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($G_{i_j^*}$).

Pattern	1	2	3	4	5	6	7	8	9	10	11	12
Class	1	1	1	2	2	2	1	1	2	2	1	1

Fig. 1. Patterns and corresponding classes for example 1.

G_1	$1^{(1)}$	$2^{(1)}$	$3^{(1)}$	$7^{(2)}$	$8^{(2)}$	$11^{(2)}$	$12^{(2)}$	Mean=6.29	Class=1
G_2	4	5	6	9	10	Mean=6.8	Class=2		

Fig. 2. Initial groups and corresponding prototypes for example 1. Superscript numbers indicate the index of the closest prototype to the pattern. The closest prototype to patterns 7, 8, 11, 12 is that of G_2.

G_1	1	2	3	Mean=2	Class=1		
G_2	$4^{(1)}$	$5^{(2)}$	$6^{(2)}$	$9^{(3)}$	$10^{(3)}$	Mean=6.8	Class=2
G_3	7	8	11	12	Mean=9.5	Class=1	

Fig. 3. Groups after manipulation of G_1.

9. If $C(P_{i_j^*}) \neq C(P_k)$, for some $x_j \in G_k$, set $M = M + 1$, remove these patterns from G_k and create a new group G_M containing these patterns. Update the means: $P_k = \text{mean}(G_k)$, $P_M = \text{mean}(G_M)$.
10. If $k = M$ and no change is made to the groups or the prototypes, then STOP.
11. If $k \neq M$, then set $k = k + 1$ and go to step 4.
12. If $k = M$, then set $k = 1$ and go to step 4.

2.1. Generalization capability

The SGP method, as described above obtains zero classification error on the training set. In practice, it is extremely beneficial, from the generalization point of view, to allow for a small training error in return for smaller model complexity [1,2]. We implement this concept here on both SGP methods. Two parameters are invoked namely, $R_{\min}$ and R_{mis}. If the size of a group divided by the size of the largest group is less than a threshold $R_{\min}$, the group is discarded. (So the patterns belonging to that group will not belong to any group.) Also, if the number of misclassified patterns in a group divided by the size of that group is less than a threshold R_{mis}, the group is maintained unchanged. These parameters can be estimated using any model assessment method such as cross validation or bootstrapping. These modifications result in a much smaller number of prototypes, especially in regions of overlap, hence a smaller complexity model and less chance for overfitting.

2.2. Example 1

To illustrate the basic idea of the SGP methods, the SGP1 algorithm (without invoking $R_{\min}$ and R_{mis}) is applied to the following example. Consider the following patterns (see Fig. 1) for a one-dimensional problem, where the white color indicates class 1 and the gray color indicates class 2.

Initial groups G_1 and G_2 and their corresponding means (prototypes) and classes are shown in Fig. 2. In manipulating G_1, patterns 7, 8, 11, 12 are eliminated from group G_1 because the nearest mean to them is that of G_2 which corresponds to the other class. The eliminated patterns form a new group G_3 (Fig. 3). Similarly, in manipulating G_2, patterns 4, 9, 10 form a new group G_4 (Fig. 4). In the same way, in manipulating G_3, patterns 7,8 form a new group G_5 (Fig. 5). In manipulating G_4, the nearest mean to pattern 4 is that of G_2 which corresponds to the same class, hence pattern 4 is shifted to group G_2 (Fig. 6). Fifth group (G_5) manipulation necessitated no change. After the second iteration for all groups, no change is made, thereby we stop and the final prototypes for class 1 are: 2, 7.5, 11.5 and those for class 2 are: 5, 9.5.

The resultant self-generating tree SGNT [4,5] (with repeated training until reaching correct classification) for the above example is shown in Fig. 7. The prototypes for class 1 are: 1.4, 3, 7.43, 11.2 and those for class 2 are: 4, 5, 6, 9, 10. It can be noticed that SGNT failed to infer the best clusters. Moreover, SGNT prototypes are biased to some patterns according to the order of presentation of patterns to

G_1	1	2	3	Mean=2	Class=1	
G_2	5	6	Mean=5.5	Class=2		
G_3	$7^{(4)}$	$8^{(4)}$	11	12	Mean=9.5	Class=1
G_4	4	9	10	Mean=7.67	Class=2	

Fig. 4. Groups after manipulation of G_2.

G1	1	2	3	Mean=2	Class=1
G2	5	6	Mean=5.5	Class=2	
G3	11	12	Mean=11.5	Class=1	
G4	$4^{(2)}$	$9^{(4)}$	$10^{(4)}$	Mean=7.67	Class=2
G5	7	8	Mean=7.5	Class=1	

Fig. 5. Groups after manipulation of G_3.

G_1	1	2	3	Mean=2	Class=1
G_2	5	6	4	Mean=5	Class=2
G_3	11	12	Mean=11.5	Class=1	
G_4	9	10	Mean=9.5	Class=2	
G_5	$7^{(5)}$	$8^{(5)}$	Mean=7.5	Class=1	

Fig. 6. Groups after manipulation of G_4.

