Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 1498–1509
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Self-generating prototypes for pattern classification

Hatem A. Fayed[a], Sherif R. Hashem[a], Amir F. Atiya[b,*]

[a]*Department of Engineering Mathematics and Physics, Cairo University, Giza, Egypt*
[b]*Department of Computer Engineering, Cairo University, Giza, Egypt*

Received 15 February 2006; received in revised form 15 February 2006; accepted 17 October 2006

Abstract

Prototype classifiers are a type of pattern classifiers, whereby a number of prototypes are designed for each class so as they act as representatives of the patterns of the class. Prototype classifiers are considered among the simplest and best performers in classification problems. However, they need careful positioning of prototypes to capture the distribution of each class region and/or to define the class boundaries. Standard methods, such as learning vector quantization (LVQ), are sensitive to the initial choice of the number and the locations of the prototypes and the learning rate. In this article, a new prototype classification method is proposed, namely self-generating prototypes (SGP). The main advantage of this method is that both the number of prototypes and their locations are learned from the training set without much human intervention. The proposed method is compared with other prototype classifiers such as LVQ, self-generating neural tree (SGNT) and *K*-nearest neighbor (*K*-NN) as well as Gaussian mixture model (GMM) classifiers. In our experiments, SGP achieved the best performance in many measures of performance, such as training speed, and test or classification speed. Concerning number of prototypes, and test classification accuracy, it was considerably better than the other methods, but about equal on average to the GMM classifiers. We also implemented the SGP method on the well-known STATLOG benchmark, and it beat all other 21 methods (prototype methods and non-prototype methods) in classification accuracy.

Keywords: Prototype classifiers; Nearest neighbor; Learning vector quantization; Self-generating neural trees; Gaussian mixture models

1. Introduction

The simplest and most intuitive approach in pattern classification is based on the concept of similarity [1,2]. Patterns that are similar (in some sense) are assigned to the same class. Prototype classifiers are one major group of classifiers that are based on similarity. A number of prototypes are designed so as they act as representatives of the typical patterns of a specific class. When presenting a new pattern, the nearest prototype determines the classification of the pattern. Two extreme ends of the scale for prototype classifiers are the nearest neighbor classifier, where each pattern serves as a prototype, and the minimum distance classifier, where there is only one prototype (the class center or mean) per class. Practically speaking, the most successful prototype classifiers are the ones that have a few prototypes per class, thus economically summarizing all data points into a number of key centers. Learning vector quantization (LVQ) [3] is probably the most well-known prototype classifier. Other methods also include self-generating neural tree (SGNT) [4,5], which is a hierarchical tree structure, where all the training patterns (or specifically the misclassified ones) are repeatedly presented to the tree until the method correctly classifies all the patterns. Some other prototype classifiers have also been developed such as methods that compactly cover each class region by a set of hyperspheres [6,7] or ones that use a set of hyperellipsoids [8] or a set of hyperrectangles [9]. Another prototype classifier is the Gaussian mixture model (GMM), which is based on modeling the class-conditional densities as a Gaussian mixture [1,10]. The well-known EM algorithm is used to design such a classifier. Each

* Corresponding author. Tel.: +20 2 3354773.
E-mail addresses: h_fayed@eng.cu.edu.eg (H.A. Fayed), shashem@ieee.org, shashem@mcit.gov.eg (S.R. Hashem), amiratiya@link.net, amir@alumni.caltech.edu (A.F. Atiya).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.10.018

Gaussian component will serve as a prototype. Many of the prototype classifiers suffer from some drawbacks. For example, in LVQ, the optimal number of prototypes is not known a priori, and it has to be determined by re-running the algorithm several times, each time with a different number of prototypes. Also, the method is sensitive to the initial prototype locations. Consequently, each run with different initial conditions can lead to a different final solution. The SGNT method [4,5] is too sensitive to the order of presentation of the patterns (the patterns presented first are too influential). Moreover, it is also sensitive to the selected value of the distance threshold. To overcome these types of problems, a new method is proposed in this paper. The distinctive feature of the proposed method is that it does not require any predefined parameters (except those often added to any algorithm to avoid over-training in real/noisy data problems), and does not depend on the order of the input patterns. In this method, both the number of prototypes and their locations are learned from the training patterns. We do not need to guess a suitable number of prototypes, or keep rerunning the algorithm many times experimenting with different numbers of prototypes, like in LVQ. The method keeps adding prototypes as needed, until it stops with a suitable number of prototypes. In addition, as the simulations show, the number of the resulting prototypes for the proposed method turned out to be usually less than those for other prototype classification methods achieving the same accuracy. This also indicates a more compact solution and a smaller model complexity for the proposed method.

