by the number of states S and Gaussians G per state in its models. For simplicity, S and G are the same for all the letter models. The parameters S and G are selected as follows: models with a number of states S ranging from 8 to 12 and number of Gaussians G between 10 an 15 are trained over the training set and tested over the validation set. Both the ranges of S and G were determined by the available training material. The system showing the best performance was retained as optimal, retrained over the union of training and validation set and finally tested over the test set. This allows one to set the values of S and G by measuring the performance over a set (the validation set) that is independent of the test set. In this way S and G are not fitted to the test set and the performance of the corresponding system is not overestimated. The lexicon size is 1370.
The baseline system was obtained using the raw feature vectors. The recognition rate of the optimal system ($S = 11$ and $G = 12$) over the test set is **92.4%**.
When using KLE, NLPCA and ICA, the parameters to be set through validation are not only S and G, but also the number of retained Principal Components P or Independent Components I that ranged between 13 and 16. The best system based on KLE is obtained with $P = 16$ (using models having $S = 9$ and $G = 13$) and its recognition rate (measured over the test set) is **94.7%**. When using NLPCA, the performance over the test set is **94.0%** (the optimal P, S and G are 14, 9 and 12 respectively). The best system using ICA has $S = 11$, $G = 14$ and $I = 14$. Its recognition rate over the test set is **93.6%**. For both PCA and NLPCA, the result corresponds to an improvement of the baseline system with a probability higher than 90%. The difference in performance between PCA and NLPCA is caused with high probability (more than 30%) from statistical fluctuations. In the ICA case, there is a significant probability ($\sim 15\%$) that the improvement with respect to the baseline system is simply due to statistical fluctuations.
To evaluate the recognition rate as a function of the number of retained Principal Components (Independent Components), the best system (over the validation set) for each value of P (I) was retrained

(over the union of training and validation set) and tested over the test set. The results are shown in the left plot of figure 3. The error rate is reduced by 30.3%, by 21.0% and by 15.8% using KLE, NLPCA and ICA respectively. The right plot of figure 3 shows the performance as a function of the word length for the best systems using raw data, KLE, NLPCA and ICA. The four systems have a similar performance when the length of the word is high (more than 7-8 letters), but for the short words (that are more difficult to recognize) the transform based systems work significantly better.

5.3 Experiments On The Bern Database

The same procedure followed for the Cambridge database was used for the Bern data. In this case, S ranges between 12 and 16 while G goes from 10 to 15.
The best baseline system (the lexicon size is 100) has $S = 14$ and $G = 12$. Its recognition rate over the test set is **77.8**%.
KLE, NLPCA and ICA were applied using the same method described in the previous subsection with P and I ranging from 11 to 16. The system selected through validation when applying KLE has $S = 15$, $G = 12$ and $P = 16$ (the best performing system with $P = 14$ was not selected because it had a lower recognition rate over the validation set). Its performance (over the test set) is **78.8**%. The best NLPCA based system has $S = 15$, $G = 12$ and $P = 14$. The recognition rate is **81.4**%. The best system using ICA has $S = 14$, $G = 11$, $I = 16$ and a recognition rate over the test set of **78.6**%.
The probability of the improvement obtained with NLPCA being a statistical fluctuation is less than 1%, it is then possible to say that the baseline system recognition rate is definitely increased. In the case of the other transforms, the probability of the improvement being due to statistical fluctuations is around 15%.
The performance of the best system for each P (I) was measured over the test set (as for the Cambridge database) and the results are shown in figure 4. The NLPCA performs in general better than KLE and ICA. The latter obtains a slight improvement of the baseline system and appears to be not so effective over this database.
The right plot of figure 4 shows the performance of the system as a function of the word length. As for the Cambridge database, the transform based systems are shown to work significantly better over the short words (less than 4 letters).

