Dynamic generation of prototypes with self-organizing feature maps for classifier design

Arijit Laha, Nikhil R. Pal*

Electronics and Communication Science Unit, Indian Statistical Institute, Calcutta 700 035, India

Received 2 June 1999; accepted 1 October 1999

Abstract

We propose a new scheme for designing a nearest-prototype classifier using Kohonen's self-organizing feature map (SOFM). The net starts with the minimum number of prototypes which is equal to the number of classes. Then on the basis of the classification performance, new prototypes are generated dynamically. The algorithm merges similar prototypes and deletes less significant prototypes. If prototypes are deleted or new prototypes appear then they are fine tuned using Kohonen's SOFM algorithm with the winner-only update strategy. This adaptation continues until the system satisfies a termination condition. The classifier has been tested with several well-known data sets and the results obtained are quite satisfactory.

Keywords: Nearest-prototype classifier; Dynamic prototype generation; Self-organizing feature map; Split-merge technique

1. Introduction

Kohonen's self-organizing feature map (SOFM) has been successfully used in numerous fields of application such as speech recognition [1], robotics [2,3], industrial process control [5], image compression [4], etc. Designing of classifiers [6] and other pattern recognition systems based on SOFM [7] are some of the most successful areas of application. SOFM [8,9] has the interesting property of achieving a distribution of the weight vectors that approximates the distribution of the input data. This property of the SOFM can be exploited for designing nearest prototype classifiers. Here we propose a new approach for this. Although our training data are labeled, the SOFM is trained without using the class information. When the training is over, the weight vectors are converted into labeled prototypes of a classifier using the class information. The performance of the classifier is then evaluated. Based on the evaluation results a tuning step consisting of deletion, merging, splitting and retraining of the net is performed. The evaluation and tuning are repeated until the number of prototypes stabilizes or the performance of the classifier reaches a satisfactory level. In case of highly overlapped class boundaries usually it is very difficult to estimate the adequate number of prototypes. Small number of prototypes suffer from large error rates, while at the other extreme a large number of prototypes make the system expensive. Here the tuning strategy is designed to strike a compromise between the classification performance and the number of prototypes for such data. These prototypes can be used to generate fuzzy rules for a fuzzy rule based pattern recognition system.

2. Self-organizing feature map

We view the self-organizing feature map as an algorithmic transformation $A^{\mathrm{D}}_{\mathrm{SOFM}}: R^p \rightarrow V(R^q)$ that is often advocated for visualization of metric-topological relationships and distributional density properties of feature vectors (signals) $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in R^p [9].

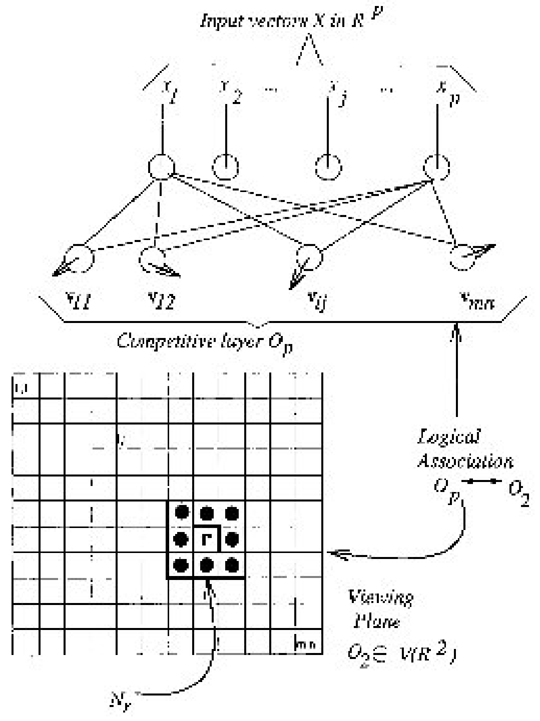

Fig. 1. The SOFM architecture.

SOFM is implemented through a neural-like network architecture as shown in Fig. 1 and it is believed to be similar in some ways to the biological neural network.

The visual display produced by $A^{\mathrm{D}}_{\mathrm{SOFM}}$ helps to form hypotheses about topological structure present in X. Although, in this article we concentrate on $(m \times n)$ displays in R^2, in principle X can be transformed onto a display lattice in R^q for any q. In practice, visual display can be made only for $q \leqslant 3$ and are usually made on a linear or planar configuration arranged as a rectangular or hexagonal lattice.