This paper is organized as follows. The basic idea of the proposed approach is introduced in the following section. Section 3 presents experimental results that examine the effectiveness of the proposed approach and compares it to common prototype classification methods. Section 4 presents a discussion of the experimental results, and finally, in Section 5 conclusions are given.

2. Self-generating prototypes (SGP)

The main idea of this method is to form a number of groups, each of which contains some patterns of the same class, and each group's mean is used as a prototype for the group. Initially, patterns of each class form a group and their mean is computed as the initial group's prototype. We then successively split some groups, shift some patterns from one group to another, and possibly merge some groups as a pruning step. All operations performed are very simple and can be classified according to the four possible situations that might occur. These are described in details below:

- If for all patterns of a group the closest prototype is the group prototype, then no modification is performed.
- If for all patterns of a group the closest prototypes is one of an incorrect class, this often occurs when patterns of the group are clustered into subgroups separated by patterns of other classes, the group is split into two subgroups. This is accomplished by separating the points by a hyperplane which passes through the original group's mean and which is perpendicular to the first principal component of the original group's patterns.
- If for some patterns of a group the closest prototype is a prototype of a different group but of the same class, these patterns are shifted from the original group to the group of that closest prototype.
- If for some patterns of a group the closest prototype is a prototype of a different group and of an incorrect class, these patterns are removed from the original group and form a new group, and its mean is computed as a new prototype.

In each case, each group mean is recomputed at the end to update the locations of the prototypes. The whole process is repeated until no change occurs to the groups. A merging step could be applied to reduce the number of prototypes. Groups A and B are merged if both A and B have the same class and the second closest prototype to the patterns of group A is the prototype (mean) of group B (P_B), and the second closest prototype to patterns of group B is P_A. A pruning step could also be used to remove redundant prototypes (whose removal will not affect the classification). In this step, if the second closest prototypes of all patterns of a certain group have the same class, the group and its prototype are removed. We call the SGP algorithm that has no merging and pruning steps as SGP1 and the one that has the merging and pruning steps as SGP2. Steps of the SGP1 algorithm can be summarized as follows:

Algorithm (SGP1)

Input: N training patterns pairs $\{x_j, C(x_j)\}$, $j = 1, 2, \ldots, N$ where $C(x_j) \in \{1, 2, \ldots, K\}$ is the class label for pattern x_j.

Output: Prototype set $\{P_k\}$, $k = 1, 2, \ldots, M$ and their corresponding class labels.

Method:

1. Set $G_k = \{x_j : C(x_j) = k\}$, $k = 1, 2, \ldots, K$.
2. Compute the initial prototypes as P_k=mean(G_k) and its class label $C(P_k) = k$, $k = 1, 2, \ldots, K$.
3. Set $k = 1$, $M = K$.
4. Compute $d_{js} = \|x_j - P_s\|_2 \; \forall x_j \in G_k, s = 1, 2, \ldots, M$.
5. Determine the index of the closest prototype to each pattern x_j as $i_j^* = \arg\min_s (d_{js})$.
6. If $i_j^* = k$, $\forall x_j \in G_k$ go to step 10.
7. If $C(P_{i_j^*}) \neq C(P_k)$, $\forall x_j \in G_k$, set $M = M + 1$, split group G_k into two subgroups G_k and G_M as described above, and update their means: P_k= mean(G_k), P_M = mean(G_M), $C(P_M) = C(P_k)$, go to step 4.
8. If $C(P_{i_j^*}) = C(P_k)$, $P_{i_j^*} \neq P_k$ for some $x_j \in G_k$, remove these patterns from G_k and include them in group $G_{i_j^*}$. P_k = mean(G_k), $P_{i_j^*}$ = mean($G_{i_j^*}$).