6 Conclusions

This work addressed the problem of giving the feature vectors a form more suitable for being modeled with HMMs. This is done by applying three transforms (KLE, NLPCA and ICA) that were compared using both multiple and single writer data.
The transforms are beneficial under several aspects. They can allow one to compress the vectors resulting in data easier to model and HMMs with less parameters. Moreover, when using KLE and ICA, the data is decorrelated and is then more suitable for modeling with Gaussians holding diagonal covariance matrices (in this case, the performance of the system can be improved even without compression).
Two data sets were used, the first is writer dependent, while the second is writer independent. The error rate was reduced by 30.3% over the single writer data and by 16.2% over the multiple writer data.
No transform appears to be systematically better than the others. On the other hand, ICA was shown to determine the lowest improvement of the baseline system over both databases. KLE and NLPCA are shown to perform close to each other for single writer data, while on multiple writer data, the NLPCA appears to be definitely superior. The presence of many writers determines probably a more complex distribution of the data. The NLPCA can then take more advantage from its better ability (due to nonlinearity) in capturing the structure underlying the data distribution.
For both databases, the baseline system showed the lowest performance. This happens because the

feature vectors extracted from the handwritten samples are both correlated and embedded in a space of dimension d higher than their ID. Any system using feature vectors with the same problems can take advantage from the application of the transforms used in this work or of other techniques able to compress and decorrelate the data.

References

[1] T. Steinherz, E. Rivlin, N. Intrator, Off-line cursive script word recognition - a survey, International Journal of Document Analysis and Recognition 2 (2) (1999) 1–33.

[2] R. Plamondon, S. Srihari, On line and off-line handwriting recognition: A comprehensive survey, IEEE Transactions on Pattern Analysis and Machine Intelligence .

[3] A. Vinciarelli, A survey on off-line cursive word recognition, Pattern Recognition 35 (7) (2002) 1433–1446.

[4] M. Chen, A. Kundu, J. Zhou, Off-line handwritten word recognition using a Hidden Markov Model type stochastic network, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (5) (1994) 481–496.

[5] H. Bunke, M. Roth, E. Schukat-Talamazzini, Off-line cursive handwriting recognition using Hidden Markov Models, Pattern Recognition 28 (9) (1995) 1399–1413.

[6] M. Mohamed, P. Gader, Handwritten word recognition using segmentation-free Hidden Markov Modeling and segmentation-based Dynamic Programming techniques, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (5) (1996) 548–554.

[7] A. Kundu, Y. He, M. Che, Alternatives to variable duration hmm in handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (11) (1998) 1275–1280.

[8] S. Knerr, E. Augustin, O. Baret, D. Price, Hidden Markov Model based word recognition and its application to legal amount reading on french checks, Computer Vision and Image Understanding 70 (3) (1998) 404–419.

[9] A. W. Senior, A. J. Robinson, An off-line cursive handwriting recognition system, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3) (1998) 309–321.

[10] A. El-Yacoubi, M. Gilloux, R. Sabourin, C. Suen, An HMM-based approach for off-line unconstrained handwritten word modeling and recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (8) (1999) 752–760.

[11] L. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, in: A. Waibel, L. Kai-Fu (Eds.), Readings in Speech Recognition, Morgan Kaufmann, Palo Alto, CA, 1989, pp. 267–296.

[12] K. Fukunaga, Statistical Pattern Recognition, Academic Press, 1990.

[13] M. Kirby, Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns, John Wiley and Sons, New York, NY, 2001.

[14] C. Bishop, Neural Networks for Pattern Recognition, Cambridge University Press, Cambridge, UK, 1995.

[15] A. Jain, P. Duin, J. Mao, Statistical pattern recognition: a review, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1) (2000) 4–37.

[16] U.-V. Marti, H. Bunke, Towards general cursive script recognition, in: Proceedings of 6^{th} International Workshop on Frontiers in Handwriting Recognition, Seoul, 1998, pp. 379–388.