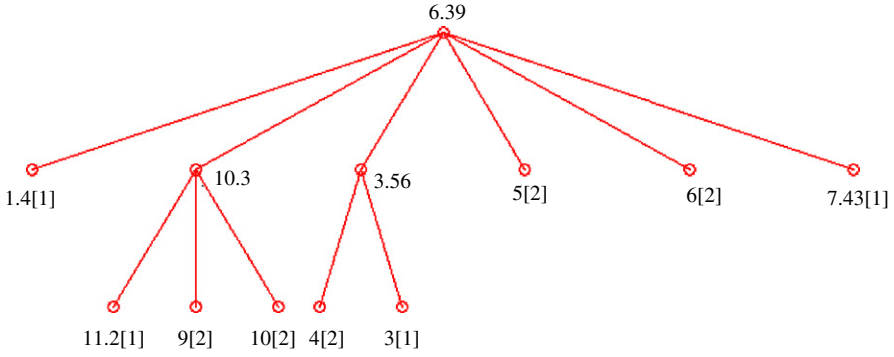

Fig. 7. Resultant SGNT for example 1, first numbers represent the prototypes and the numbers in brackets represent the class label.

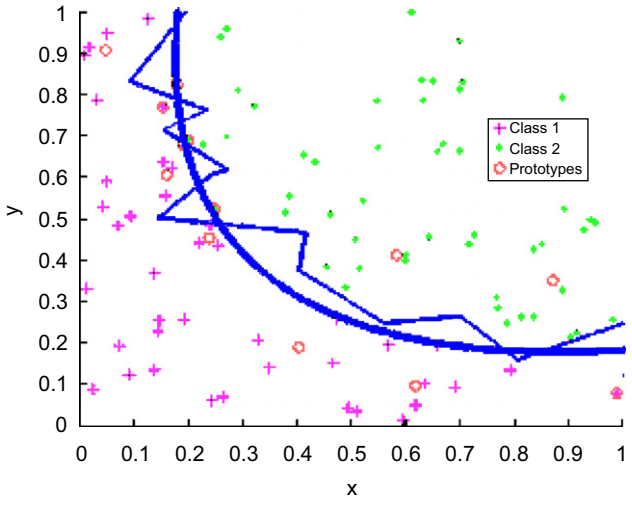

Fig. 8. Resulting piecewise linear boundary of SGP2 to Parabola example.

the tree. On the other hand, SGP prototypes succeeded to infer the best clusters of each class patterns and the final prototypes are the groups' means. It might be expected that SGNT would perform poorly compared to a classification algorithm, as can be seen from the results, since it was originated as a clustering algorithm and then adapted to handle classification problems.

2.3. Example 2

We generated 100 uniformly randomly distributed points in two dimensions and assigned two classes according to the following predetermined parabolic boundary:

$$\begin{aligned} &(x-y)^2 - \sqrt{2}(x+y) + 1 > 0 \quad \text{Class 1,} \\ &\text{otherwise} \qquad\qquad\qquad\qquad\quad \text{Class 2,} \end{aligned} \tag{1}$$

where both x and $y \in [0, 1]$. SGP2 algorithm results in 13 prototypes shown as circles in Fig. 8. An interesting result can be noted that the resultant prototypes are somehow near the boundary thus they approximate the boundary by the equidistance lines between the adjacent prototypes of different classes.

3. Experimental results

To validate the proposed SGP methods, a number of synthetic and real world data sets are tested using nearest neighbor (1-NN) method, K-nearest neighbor (K-NN) method, LVQ, SGNT, SGP1 and SGP2. Two types of GMM are also compared: GMM1 that uses the spherical covariance matrix ($\Sigma = \sigma^2 I$) and GMM2 that uses the full covariance matrix. In our implementation we used MATLAB 6.5 [11] on Windows 2000 operating system running on Intel PC 1.0 GHz 256 MB RAM. In all methods, the optimum values for the parameters are determined using five-fold cross-validation and early stopping is utilized to stop the training

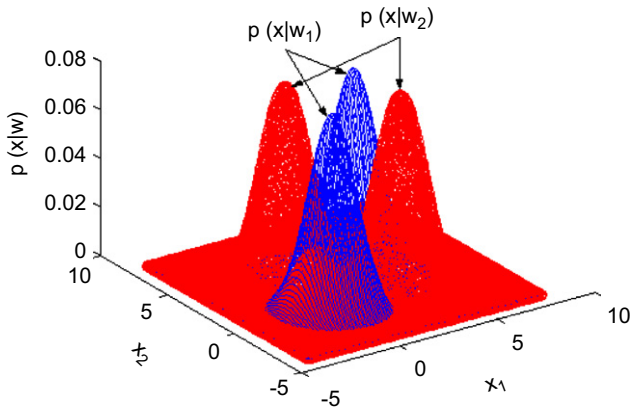

Fig. 9. Distributions of the *DIAGONAL* problem.

