IDIAP RESEARCH REPORT

IDIAP

Martigny - Valais - Suisse

Cursive Character Recognition by Learning Vector Quantization

Francesco Camastra [a] [b]
Alessandro Vinciarelli [c]

IDIAP–RR 00-47

December 2000

PUBLISHED IN
Pattern Recognition Letters, Vol.22, No.6-7, pp.625-629, May 2001.

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] Elsag spa, Via Puccini 2 - 16154 Genova (Italy), e-mail: francesco.camastra@elsag.it
[b] University of Genoa, Computer Science Department, Via Dodecaneso 35 - 16146 Genova (Italy), e-mail: francesco.camastra@disi.unige.it
[c] IDIAP - Institut Dalle Molle d'Intelligence Artificielle Perceptive, Rue du Simplon 4, CP592 - 1920 Martigny, Switzerland, e-mail: Alessandro.Vinciarelli@idiap.ch

IDIAP Research Report 00-47

Cursive Character Recognition by Learning Vector Quantization

Francesco Camastra Alessandro Vinciarelli

December 2000

published in

Pattern Recognition Letters, Vol.22, No.6-7, pp.625-629, May 2001.

Abstract. This paper presents a cursive character recognizer embedded in an off-line cursive script recognition system. The recognizer is composed of two modules: the first one is a feature extractor, the second one an LVQ. The selected feature set was compared to Zernike polynomials using the same classifier. Experiments are reported on a database of about 49000 isolated characters.

1 Introduction

Off-line Cursive Script Recognition (CSR) has several industrial applications such as the reading of postal addresses and the automatic processing of forms, checks and faxes. Among other CSR approaches (Senior and Robinson, 1998)(Kim and Govindaraju, 1997) one attempts to segment words into letters (Steinherz et al., 1999). Since such a task is difficult and error prone, words are usually oversegmented, i.e. the fragments isolated by the segmentation (*primitives*) are expected to be not only characters, but also parts of characters. In order to obtain a letter, it is often necessary to aggregate several primitives. The best complete bipartite match between blocks of primitives and word letters is usually found by applying Dynamic Programming techniques (Bellman and Dreyfus, 1962). Given a word, the matching with the handwritten data is obtained by averaging over the costs due to classifying each aggregation of primitives as one of its letters. The word with the best match, i.e. the lowest cost, is selected as the interpretation of the handwritten sample.
A crucial role is then played by the *cursive character recognizer*. This takes as input a pattern composed of one or more aggregated primitives and gives as output a cost for classifying it as any possible letter.
The *Learning Vector Quantizer* (*LVQ*) was selected as neural classifier because, being a vector quantizer, it yields for each pattern the cost [1] of assigning a pattern to a given letter class.
This paper is organized as follows: in Section 2 the method for extracting features for character representation is presented; a review of LVQ is provided in Section 3; in Section 4 reports some experimental results; in Section 5 some conclusions are drawn.

2 Feature Extraction

Most character recognizers do not work on the raw image, but on a suitable compact representation of the image by means of a vector of features. Since cursive characters present high variability in shapes, a feature extractor should have negligible sensitivity to local shifts and distortions. Therefore feature extractors that perform local averaging are more appropriate than others that yield an exact reconstruction of the pattern (e.g. Zernike polynomials, moments). The feature extraction process used in our recognizer is similar to the one presented in (Pedrazzi and Colla, 1995) and applied to digit and handprinted character recognition. The feature extractor, fed with the binary image of an isolated cursive character, generates local and global features. The local features are extracted from

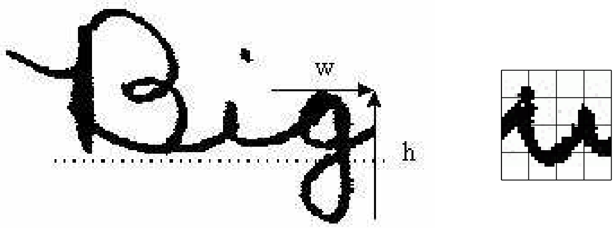

Figure 1: Global features. The dashed line is the *baseline*, the fraction of h below is used as first global feature. The second global feature is the ratio w/h.