[17] G. Kaufmann, H. Bunke, Automated reading of cheque amounts, Pattern Analysis and Applications 3 (2000) 132–141.

[18] B. Ripley, Statistical aspects of neural networks, in: J. Bornndorff-Nielsen, J. Jensen, W. Kendal (Eds.), Networks on Chaos: Statistical and Probabilistic Aspects, Chapman and Hall, 1993.

[19] J. Mao, J. A.K., Artificial neural networks for feature extraction and multivariate data projection, IEEE Transactions on Neural Networks 6 (2) (1995) 296–317.

[20] B. Lerner, H. Guterman, M. Aladjem, I. Dinstein, A comparative study of neural network based feature extraction paradigms, Pattern Recognition Letters 20 (1) (1999) 1999.

[21] Y. Le Cun, Y. Bengio, Pattern recognition and neural networks, in: M. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.

[22] P. Baldi, K. Hornik, Learning in linear neural networks: a survey, IEEE Transactions on Neural Networks 6 (4) (1995) 837–858.

[23] H. Bourlard, Y. Kamp, Auto-association by multilayer perceptrons and singular value decomposition, Biological Cybernetics 59 (1988) 291–294.

[24] D. Foteringhame, R. Baddeley, Nonlinear principal components analysis of neuronal spike train data, Biological Cybernetics 77 (4) (1997) 282–288.

[25] E. Malthouse, Limitations of nonlinear pca as performed with generic neural networks, IEEE Transactions on Neural Networks 9 (1) (1998) 165–173.

[26] P. Comon, Independent Component Analysis, a new concept?, Signal Processing 36 (3) (1994) 287–314.

[27] A. Hyvärinen, J. Karhunen, E. Oja, Independent Component Analysis, John Wiley and Sons, New York, NY, 2001.

[28] A. Hyvärinen, Survey on Independent Component Analysis, Neural Computing Surveys 2 (1999) 94–128.

[29] A. Hyvärinen, Fast and robust fixed-point algorithms for Independent Component Analysis, IEEE Transactions on Neural Networks 10 (3) (1999) 626–634.

[30] A. Hyvärinen, E. Oja, Independent Component Analysis: A tutorial, Neural Networks 13 (4-5) (2000) 411–430.

[31] A. Hyvärinen, A fast fixed-point algorithm for Independent Component Analysis, Neural Computation 9 (7) (1997) 1483–1492.

[32] A. Vinciarelli, J. Lüttin, Off-line cursive script recognition based on continuous density HMM, in: Proceedings of the 7^{th} International Workshop on Frontiers in Handwriting Recognition, World Publishing, 2000, pp. 493–498.

[33] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and Cybernetics 9 (1) (1979) 62–66.

[34] A. Vinciarelli, J. Lüttin, A new normalization technique for cursive handwritten words, Pattern Recognition Letters 22 (9) (2001) 1043–1050.

[35] F. Jelinek, Statistical methods for speech recognition, MIT Press, 1997.

[36] A. Viterbi, Error bounds for convolutionalcodes and an asymptotically optimal decoding algorithm, IEEE Transactions on Information Theory 13 (1967) 260–269.

[37] U.-V. Marti, H. Bunke, A full english sentence database for off-line handwriting recognition, in: Proceedings of 5^{th} International Conference on Document Analysis and Recognition, Vol. 1, Bangalore, 1999, pp. 705–708.

[38] M. Stone, Cross-validatory choice and assessment of statistical prediction, Journal of the Royal Statistical Society 36 (1) (1974) 111–147.