when the validation error begins to increase [1,2]. For the K-NN method, suggested values for K are $\{1, 3, 5, \ldots, 19\}$. For the LVQ method, the number of prototypes (P_k) for class k is selected as: $\delta \times N_k$ where $\delta \in \{0.01, 0.1, 0.2\}$, N_k is the number of training patterns for class k and values suggested for the learning rate (α) are: $\{0.01, 0.02, \ldots, 0.1\}$. In the SGNT method, suggested values for the distance threshold are: $\{0.01, 0.02, \ldots, 0.1\}$. In the SGP methods, values suggested for $R_{\min}$ and R_{mis} are $\{0.01, 0.02, \ldots, 0.2\}$. For GMM methods, the number of mixture components examined is 1–50 for large data sets (*Thyroid*, *Letter*) and 1–10 for the others. The number of mixture components is determined using the Bayesian Information Criterion (BIC) (see Ref. [2]). The performance measures tested are the following:

- CPU time elapsed in training. That also includes the cross-validation step to tune the parameters.
- CPU time elapsed in testing. By testing, we mean just implementing the compared methods to obtain a classification.
- Number of obtained prototypes (number of mixture components for GMM methods).
- Classification error for the test set.

We now describe the data sets used in the comparison.

3.1. Synthetic data sets

3.1.1. Diagonal data set (DIAGONAL)

We have here a two-class data set generated using the distributions used in Ref. [12]. Each class consists of two normal distributions as follows:

$$\begin{aligned} p(x|w_1) &= \tfrac{1}{2} N(\mu_{11}, I) + \tfrac{1}{2} N(\mu_{12}, I), \\ p(x|w_2) &= \tfrac{1}{2} N(\mu_{21}, I) + \tfrac{1}{2} N(\mu_{22}, I), \end{aligned} \tag{2}$$

where w_1, w_2 are the two classes and $\mu_{11} = [0\ 0]^{\mathrm{T}}$, $\mu_{12} = [\mu\ \mu]^{\mathrm{T}}$, $\mu_{21} = [0\ \mu]^{\mathrm{T}}$, $\mu_{22} = [\mu\ 0]^{\mathrm{T}}$ and $N(\mu, I)$ is the multivariate normal distribution with mean μ and covariance matrix equals to the identity matrix I. We used $\mu = 3.5$

Table 1
Average CPU training time over all runs with different parameters for *Diagonal* data set

Training set size	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	0.10	0.84	1.57	0.04	0.05	0.07	0.12
200	0.29	0.89	1.94	0.05	0.07	0.10	0.39
300	0.55	1.68	7.39	0.07	0.09	0.11	0.93
400	1.04	2.25	9.76	0.07	0.09	0.12	1.10
500	1.61	1.72	10.69	0.09	0.12	0.14	1.93
600	2.24	1.15	16.15	0.11	0.15	0.17	1.98
700	3.46	1.50	17.83	0.10	0.13	0.15	2.15
800	4.47	1.86	20.54	0.15	0.20	0.20	1.93
900	5.58	1.46	26.08	0.13	0.17	0.21	3.42
1000	6.81	1.92	29.50	0.15	0.16	0.25	3.59

Table 2
CPU test classification time for *Diagonal* data set

Training set size	1-NN	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	0.78	1.85	0.16	22.30	0.04	0.04	0.08	0.07
200	1.61	2.53	0.26	26.70	0.07	0.06	0.08	0.09
300	2.39	3.63	0.53	29.55	0.04	0.04	0.08	0.10
400	3.18	4.17	0.67	31.09	0.04	0.04	0.09	0.09
500	4.03	4.95	0.92	32.70	0.04	0.04	0.10	0.08
600	4.84	5.60	0.36	35.10	0.04	0.04	0.09	0.11
700	5.52	10.59	2.32	34.01	0.04	0.04	0.11	0.09
800	6.17	11.98	0.46	34.96	0.04	0.04	0.10	0.08
900	11.77	14.76	0.96	35.77	0.04	0.04	0.12	0.10
1000	25.69	57.30	0.56	35.77	0.04	0.04	0.10	0.10

Table 3
Number of prototypes for *Diagonal* data set

Training set size	1-NN or *K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	100	20	53	4	4	7	4
200	200	40	106	16	12	8	5
300	300	90	131	4	4	9	6
400	400	120	166	4	4	8	5
500	500	150	295	4	4	8	5
600	600	60	250	4	4	9	6
700	700	210	248	4	4	9	5
800	800	80	427	4	4	8	5
900	900	180	438	4	4	10	6
1000	1000	100	446	4	4	9	6

in our experiments, 10,000 testing examples and a varying number of training examples: 100, 200, 300, ..., 1000. The distribution is shown in Fig. 9. Tables 1 and 2 show the CPU time elapsed in training and testing for the different methods respectively. Tables 3 and 4 show comparison of the number of prototypes and the test classification error respectively. The test classification errors for different methods are shown in Fig. 10.