subimages (*cells*) arranged in a regular grid [2] covering the whole image. A fixed set of operators is applied to each cell. The first operator is a counter that computes the percentage of foreground pixels in the cell (*gray feature*) with respect to the total number of foreground pixels in the character image. If n_i is the number of foreground pixels in cell i and M is the total number of foreground pixels in

[1] i.e. the distance from the closest prototype of the class.
[2] Small translations of the input patterns can significantly change the distribution of the pixels across the cells. In order to smooth this effect, the cells are partially overlapped.

the pattern, then the gray feature related to cell i is $\frac{n_i}{M}$.
The other operators try to estimate to which extent the black pixels in the cell are aligned along some directions. For each direction of interest, a set of N, equally spaced, straight lines are defined, that span the whole cell and that are parallel to the chosen direction. Along each line $j \in [1, N]$ the number n_j of black pixels is computed and the sum $\sum_i^N n_j^2$ is then obtained for each direction. The difference between the sums related to orthogonal directions is used as feature. In our case the directions of interest were 0^o and 90^o.
We enriched the local feature set with two global features giving information about the overall shape of the cursive character and about its position with respect to the *baseline* of the cursive word. As shown in figure 1, the baseline is the line on which a writer implicitly aligns the word in the absence of rulers. The first global feature measures the fraction of the character below the baseline and detects eventual descenders. The second feature is the $width/height$ ratio.
The number of local features can be arbitrarily determined by changing the number of cells or directions examined in each cell. Since classifier reliability can be hard when the number of features is high (*curse of dimensionality*, (Bellman, 1961)), we use simple techniques for feature selection in order to keep the feature number as low as possible. Directional features corresponding to different directions were applied and the one having the maximal variance was retained. Therefore the feature set was tested changing the number of cells and the grid giving the best results (4×4) was selected.
In the reported experiments we used a feature vector of 34 elements. Two features are global (*baseline* and *width/height ratio*) while the remaining 32 are generated from 16 cells, placed on a regular 4×4 grid; from each cell, the gray feature and one directional feature are extracted.

3 Learning Vector Quantization

Learning Vector Quantization (*LVQ*) is a supervised version of vector quantization and generates codevectors to produce *"near-optimal decision boundaries"* (Kohonen, 1997).
LVQ consists of the application of three consecutive different learning techniques, i.e. LVQ1, LVQ2, LVQ3 [3] . LVQ1 uses for classification the nearest-neighbour decision rule; it chooses the class of the nearest codebook vector.
LVQ1 learning is performed in the following way: if $\bar{m}_t^c$ [4] is the nearest codevector to the input vector $\bar{x}$, then

$$\begin{array}{ll} \bar{m}_{t+1}^c = \bar{m}_t^c + \alpha_t[\bar{x} - \bar{m}_t^c] & \text{if } \bar{x} \text{ is classified correctly} \\ \bar{m}_{t+1}^c = \bar{m}_t^c - \alpha_t[\bar{x} - \bar{m}_t^c] & \text{if } \bar{x} \text{ is classified incorrectly} \\ \bar{m}_{t+1}^i = \bar{m}_t^i & i \neq c \end{array} \tag{1}$$

where α_t is the learning rate at time t.
In our experiments, we used a particular version of LVQ1, that is *Optimized Learning Vector Quantization (OLVQ1)* ((Kohonen, 1997)), a version of the model that provides a different learning rate for each codebook vector. Since LVQ1 tends to push codevectors away from the decision surfaces of the Bayes rule, it is necessary to apply to the codebook generated a successive learning technique called LVQ2.
LVQ2 tries harder to approximate the Bayes rule by pairwise adjustments of codevectors belonging to adjacent classes. If $\bar{m}^s$ and $\bar{m}^p$ are nearest neighbours of different classes and the input vector $\bar{x}$, belonging to the $\bar{m}^s$ class, is closer to $\bar{m}^p$ and falls into a zone of values called *window* [5] , the following rule is applied:

$$\begin{array}{l} \bar{m}_{t+1}^s = \bar{m}_t^s + \alpha_t[\bar{x} - \bar{m}_t^s] \\ \bar{m}_{t+1}^p = \bar{m}_t^p - \alpha_t[\bar{x} - \bar{m}_t^p] \end{array} \tag{2}$$