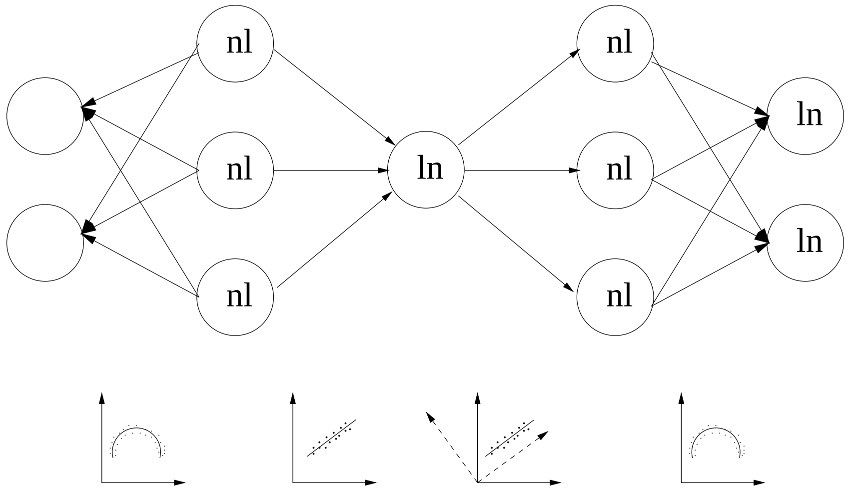

Figure 1: Five layer network performing Nonlinear Principal Component Analysis. The neurons labeled with *ln* have linear activation functions, while those labeled with *nl* have nonlinear activation functions. The lower part of the figure shows the effect on the data of the different layers. At the input the data is distributed with some nonlinear structure. After the first nonlinear layer, the data is mapped into a space where the nonlinear structure becomes linear. The central layer performs a linear PCA (represented by the dashed reference frame). The remaining part of the network maps the data back into the original space.

data	S	G	D	Acc.(%)
raw	11	12	16	92.4
KLE	9	13	16	**94.7**
NLPCA	9	12	14	94.0
ICA	11	14	14	93.6

Table 1: Performance over the test set for Cambridge database. For each system selected through validation, the number of states S per letter model, the number of Gaussians G per state and the dimension D of the feature vector are reported together with the performance over the test set.

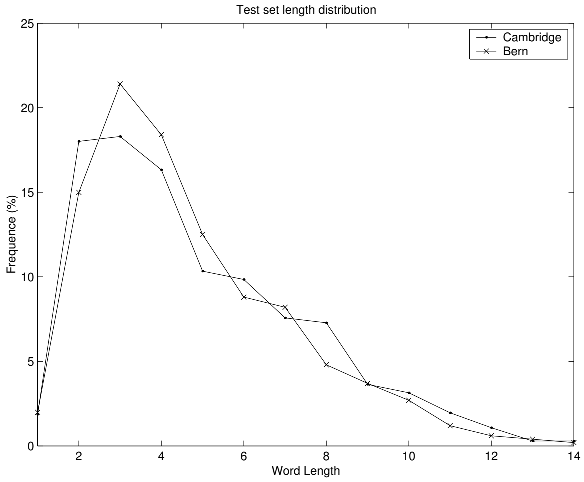

Figure 2: Word length distribution of the Bern and Cambridge Database.

data	S	G	D	Acc.(%)
raw	14	12	16	77.8
KLE	15	12	16	78.8
NLPCA	15	12	14	**81.4**
ICA	14	11	16	78.6

Table 2: Performance over the test set for Bern database. For each system selected through validation, the number of states S per letter model, the number of Gaussians G per state and the dimension D of the feature vector are reported together with the performance over the test set.