3.1.2. I–I Data set (I–I)

Here we considered a higher-dimensional data set used in Refs. [12,13]. We generated the data points of the two classes from the n-dimensional normal distributions $N(\mu_i, \Sigma_i)$, $i = 1, 2$. The parameters are: $\mu_1 = [00\ldots0]^T$, $\mu_2 = [\mu0\ldots0]^T$, $\Sigma_1 = \Sigma_2 = I$. The value of μ controls the degree of overlap between the two distributions. We used six-dimensional normal distributions and set $\mu = 2.56$ in the experiments, 10,000 testing examples and a varying number of training examples: 100, 200, 300, ..., 1000. Tables 5 and 6 show CPU time elapsed in training and testing for different methods respectively. Tables 7 and 8 show comparison of the number of prototypes and the test classification error, respectively. The test classification errors for different methods are shown in Fig. 11.

Table 4
Test classification error (%) for *Diagonal* data set

Training set size	1-NN	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	9.8	**8.3**	10.6	11.7	9.4	9.4	8.9	9.2
200	10.4	8.6	11.2	12.3	9.3	9.6	**8.0**	8.4
300	10.5	8.4	10.8	11.1	**8.0**	8.3	8.1	8.1
400	10.8	8.6	10.4	11.4	**8.1**	**8.1**	8.2	8.5
500	11.2	8.4	10.7	11.4	**7.9**	**7.9**	8.1	8.7
600	10.6	8.2	10.9	11.4	**8.1**	**8.1**	8.2	8.3
700	10.4	8.4	10.4	11.1	8.2	8.2	**8.1**	8.3
800	10.7	**8.0**	10.6	11.5	8.5	8.5	8.1	8.1
900	10.7	8.3	10.3	10.4	**8.0**	**8.0**	**8.0**	8.2
1000	10.9	8.2	10.2	10.8	8.1	8.1	**8.0**	8.1

Bold numbers indicate the minimum test classification error while underlined numbers indicate the second minimum.

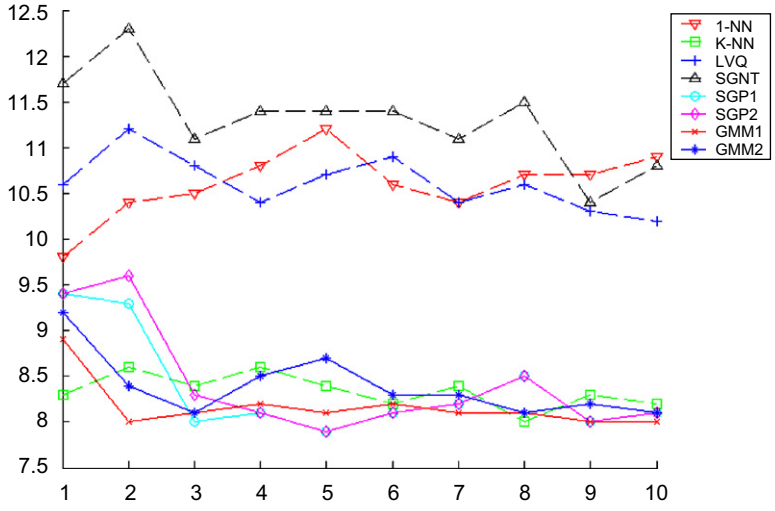

Fig. 10. Test classification error comparison of different methods for *Diagonal* data set.

Table 5
Average CPU training time over all runs with different parameters for *I–I* data set

Training set size	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	0.13	0.27	1.20	0.02	0.03	0.10	0.16
200	0.30	0.55	2.98	0.03	0.04	0.13	0.41
300	0.60	0.38	3.93	0.03	0.04	0.15	0.88
400	1.06	0.52	4.88	0.04	0.06	0.13	1.16
500	1.66	0.63	11.07	0.06	0.07	0.14	1.92
600	2.33	0.72	16.50	0.05	0.08	0.17	2.42
700	3.47	0.84	11.79	0.05	0.08	0.19	1.88
800	4.76	1.00	27.74	0.07	0.10	0.19	2.66
900	7.99	0.85	26.00	0.09	0.12	0.17	3.74
1000	7.97	1.29	25.40	0.10	0.11	0.19	3.22

3.2. Real-world data sets

3.2.1. Letter recognition (STATLOG database)

This dataset is a well-known benchmark, constructed by David J. Slate of Odesta Corporation, Evanston, Illinois. The objective here is to classify each of a large number of black and white rectangular pixel displays as one of the 26 capital letters of the English alphabet. The character images produced were based on 20 different fonts and each letter within these fonts was randomly distorted to produce a file of 20,000 unique images. For each image, 16 numerical attributes were calculated using edge counts and measures