Since the application of LVQ2 tends to overcorrect the class boundaries, it is necessary to include additional corrections that ensure that the codebook continues approximating the class distributions.

[3] LVQ2 and LVQ3 were proposed, on empirical basis, in order to improve LVQ1 algorithm.
[4] $\bar{m}_t^c$ stands for the value of $\bar{m}^c$ at time t.
[5] The window is defined around the midplane of $\bar{m}^s$ and $\bar{m}^p$.

In order to assure that, it is necessary to apply a further algorithm (LVQ3).
If $\bar{m}^i$ and $\bar{m}^j$ are the two closest codevectors to input $\bar{x}$ and $\bar{x}$ falls in the window, the following rule is applied[6]:

$$\begin{array}{ll}
\bar{m}^i_{t+1} = \bar{m}^i_t & \text{if } C(\bar{m}^i) \neq C(\bar{x}) \wedge C(\bar{m}^j) \neq C(\bar{x}) \\
\bar{m}^j_{t+1} = \bar{m}^j_t & \text{if } C(\bar{m}^i) \neq C(\bar{x}) \wedge C(\bar{m}^j) \neq C(\bar{x}) \\
\bar{m}^i_{t+1} = \bar{m}^i_t - \alpha_t[\bar{x}_t - \bar{m}^i_t] & \text{if } C(\bar{m}^i) \neq C(\bar{x}) \wedge C(\bar{m}^j) = C(\bar{x}) \\
\bar{m}^j_{t+1} = \bar{m}^j_t + \alpha_t[\bar{x}_t - \bar{m}^j_t] & \text{if } C(\bar{m}^i) \neq C(\bar{x}) \wedge C(\bar{m}^j) = C(\bar{x}) \\
\bar{m}^i_{t+1} = \bar{m}^i_t + \alpha_t[\bar{x}_t - \bar{m}^i_t] & \text{if } C(\bar{m}^i) = C(\bar{x}) \wedge C(\bar{m}^j) \neq C(\bar{x}) \\
\bar{m}^j_{t+1} = \bar{m}^j_t - \alpha_t[\bar{x}_t - \bar{m}^j_t] & \text{if } C(\bar{m}^i) = C(\bar{x}) \wedge C(\bar{m}^j) \neq C(\bar{x}) \\
\bar{m}^i_{t+1} = \bar{m}^i_t + \epsilon\alpha_t[\bar{x}_t - \bar{m}^i_t] & \text{if } C(\bar{m}^i) = C(\bar{m}^j) = C(\bar{x}) \\
\bar{m}^j_{t+1} = \bar{m}^j_t + \epsilon\alpha_t[\bar{x}_t - \bar{m}^j_t] & \text{if } C(\bar{m}^i) = C(\bar{m}^j) = C(\bar{x})
\end{array} \tag{3}$$

where $\epsilon \in [0, 1]$ is a fixed parameter.
LVQ3 is self-stabilizing, i.e. the optimal placement of the codebook does not change while continuing learning.

4 Experimental Results

The feature set was tested on a database of cursive characters in conjunction with the LVQ classifier. The database contains isolated characters extracted from handwritten words coming from two sources: the first is the database of handwritten words produced by the Center of Excellence for Document Analysis and Recognition (CEDAR) [7] at the State University of New York (SUNY) at Buffalo ((Hull, 1994)).The second is a database of handwritten postal addresses digitalized by the USPS (United States Postal Service). Word images from which the characters were extracted were all preprocessed according to a scheme that includes morphologic filtering, deslanting and deskewing ((Nicchiotti and Scagliola, 1999)). The character database contains both uppercase and lowercase letters. Uppercase and lowercase versions of the same letter were joined in a single class.
The number of LVQ codevectors, assigned to each class, was proportional to the a-priori class probability. In our experiments all three learning techniques (i.e. LVQ1, LVQ2 and LVQ3) were applied.