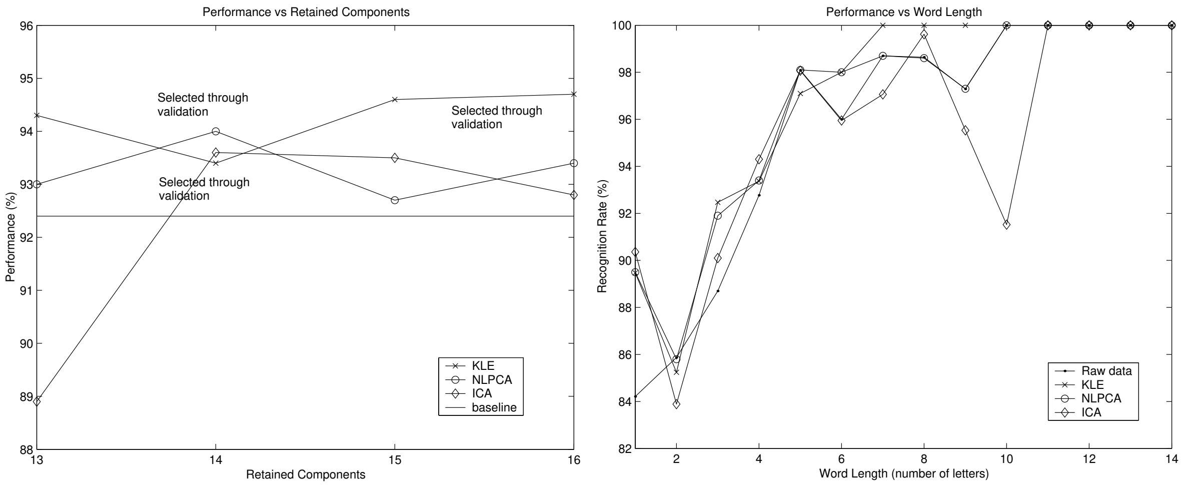

Figure 3: Performance over the Cambridge database. The left plot shows the performance of the system over the test set as a function of the number of Principal (Independent) Components retained. The systems selected through validation are highlighted. The right plot shows the performance (over the test set) of the systems selected through validation as a function of the word length.

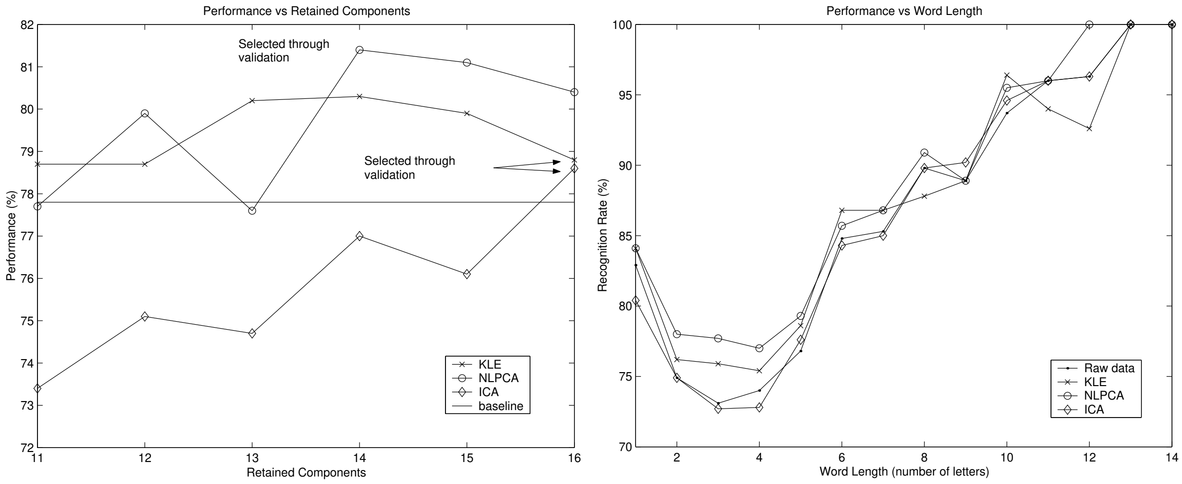

Figure 4: Performance over the Bern database. The left plot shows the performance of the system over the test set as a function of the number of Principal (Independent) Components retained. The systems selected through validation are highlighted. The right plot shows the performance (over the test set) of the systems selected through validation as a function of the word length.