Table 6
CPU test classification time for *I–I* data set

Training set size	1-NN	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	0.82	2.13	0.24	26.62	0.06	0.06	0.08	0.08
200	1.61	2.72	0.52	33.19	0.05	0.04	0.08	0.07
300	2.40	3.39	0.23	35.64	0.05	0.05	0.08	0.08
400	3.21	4.29	0.28	38.55	0.04	0.04	0.08	0.08
500	4.05	5.15	0.35	43.42	0.07	0.07	0.08	0.09
600	4.90	5.84	1.35	45.89	0.04	0.04	0.09	0.08
700	5.65	6.58	1.20	45.34	0.05	0.04	0.09	0.09
800	6.37	8.24	1.75	50.39	0.05	0.05	0.09	0.08
900	11.22	32.11	1.92	51.00	0.04	0.04	0.08	0.08
1000	34.22	31.40	2.31	50.22	0.04	0.04	0.08	0.08

Table 7
Number of prototypes for *I–I* data set

Training set size	1-NN or *K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
100	100	30	50	4	4	3	2
200	200	80	83	3	2	3	2
300	300	30	154	3	3	3	2
400	400	40	238	2	2	3	2
500	500	50	256	6	6	3	2
600	600	240	301	2	2	3	2
700	700	210	323	3	2	3	2
800	800	320	418	3	3	3	2
900	900	360	512	2	2	3	2
1000	1000	300	450	2	2	3	2

Table 8
Test classification error (%) for *I–I* data set

Training set size	1-NN	*K*-NN	LVQ	SGNT	-SGP1	SGP2	GMM1	GMM2
100	19.2	13.6	19.0	20.6	15.6	15.6	**10.7**	14.3
200	17.2	11.7	18.0	19.0	11.3	11.0	**10.3**	11.3
300	15.4	11.1	18.0	16.2	10.4	10.4	**9.9**	10.2
400	15.6	10.9	16.8	18.6	10.0	10.0	**9.9**	10.6
500	15.3	10.4	13.0	16.9	12.1	12.0	**9.8**	10.3
600	14.8	10.7	16.1	16.2	10.1	10.1	**9.8**	10.2
700	14.9	10.6	14.7	16.7	10.2	10.3	**9.9**	10.1
800	14.9	10.4	15.1	16.9	10.8	10.8	**9.9**	10.1
900	15.6	11.1	17.2	17.7	10.0	10.0	**9.9**	10.1
1000	15.5	10.5	16.4	17.3	10.0	10.0	**9.9**	10.0

Bold numbers indicate the minimum test classification error while underlined numbers indicate the second minimum.

of statistical moments. The attributes were scaled and discretized into a range of integer values from 0 to 15. The size of the training set is the first 15,000 items and the resulting model is used to predict the letter category for the remaining 5000. This data set is one of the data sets used in the STATLOG project. For more details and description of methods applied to this data set, see Ref. [14]. In this study the results of different methods applied to this data set in the STATLOG project are summarized in Table 9 and results of SGP1, SGP2, GMM1 and GMM2 methods are shown in Table 10.

3.2.2. Cancer data set

In this breast cancer data set, the goal is to classify a tumor as benign or malignant based on cell descriptions gathered by microscopic examination. Input attributes are for instance the clump thickness, the uniformity of cell size and cell shape, the amount of marginal adhesion and the frequency of bare nuclei. The data set consists of nine inputs, one binary output and 699 examples; 65.5% of the examples are benign, 525 examples were used for training and the remaining 174 examples were used for testing. This data set was obtained from cancer1.dt file from Proben1

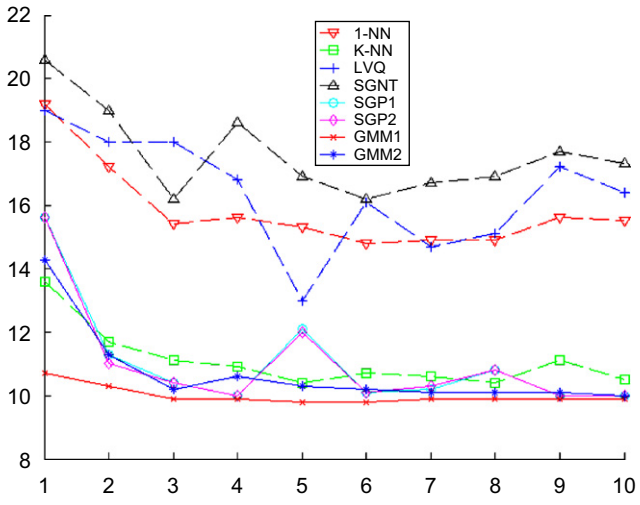

Fig. 11. Test classification error comparison of different methods for *I–I* data set.