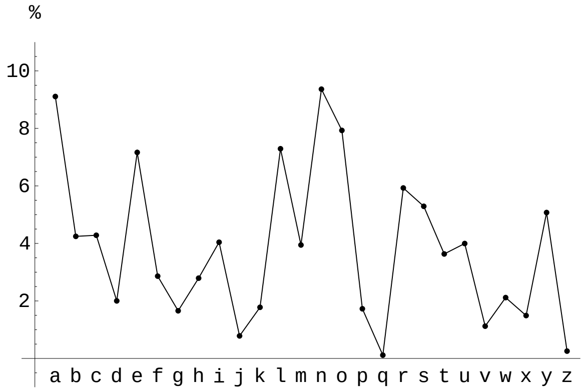

Figure 2: Letter distribution in the test set.

We trained several LVQ nets by specifying different combinations of learning parameters (different learning rates for LVQ1, LVQ2 and LVQ3, and various total number of codevectors). The best LVQ net was selected by means of cross-validation [8] ((Stone, 1974)).

[6] $C(\bar{q})$ stands for the class of $\bar{q}$.
[7] All images belonging to directories train/cities and train/states were used.
[8] This technique consists of dividing the training set into two disjoint sets. On the first set are trained different nets. The second set (*validation set*) is used to select the net with the best performance. Since this procedure can lead some

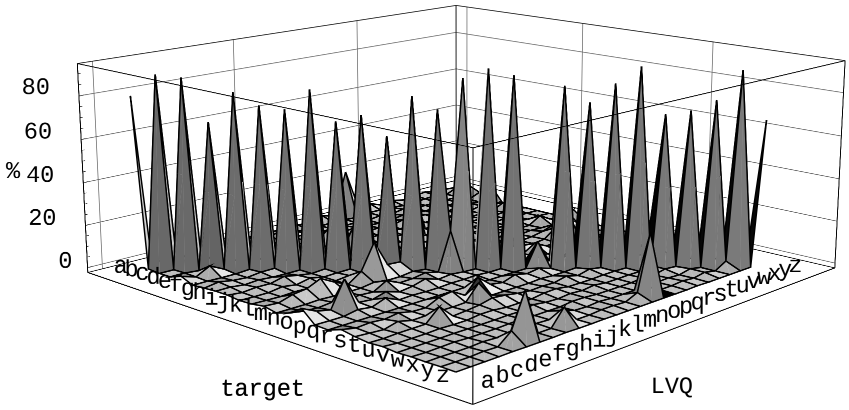

Figure 3: Confusion Matrix for LVQ1+LVQ2+LVQ3 on the test set.

As a comparison Zernike Polynomials ((Khotanzad and Hong, 1990)), in conjunction with LVQ classifier were tried in order to verify the hypothesis that our 34-feature set is more appropriate for cursive character recognition than others that yield a reconstruction of the pattern. Finally experiments were performed using a feature set formed by Zernike Polynomials and 34 features. The experiments were carried out on a training and a test set of 32426 and 16213 characters respectively. Figure 2 shows the percentage distribution of letters in the test set. Since the data is extracted from a database collected by USPS in a real postal plant (Hull, 1994), our database distribution reflects the prior distribution of that site. For this reason some letters are less represented than others or almost absent. In table 1, for different feature sets, the performances on the test set, measured in terms of recognition rate in absence of rejection, are reported. Our best result in terms of recognition rate is **81.72%**. The only

model	34 features	zernike	zernike + 34 features
kNN	62.14%	22.63%	55.65%
LVQ1	77.10%	24.18%	74.67%
LVQ1+LVQ2	81.09%	31.15%	80.25%
LVQ1+LVQ3	81.35%	29.68%	79.55%
LVQ1+LVQ2+LVQ3	81.72%	31.38%	80.64%