As shown in Fig. 1 input vectors $\mathbf{x} \in R^p$ are distributed by a fan-out layer to each of the $(m \times n)$ output nodes in the competitive layer. Each node in this layer has a weight vector (prototype) $\mathbf{w}_{ij}$ attached to it. Let $O_p = \{\mathbf{w}_{ij}\} \subset R^p$ denote the set of $m \times n$ weight vectors. O_p is (logically) connected to a display grid $O_2 \subset V(R^2)$. (i, j) in the index set $\{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\}$ is the logical address of the cell. There is a one-to-one correspondence between the $m \times n$ p-vectors $\mathbf{w}_{ij}$ and the $m \times n$ cells $(\{i, j\})$, i.e., $O_p \leftrightarrow O_2$. In the literature display cells are sometimes called nodes, or even neurons, and we shall use both of them.

The feature mapping algorithm starts with (usually) a random initialization of the weight vectors $\mathbf{w}_{ij}$. For notational clarity we suppress the double subscripts. Now let $\mathbf{x} \in R^p$ enter the network and let t denote the current iteration number. Find $\mathbf{w}_{r,t-1}$, that best matches $\mathbf{x}$ in the sense of minimum Euclidian distance in R^p. This vector has a (logical) "image" which is the cell in O_2 with subscript r. Next, a topological (spatial) neighborhood $N_r(t)$ centered at r is defined in O_2, and its display cell neighbors are located. A 3×3 window, $N(r)$, centered at r corresponds to updating nine prototypes in R^p. Finally, $\mathbf{w}_{r,t-1}$ and the other weight vectors associated with cells in the spatial neighborhood $N_t(r)$ are updated using the rule

$$\mathbf{w}_{i,t} = \mathbf{w}_{i,t-1} + h_{ri}(t)(\mathbf{x} - \mathbf{w}_{i,t-1}). \tag{1}$$

Here r is the index of the "winner" prototype

$$r = \underbrace{\arg\min}_{i} \{\|\mathbf{x} - \mathbf{w}_{i,t-1}\|\} \tag{2}$$

and $\| * \|$ is the Euclidian norm on R^p. The function $h_{ri}(t)$ which expresses the strength of interaction between cells r and i in O_2 usually decreases with t, and for a fixed t it decreases as the distance (in O_2) from cell r to cell i increases. $h_{ri}(t)$ is usually expressed as the product of a learning parameter α_t and a lateral feedback function $g_t(\mathrm{dist}(r, i))$. A common choice for g_t is $g_t(\mathrm{dist}(r, i)) = \exp^{-\mathrm{dist}^2(r,i)/\sigma_t^2}$. α_t and σ_t both decrease with time t. The topological neighborhood $N_t(r)$ also decreases with time. This scheme, when repeated long enough, usually preserves spatial order in the sense that weight vectors which are metrically close in R^p generally have, at termination of the learning procedure, visually close images in the viewing plane. We next provide a schematic description of the algorithm.

```
Algorithm $A^{\mathrm{d}}_{\mathrm{SOFM}}$(Kohonen):
Begin
  Input $X$   /** unlabeled data set $X = \{\mathbf{x}_i \in R^p$:
                 $i = 1, 2, \ldots, N\}$ **/
  Input $m, n$   /** the display grid size, a rectangle of size
                 $m \times n$ is assumed **/
  Input maxstep   /** maximum number of updating
                 steps **/
  Input $N_0$   /** initial neighborhood size **/
  Input $\alpha_0$   /** the initial step size (learning coefficient)
                 **/
  Input $\sigma_0$ and $\sigma_f$   /** parameters to control effective
                 step size **/
/** Learning phase **/
  Randomly generate initial weight vectors
    $\{\mathbf{w}_{ij}, i = 1, 2, \ldots, m; j = 1, 2, \ldots, n, \mathbf{w}_{ij} \in R^p\}$
  $t \leftarrow 0$
  While($t <$ maxstep)
    Select randomly $\mathbf{x}(t)$ from $X$;

      find $r = \underbrace{\arg\min}_{i} \{\|\mathbf{x}(t) - \mathbf{w}_i(t)\|\}$

      /** $r$ and $i$ stands for two-dimensional indices that
      uniquely identify a weight vector in $O_p$ **/
      $\mathbf{w}_i(t+1) \leftarrow \mathbf{w}_i(t) + \alpha_t g_t(\text{dist}(r,i))[\mathbf{x}(t) - \mathbf{w}_i(t)]\ \forall i \in N_t(r)$
      $\mathbf{w}_i(t+1) \leftarrow \mathbf{w}_i(t)\ \forall i \notin N_t(r)$
        /** dist(i, j) is the Euclidian distance between the
        centers of nodes r and i on the display lattice, $g_t(d)$
        is the lateral feedback function, usually
        $g_t(d) = e^{-d^2/\sigma_t^2}$ **/
      $t \leftarrow t+1$
      $\alpha_t \leftarrow \alpha_0(1 - t/\text{maxstep})$
      $N_t \leftarrow N_0 - t(N_0 - 1)/\text{maxstep}$
      $\sigma_t \leftarrow \sigma_0 - t(\sigma_0 - \sigma_f)/\text{maxstep}$
      /** there are other ways to readjust $\alpha_t, N_t$ and $\sigma_t$,
      and many choices
      for $g_t$ **/
  End While
/** Display phase **/
  For each $\mathbf{x} \in X$ find
      $r = \underbrace{\arg\min}_{i} \{\|\mathbf{x} - \mathbf{w}_i\|\}$, and mark the associated
      cell r in $O_2$.
End.
```

