Available online at www.sciencedirect.com
SCIENCE DIRECT
Pattern Recognition 39 (2006) 624–634
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

A prototype classification method and its use in a hybrid solution for multiclass pattern recognition

Chien-Hsing Chou[a], Chin-Chin Lin[b], Ying-Ho Liu[a], Fu Chang[a,*]

[a]*Institute of Information Science, Academia Sinica, 128 Section 2, Academia Road, Nankang, Taipei 115, Taiwan*
[b]*Department of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan*

Received 12 October 2004; received in revised form 28 October 2005; accepted 28 October 2005

Abstract

In this paper, we propose a prototype classification method that employs a learning process to determine both the number and the location of prototypes. This learning process decides whether to stop adding prototypes according to a certain termination condition, and also adjusts the location of prototypes using either the K-means (KM) or the fuzzy c-means (FCM) clustering algorithms. When the prototype classification method is applied, the support vector machine (SVM) method can be used to post-process the top-rank candidates obtained during the prototype learning or matching process. We apply this hybrid solution to handwriting recognition and address the convergence behavior and runtime consumption of the prototype construction process, and discuss how to combine our prototype classifier with SVM classifiers to form an effective hybrid classifier.

Keywords: Fuzzy c-means clustering algorithm; Handwritten character recognition; Hybrid classifier; K-means clustering algorithm; Prototype learning; Support vector machine

1. Introduction

The support vector machine (SVM) classification method [1] represents a major development in pattern recognition research. Two innovations of SVM are responsible for the success of this method, namely, the ability to find a hyperplane that divides samples into two groups with the widest margin between them, and the extension of this concept to a higher-dimensional setting using a kernel function to represent a similarity measure on that setting. Both innovations can be formulated in a quadratic programming framework whose optimal solution is obtained in a computation time of a polynomial order. This makes SVM a practical and effective solution for many pattern recognition problems.

SVM is essentially a method for binary classification, in which each object is classified as one of two classes. When dealing with a multiclass classification, in which each object is classified as one of N classes, where $N > 2$, the problem must be decomposed into binary classification sub-problems and the SVM method must then be applied to these sub-problems. Two useful approaches for decomposing the problem are one-against-one [2] and DAGSVM [3]. In the training phase, both approaches require $N(N-1)/2$ binary classification problems to be solved. In the testing phase, the one-against-one approach conducts $N(N-1)/2$ classifications, while DAGSVM employs a directed acyclic graph with $N(N-1)/2$ nodes and N leaves, which reduces the number of classifications to $N-1$. There is another decomposition approach called one-against-all, or one-against-others. In this approach, there are Nsub-problems, each of which classifies an object as A or not A. Some comparative work [4], and our own experience, show that this approach requires more training time and more support vectors than the other two approaches. In this paper, therefore, we only consider the one-against-one and DAGSVM approaches.

* Corresponding author. Tel.: +886 2 2788 3799X1819; fax: +886 2 2782 4814.
E-mail addresses: ister@iis.sinica.edu.tw (C.-H. Chou), erikson@iis.sinica.edu.tw (C.-C. Lin), daxliu@iis.sinica.edu.tw (Y.-H. Liu), fchang@iis.sinica.edu.tw (F. Chang).

0031-3203/$30.00
doi:10.1016/j.patcog.2005.10.022

When the number of classes N is large, both one-against-one and DAGSVM incur an excessive amount of training time and produce an extremely large set of support vectors. However, these problems can be overcome by reducing the binary classification sub-problems to a much smaller set. This can be done, for example, by way of the *k-nearest neighbor* (k-NN) method [5–7], which matches each object with all possible training samples and considers k-nearest samples, $k \geqslant 1$, in its classification decision. It has been shown that the asymptotic error rate of k-NN is less than twice the Bayes rate [8]. In many applications, k-NN does indeed achieve good accuracy rates, although usually not as good as those achieved by SVM. Because of the complementary properties of k-NN and SVM, we can use a hybrid solution that first employs k-NN to determine the top-rank candidates and then applies SVM to post-process them. By so doing, we reduce the number of binary classification sub-problems, since in most applications not all class types are included in the top-rank candidates.