Table 9
Results of different methods used in STATLOG project for Letter data set

Method	Classification error (%)	
	Train	Test
Alloc80	6.5	6.4
K-NN	0.0	6.8
LVQ	5.7	7.9
QuaDisc	10.1	11.3
Cn2	2.1	11.5
BayTree	1.5	12.4
NewId	0.0	12.8
IndCart	1.0	13.0
C4.5	4.2	13.2
Dipol92	16.7	17.6
Radial	22.0	23.3
LogDisc	23.4	23.4
Ac2	0.0	24.5
Castle	23.7	24.5
Kohonen	21.8	25.2
Cal5	15.8	25.3
Smart	28.7	29.5
Discrim	29.7	30.2
BackProp	32.3	32.7
Bayes	51.6	52.9
Itrule	58.5	59.4

Table 10
Results of SGP and GMM methods applied to *Letter* data set

Method	Average CPU time		Number of prototypes	Test classification error (%)
	Train	Test		
SGP1	25.38	9.39	1546	**5.94**
SGP2	72.11	4.59	1338	6.68
GMM1	2.33	21.72	502	12.70
GMM2	1.46	2.62	61	7.18

database [15], which was created based on the "breast cancer Wisconsin" problem data set from the UCI repository of machine learning databases [16].

3.2.3. Diabetes data set

This data set is about the diagnosis of diabetes of Pima Indians. Based on personal data (age, number of times pregnant) and the results of medical examinations (e.g. blood pressure, body mass index, result of glucose tolerance test, etc.), the goal is to classify a Pima Indian individual as diabetes positive or negative. The data set consists of eight inputs, one binary output, and 768 examples; 65.1% of the examples are diabetes negative, 576 examples were used for training and the remaining 192 examples were used for testing. This data set was obtained from diabetes1.dt file from Proben1 database [15], which was created based on the "Pima Indians Diabetes" problem data set from the UCI repository of machine learning databases [16].

3.2.4. Thyroid data set

This data set is about the diagnosis of thyroid hypofunction. Based on patient query data and patient examination data, the task is to decide whether the patient's thyroid has overfunction, normal function, or underfunction. The data set consists of 21 inputs, one discrete output, 7200 examples. The class probabilities are 5.1%, 92.6% and 2.3%, respectively; 5400 examples were used for training and the remaining 1800 examples were used for testing. This data set was obtained from thyroid1.dt file from Proben1 database [15], which was created based on the "ann" version of the "thyroid disease" problem data set from the UCI repository of machine learning databases [16].

Tables 11 and 12 show the CPU time elapsed in training and testing for different methods, respectively, while Tables 13 and 14 show comparison of the number of prototypes and the test classification error, respectively.

4. Discussion

It can be noted from the above results that SGP1 and SGP2 require less computational effort and thus are faster than the other methods, both for training and actual classification, with the exception of the STATLOG experiment where GMM2 was somewhat ahead. The improvement in speed compared to all methods other than the GMM methods is almost one order of magnitude in general. Concerning the resulting number of prototypes, SGP2 and GMM2 were mostly better (i.e. they resulted in a smaller number of prototypes). However, SGP1 was not far behind. The classification accuracy of SGP1 and SGP2 is significantly better than LVQ, 1-NN and SGNT, a little better than *K*-NN, and almost tie with GMM1 and GMM2. This is perhaps because the other lagging methods lead to much more prototypes and hence a higher complexity classifier. Not only did the SGP methods beat other prototype methods, but they beat

Table 11
Average CPU training time over all runs with different parameters for *Real World* data sets

Data set	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
Cancer	2.21	5.21	6.51	0.01	0.03	0.32	0.03
Diabetes	2.56	0.68	8.34	0.20	0.23	0.24	0.21
Thyroid	25.87	185.12	94.82	5.00	5.16	22.7	0.14

Table 12
CPU test time for *Real World* data sets

Data set	1-NN	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
Cancer	0.11	0.10	0.03	0.59	0.01	<0.01	<0.01	0.01
Diabetes	0.10	0.11	0.06	0.75	<0.01	<0.01	0.01	0.01
Thyroid	7.44	10.99	1.50	6.9	0.03	<0.01	0.55	0.43

Table 13
Number of prototypes for *Real World* data set

Data set	1-NN or *K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
Cancer	525	52	141	3	2	18	2
Diabetes	576	288	392	22	22	8	3
Thyroid	5400	2700	1112	9	1	41	3

Table 14
Test classification error (%) for *Real World* data set

Data set	1-NN	*K*-NN	LVQ	SGNT	SGP1	SGP2	GMM1	GMM2
Cancer	2.90	1.70	5.17	4.02	**1.15**	**1.15**	**1.15**	1.72
Diabetes	30.20	27.60	30.73	31.25	**27.08**	28.13	**27.08**	31.77
Thyroid	7.00	**6.30**	13.83	9.33	7.28	7.28	14.83	7.28

Bold numbers indicate the minimum test classification error while underlined numbers indicate the second minimum.

(at least SGP1) all 21 classification methods documented in the well-known STATLOG comparison in terms of test classification accuracy, and were among the top ones in terms of training and testing times. Note that STATLOG is one of the few large-scale comparisons of classification performance. An interesting result can be observed for the mixture of Gaussian *Diagonal* data set. The four prototypes result from SGP method represent an approximation for the Gaussian means (Fig. 12). So, overall, the SGP methods are among the best in almost all performance measures, with the GMM methods competitive with them.