Table 1: Recognition rates on the Test Set, in absence of rejection, for several feature sets.

other result we know ((Yamada and Nakano, 1996)) is (approximately **75%**) .
In figure 3, the confusion matrix of 34-feature set with the LVQ1+LVQ2+LVQ3 classifier, on the test set, is shown. Figure 4 shows the probability distribution of correct classification for LVQ1+LVQ2+LVQ3 classifier. The probabilities of classification of a character correctly in top, top three, top twelve are respectively 81.72%, 94.10% and 98.99%.
In our opinion, there are two fundamental sources of misclassification for our classifier are two. The first one (for the most rare letters) is the low number of available samples. The second is the intrinsic ambiguity in cursive characters. In fact, some couples of letters (e.g. e/l or a/o) are difficult to be distinguished. This is confirmed by the confusion matrix and by the high recogniton rate in the top three positions.

overfitting on the validation set, the performance of the selected net has to be confirmed on the test set.

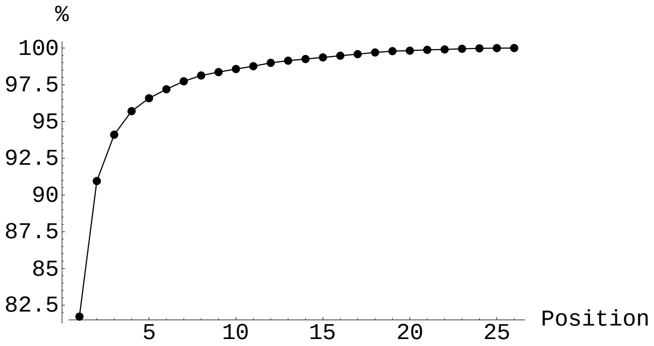

Figure 4: Cumulative probability function of the correct classification of LVQ1+LVQ2+LVQ3 classifier.

5 Conclusion

We have presented a cursive character recognizer embedded in a offline CSR system. The recognizer is formed by a feature extractor, that performs local averaging, and a LVQ classifier. The feature extractor performed better than Zernike polynomials when tested using the same classifier. Besides, the cursive character recognizer yielded better recognition results than others previously reported in literature, ever though they were obtained on a smaller test set.

Acknowledgement The authors wish to thank S.Bruzzo (Elsag Spa) and D.Ugolini (Polo Nazionale Bioelettronica) for commenting on the draft. M.Spinetti (Polo Nazionale Bioelettronica) is gratefully acknowledged for technical support.

References

Bellman, R., 1961. Adaptive Control Processes: A Guided Tour. Princeton University Press.

Bellman, R., Dreyfus, S., 1962. Applied Dynamic Programming. Princeton University Press.

Hull, J. J., May 1994. A database for handwritten text recognition research. IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (5), 550–554.

Khotanzad, A., Hong, Y., 1990. Invariant image recognition by zernike moments. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (5), 489–497.

Kim, G., Govindaraju, V., April 1997. A lexicon driven approach to handwritten word recognition for real time applications. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (4), 366–379.

Kohonen, T., 1997. Self-Organizing Maps. Springer Verlag, Berlin.

Nicchiotti, G., Scagliola, C., 1999. Generalised projections: a tool for cursive character normalization. In: Proceedings of 5^{th} International Conference on Document Analysis and Recognition. IEEE Press.

Pedrazzi, P., Colla, A., 1995. Simple feature extraction for handwritten character recognition. In: Proceedings of IEEE Conference on Image Processing. Vol. 3. IEEE Press.

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

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

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

Yamada, H., Nakano, Y., 1996. Cursive handwritten word recognition using multiple segmentation determined by contour analysis. IEICE Transactions on Informations and Systems 79 (5), 464–470.