The k-NN method, however, is too slow for large-scale applications and also requires storage of a large set of training samples in the memory. In this paper, we explore an alternative solution by reducing training samples to a much smaller set of prototypes that can be derived from training samples in a learning process. If the training samples are represented as vectors in *D-dimensional* Euclidean space, prototypes can reside in the same space, although they are not necessarily training samples. We use this prototype classification method, instead of the k-NN method in the hybrid solution.

The merit of the hybrid solution is demonstrated by the following example. In a handwriting application, training an SVM classifier for 3036 character types out of 303,600 training samples takes 32 days on a PC with a Pentium IV 2.4 GHz CPU and 2 GB RAM. In the testing phase, DAGSVM requires 31.78 s to recognize a character and has to store approximately 1.5×10^8 support vectors in the memory. If, however, we use one of the hybrid approaches proposed in this paper, it is only necessary to store 9678 prototypes (3.2% of the number of training samples), and 10,547,211 support vectors (7% of the total number of support vectors) that DAGSVM requires when it alone is used for the classification. The substantial saving in computation time is due to the fact that only 7.2% of class pairs require further SVM post-processing, resulting in 60.9 h of training time and only 0.04 s to recognize a character in the testing phase.

We propose two learning algorithms to determine the number and the location of prototypes. They differ in the method used to adjust prototype locations, and in the stopping criterion for the prototype construction process. The first method employs the K-means (KM) clustering algorithm to determine prototypes by way of samples that take them as their nearest prototypes. In contrast, the second method applies the fuzzy c-means (FCM) clustering algorithm, which determines prototypes as a weighted average of all samples, where the weight contributed by each sample is inversely proportional to its distance from the prototypes. We compare the convergence behavior and runtime performance of these two learning algorithms. An important issue is how to combine the prototype and SVM classifiers into a computationally effective classifier and attain comparable accuracy rates to those achieved by using SVM solely.

The remainder of this paper is organized as follows. Section 2 contains the proposed learning algorithms used to construct prototypes from training samples. In Section 3, we describe the disambiguation process that uses SVM in the training and testing phases. Section 4 details the application of our hybrid solution to handwritten character recognition. Finally, in Section 5, we present our conclusions.

2. Prototype-construction method

We assume that all training samples are represented as vectors in D-dimensional Euclidean space. Prototypes are also vectors in the same space, but they do not have to be samples themselves. The distance between any two vectors $\mathbf{v} = (v_1, v_2, \ldots, v_D)$ and $\mathbf{w} = (w_1, w_2, \ldots, w_D)$ is

$$\|\mathbf{v} - \mathbf{w}\| = \left[\sum_{d=1}^{D} (v_d - w_d)^2\right]^{1/2}. \quad (1)$$

We propose two algorithms to determine the number and the location of prototypes. Although employing a similar learning process, they differ in the way the location of prototypes is computed and in the criterion for stopping the process. We first describe the process that uses KM to adjust the location of prototypes. Since KM is a crisp or non-fuzzy method, it is referred to as a *crisp construction process* (CCP). The steps of the process are as follows:

K1 Initiation: For each class type C, we use the statistical average of all C-samples to initiate a C-prototype.
K2 Absorption: For each sample **s**, find the prototype **p** that is nearest to **s**. If the class type of **s** matches the class type of **p**, declare **s** as *absorbed*; otherwise, **s** is unabsorbed. If all samples are absorbed, we terminate the process; otherwise, we proceed to K3.
K3 Prototype augmentation: For each C, if there exist any unabsorbed C-samples, we select one of them as a new C-prototype; otherwise, no new prototype is added to class type C.
K4 Prototype adjustment: For the C to which a new C-prototype is added in K3, we apply KM to adjust all C-prototypes, using the current C-prototypes as seeds. We then proceed to K2.

In K3, an unabsorbed C-sample is selected as follows. We focus on a set Ψ_C, consisting of unabsorbed C-samples that are *not* themselves C-prototypes, and let each sample in Ψ_C cast a vote to the nearest sample in Ψ_C. We then select the