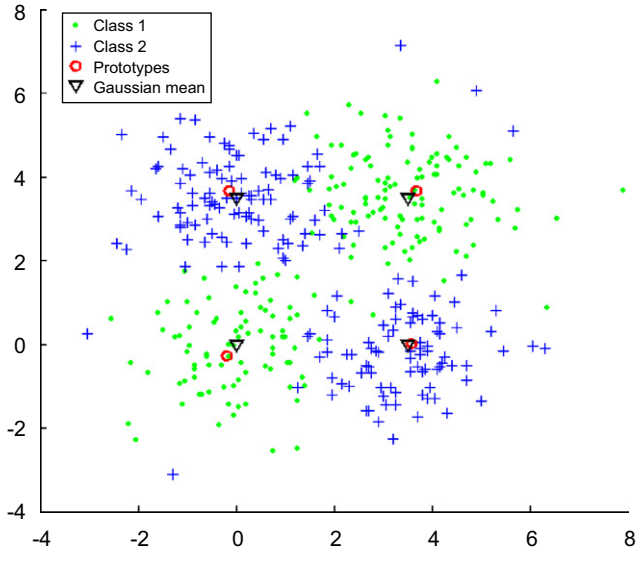

Fig. 12. Resulting prototypes of SGP compared to the Gaussian means for the *Diagonal* data set for 500 training patterns.

5. Conclusions

In this article, we proposed new prototype methods (SGP1, SGP2) for pattern classification problems. The main advantage of these methods is that they learn the number of prototypes required to represent each class region and the prototypes' locations from the training patterns. The idea of

the proposed methods is to start with one group/prototype per class, and successively split groups, shift patterns from one group to another, merge groups, etc according to a proposed algorithm. The proposed approaches have significantly reduced the number of prototypes compared to *K*-NN without sacrificing the classification accuracy. Moreover, they achieve superior performance in terms of classification accuracy, classification speed and number of prototypes compared to LVQ. In addition, the SGP methods achieve superior performance or competitive with the GMM methods in terms of many criteria such as accuracy, speed and the number of prototypes.

References

[1] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second ed., Wiley, New York, 2001.

[2] T. Hastie, R. Tibshurani, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer Series in Statistics, 2001.

[3] T. Kohonen, Self-organization and Associative Memory, third ed., Springer, Heidelberg, Germany, 1989.

[4] W.X. Wen, A. Jennings, H. Liu, Learning a neural tree, International Joint Conference on Neural Networks, Beijing, 1992, pp. 751–756.

[5] H. Inoue, A study of ensemble self-generating neural networks, Ph.D. Thesis, Graduate School of Engineering, Okayama University of Science, 2002.

[6] D. Reilly, L. Cooper, C. Elbaum, A neural model for category learning, Biol. Cybernet. 45 (1982) 35–41.

[7] A. Atiya, S. Hashem, H. Fayed, New hyperspheres for pattern classification, in: Proceedings of the First International Computer Engineering Conference (ICENCO-2004), Cairo, Egypt, December 2004, pp. 258–263.

[8] M. Kositsky, S. Ullman, Learning class regions by the Union of Ellipsoids, in: Proceedings of the 13th International Conference on Pattern Recognition (ICPR), vol. 4, IEEE Computer Society Press, Silver Spring, MD, 1996, pp. 750–757.

[9] S. Salzberg, A nearest hyperrectangle learning method, Mach. Learning 6 (1991) 277–309.

[10] S.Y. Kung, M.W. Mak, S.H. Lin, Biometric Authentication: A Machine Learning Approach, Prentice-Hall, Englewood Cliffs, NJ, 2005.

[11] MATLAB Version 6.5, The MathWorks Inc., 2002.

[12] H. Zhang, G. Sun, Optimal reference subset selection for nearest neighbor classification by tabu search, Pattern Recognition 35 (2002) 1481–1490.

[13] K. Fukunaga, P.M. Narendra, A branch and bound algorithm for computing *k*-nearest neighbors, IEEE Trans. Comput. 24 (1975) 750–753.

[14] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine Learning, Neural and Statistical Classification, Ellis Horwood, New York, NY, 1994.

[15] L. Prechelt, Proben1 A Set of Neural-Network Benchmark Problems, University of Karlsruhe, Germany, 1994 Available FTP: ⟨ira.uka.de/pub/neuron/proben1.tar.gz⟩.

[16] UCI Repository of Machine Learning Database, University of California, Irvine, Department of Information and Computer Science, 1994. Available: ⟨http://www.ics.uci.edu/∼mlearn⟩.

About the Author—SHERIF HASHEM is the Executive Vice President of the Information Technology Industry Development Authority (ITIDA) at (2004-present), Egypt (http://www.itida.gov.eg/; http://www.mcit.gov.eg/). He is also an Associate Professor at the Faculty of Engineering, Cairo University, Egypt (2001-present).

