Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 3721–3727
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

A SVM-based cursive character recognizer

Francesco Camastra*

Department of Applied Science, University of Naples Parthenope, Via A. De Gasperi 5, 80133 Napoli, Italy

Received 1 September 2005; received in revised form 20 February 2007; accepted 21 March 2007

Abstract

This paper presents a cursive character recognizer, a crucial module in any cursive word recognition system based on a segmentation and recognition approach. The character classification is achieved by using *support vector machines* (SVMs) and a *neural gas*. The neural gas is used to verify whether lower and upper case version of a certain letter can be joined in a single class or not. Once this is done for every letter, the character recognition is performed by SVMs. A database of 57 293 characters was used to train and test the cursive character recognizer. SVMs compare notably better, in terms of recognition rates, with popular neural classifiers, such as learning vector quantization and multi-layer-perceptron. SVM recognition rate is among the highest presented in the literature for cursive character recognition.

Keywords: Support vector machines; Neural gas; Learning vector quantization; Multi-layer-perceptron; Crossvalidation; Cursive character recognition

1. Introduction

Off-line cursive word recognition has many applications such as the reading of postal addresses and the automatic processing of forms, checks and faxes [1,2]. The main approaches [3,4] for off-line cursive word recognition can be divided into segmentation-based and holistic one. The former is based on the word segmentation into letters [5,6] and the recognition of individual letters; the latter tries to recognize the word image as a whole [2].

In the segmentation-based strategy for cursive word recognition, no method is available to achieve a perfect segmentation. Hence the word is first oversegmented, i.e. fragmented into primitives that are characters or parts of them, to ensure that all appropriate letter boundaries have been dissected. To find the optimal segmentation, a set of segmentation hypotheses is tested by merging neighboring primitives and invoking a classifier to score the combination. Finally, the word with the optimal score is generally found by applying dynamic programming techniques [7].

A crucial module in the segmentation-based approach is a cursive character recognizer for scoring individual characters. It has to cope with the high variability of the cursive letters and their intrinsic ambiguity (letters like *e* and *l* or *u* and *n* can have the same shape).

In this paper, we present a cursive character recognizer where the character classification is achieved by using *support vector machines* (SVMs) and a *neural gas* (NG). The NG is used to verify when the upper and lower case versions of a letter can form a common class. This happens when the two characters (e.g. *o* and *O*) are similar in shape and their vectors in the feature space occupy neighboring or even overlapping regions. By grouping the characters in this way, the number of classes is reduced and a more suitable representation of the data is obtained. The classifier, based on SVMs, provides for the character the class attribution. To our best knowledge, the use of SVMs in the cursive character recognition represents a novelty.

The paper is organized as follows: in Section 2 the method for extracting features for character representation is presented; a review of SVM and NG is provided in Sections 3 and 4, respectively; in Section 5 reports some experimental results; in Section 6 some conclusions are drawn.

* Tel.: +39 338 4447991.
E-mail address: francesco.camastra@uniparthenope.it.

0031-3203/$30.00
doi:10.1016/j.patcog.2007.03.014

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) as shown in Ref. [8]. The feature extractor, fed with the binary image of an isolated cursive character, generates local and global features. The local features are extracted from subimages (*cells*) arranged in a regular grid covering the whole image, as shown in Fig. 1. 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 the pattern, then the gray feature related to cell i is 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 $\sum_{j=1}^{N} n_j^2$ 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 and the computation of the directional feature becomes easier. If we indicate with h

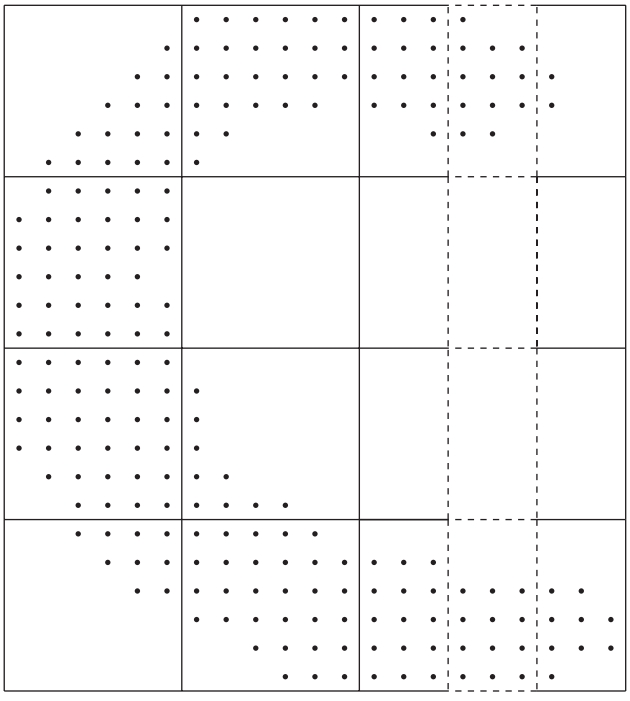

Fig. 1. The image of the character is divided in cells of equal size, arranged in a 4×4 grid. The dashed lines indicate the parts of the cells which are overlapped.

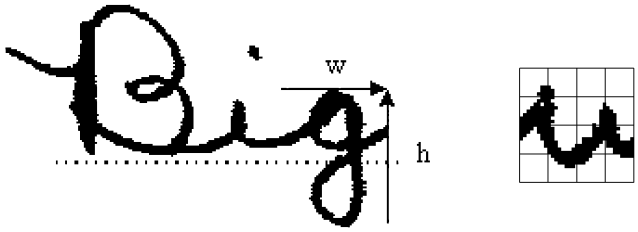

Fig. 2. Global features. The dashed line is the *baseline*, the fraction of h below is used as first global feature. The second global feature is the ratio w/h.

and w, respectively, the height and the width in pixels of the cell, the directional feature d is given by

$$d = \frac{1}{2}\left(1 + \frac{1}{hw^2}\sum_{j=1}^{h} n_j^2 - \frac{1}{h^2 w}\sum_{i=1}^{w} n_i^2\right), \quad (1)$$

where n_j and n_i indicate, respectively, the number of the black pixels along the jth row and the ith column.

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 Fig. 2, 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*, [9]), 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. An implementation, in C language, of the feature extraction process is available on request.

3. SVM for classification

Firstly we recall the definition of *Mercer kernel* [10].

Definition 1. Let X be a nonempty set. A function $G : X \times X \to \mathbb{R}$ is called a *Mercer kernel* (or *positive definite kernel*) if and only if is *symmetric* (i.e. $G(x, y) = G(y, x)\ \forall x, y \in X$) and $\sum_{j=1}^{n}\sum_{k=1}^{n} c_j c_k G(x_j, x_k) \geqslant 0$ for all $n \geqslant 2$, $x_1, \ldots, x_n \subseteq X$ and $c_1, \ldots, c_n \subseteq \mathbb{R}$. Each Mercer kernel $G(\cdot)$ can be represented as: $G(x, y) = \langle \Phi(x), \Phi(y) \rangle$ where $\langle \cdot, \cdot \rangle$ is the inner product and $\Phi : X \to \mathscr{F}$, $\mathscr{F}$ is called *feature space*.