Dr. Hashem received a B.Sc. in Communication & Electronic Engineering (Distinction with honor) and an M.Sc. in Engineering Mathematics from Cairo University (Egypt—1985 and 1988), and a Ph.D. in Industrial Engineering from Purdue University (USA—1993). He also completed the Senior Executive Program at Harvard Business School (USA—2001).

Dr. Hashem authored and co-authored more than 30 articles and book chapters in the areas of information technology, e-commerce, computational intelligence, and operations research, with applications in engineering, energy, environment, and computer sciences.

Dr. Hashem held research positions at Purdue University (USA, 1992–1993), West Virginia University (USA, 1997), Honeywell Sensor and Systems Development Center (USA, 1991–1992), and was a Postdoctoral Research Fellow of the US-Department of Energy working at Pacific Northwest National Laboratory—PNNL (USA, 1994–1995).

Dr. Hashem is currently the Executive Vice President of the ITIDA responsible for E-Business Initiatives and E-Signature regulations. Prior to joining ITIDA in December 2004, Dr. Hashem was the Director of Information Society Development Office (ISDO) and the manager of the E-Business Program at the Ministry of Communications and Information Technology (MCIT). He was also the coordinator of the national committee that prepared Egypt's electronic signature law, and the Director of the Egyptian Focal Point for the EUropean MEDiterranean Information Society (EUMEDIS) Initiative. From 1995 to 1999, Dr. Hashem worked as a senior consultant and senior project manager at the Egyptian Cabinet Information and Decision Support Center (IDSC), and at the Regional Information Technology and Software Engineering Center (RITSEC), where he managed several information technology projects in various sectors including electronic commerce, tourism, culture heritage preservation, healthcare, and community empowerment, including the award winning project: Egypt Information Highway (http://www.idsc.gov.eg/).

Dr. Hashem received several awards and recognition including: the *Global Bangemann Challenge Award* (from the *King of Sweden*: Stockholm—1999), the *International G7-GIP Information Society Award* (G7/8: Warsaw—1997), the *Excellence in Teaming Award* (PNNL—1994), and the *Spirit Award* (Honeywell—1992). Dr. Hashem is listed in MARQUIS Who's Who in the World (16th edition—1999 & Millennium edition—2000) and in Who's Who in America—Science and Engineering (fifth edition—2000).

He is a founding member of the Internet Society of Egypt (ISE), and a member of the Institute of Electrical and Electronics Engineers (IEEE), and Egyptian Syndicate for Engineers. Dr. Hashem is a member of the Board of Directors of Egypt's National Postal Organization.

About the Author—AMIR F. ATIYA received his B.Sc. degree in 1982 from Cairo University and the M.S. and Ph.D. degrees in 1986 and 1991 from Caltech, Pasadena, CA, all in Electrical Engineering.

Dr. Atiya is currently an Associate Professor at the Department of Computer Engineering, Cairo University.

He held various appointments in industry, such as in QANTXX, Houston, TX, Simplex Technology, Hong Kong, Countrywide, California, and Dunn Capital Management, Florida.

Recently, from 1997 to 2001 he was a Visiting Associate in Caltech.

His research interests are in the areas of neural networks, learning theory, pattern recognition, Monte Carlo methods, data compression, and optimization theory. His most recent interests are the application of learning theory and computational methods to financial prediction and to communications networks prediction problems. Dr. Atiya received the highly regarded Kuwait Prize in 2005. He also received the Egyptian State Prize for Best Research in Science and Engineering, in 1994, and received the Young Investigator Award from the International Neural Network Society, in 1996. Currently, he is an Associate Editor for IEEE Transactions on Neural Networks, since 1998.

He was a Guest Co-editor of the special issue of IEEE Transactions on Neural Networks on 'Neural Networks in Financial Engineering', July 2001, and was a Guest Co-editor of the special issue of IEEE Transactions on Neural Networks on 'Adaptive Learning Systems in Communications Networks', September 2005.
He served on the organizing and program committees of several conferences, including being Program Co-Chair for the IEEE Conference on Computational Intelligence in Financial Engineering (CIFER-03), March 2003, Hong Kong.

About the Author—HATEM A. FAYED received his B.Sc. degree in Communication & Electronic Engineering (Distinction with honor) in 1993 from Cairo University and the M.Sc. in Engineering Mathematics and Physics in 2000 from Cairo University. He received courses in image processing at Electrical and Computer Engineering department, University of Miami, USA 2001and received his Ph.D. degree in Engineering Mathematics and Physics in 2005 from Cairo University. Dr. Fayed worked as a programmer/developer/designer for database systems at Information and Decision Support Center (IDSC), Egypt 1994–1996. Dr. Fayed is currently an Assistant Professor at the Engineering Mathematics and Physics Department, Cairo University.
His research interests are in the areas of neural networks, learning theory, pattern recognition, Time series forecasting, data compression, and optimization theory.
Dr. Fayed co-authored four articles in the areas of neural networks and pattern recognition.