3. Labeling of SOFM prototypes

In this investigation we use a 1-D SOFM, but the algorithm can be extended to 2-D SOFM also. First we train a one-dimensional SOFM using the training data, of course, without using the class information of the input data. Initially the number of nodes in the SOFM is the same as the number of classes c. This is motivated by the fact that the smallest number of prototypes that may be required is equal to the number of classes. At the end of the training the weight vector distribution of the SOFM will reflect the distribution of the input data. These unlabeled prototypes are then labeled using class information. For each of N input feature vectors we identify the prototype closest to it, i.e., the winner node. Since no class information is used during the training, it is only natural that some prototypes may become the winner for data from more than one classes. For each prototype $\mathbf{v}_i$ we compute a score D_{ij}, which is the number of data points from class j to which $\mathbf{v}_i$ is the closest prototype. Due to strong interaction among the neighboring nodes of the SOFM during the training some prototypes may be so placed that for no input data they are the closest prototypes; i.e., D_{ij} is 0 for all j. Naturally we reject such prototypes. For the remaining prototypes the class label C_i of the prototype $\mathbf{v}_i$ is determined as

$$C_i = \underbrace{\arg\max}_{j} \mathrm{D}_{ij}. \tag{3}$$

The scheme will assign a label to each of the c prototypes, but such a set of prototypes may not classify the data satisfactorily. For example, from (3) it is clear that $\sum_{j \neq C_i} D_{ij}$ data points will be wrongly classified by the prototype $\mathbf{v}_i$. Hence we need further refinement of the initial set of prototypes $V_0 = \{\mathbf{v}_{10}, \mathbf{v}_{20}, \ldots, \mathbf{v}_{c0}\} \subset R^p$. which we do next.

3.1. Refinement of prototypes

The prototypes generated by the SOFM algorithm represent the overall distribution of the input data. A set of prototypes useful for classification job must be capable of dealing with class specific characteristics (such as class boundaries) of the data. We present a strategy of modifying the initial set of prototypes V_0 leading to the enhancement of performance of the classifier. This process of modification is repeated till the number of prototypes and their performance stabilize within an acceptable level.

On mth iteration the prototype set V_{m-1} from previous iteration is used to generate the new set of prototypes V_m. The labeled prototypes V_{m-1} are used to classify a set of training data and their performance is monitored. Let W_i be the number of training data to which prototype $\mathbf{v}_i$ is the closest one. Let $S_i = \max_j\{D_{ij}\} = D_{iC_i}$. Thus when $\mathbf{v}_i$ is labeled as a prototype for class C_i, S_i training data points will be correctly classified by $\mathbf{v}_i$ and $F_i = \sum_{j \neq C_i} D_{ij}$ data points will be incorrectly classified. Thus,

$$W_i = S_i + F_i$$

and

$$W_i = \sum_j D_{ij}.$$

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the set of training data and N_j be the number of training data from class j. The refinement stage uses $(c+1)$ parameters, a global retention parameter α and a set of class-wise retention parameters β_k (one for each class), to evaluate the performance of each prototype. α and β_k are computed dynamically (not fixed) for mth iteration using the following formula:

$$\alpha_m = \frac{1}{K_1|V_{m-1}|},$$

$$\beta_{mk} = \frac{1}{K_2|V_{mk}^{*}|},$$

where $V_{mk}^{*} = \{\mathbf{v}_i \mid \mathbf{v}_i \in V_{m-1}, C_i = k\}$, K_1 and K_2 are two user supplied constants whose choice is discussed in detail later. Thus, actually the algorithm uses 2 (not $c+1$) user supplied parameters.

Based on the result of the evaluation, different operations may be performed on the prototypes to generate a new set of prototypes. We use the following operations in our algorithm.

Merging of a prototype with respect to a class: Let a prototype $\mathbf{v}_i$ represent D_{ik} training data from class k. To merge $\mathbf{v}_i$ w.r.t. class k we identify the prototype $\mathbf{v}_l$ closest to $\mathbf{v}_i$ where $C_l = k$ (i.e., $\mathbf{v}_l$ also is a prototype for class k). Let us denote X_{ij} as the set of training data vectors from class j whose nearest prototype is $\mathbf{v}_i$. When we merge $\mathbf{v}_i$ with $\mathbf{v}_l$ w.r.t. class k, $\mathbf{v}_l$ is updated according to the equation,

$$\mathbf{v}_l = \frac{W_l \mathbf{v}_l + \sum_{\mathbf{x} \in X_{ik}} \mathbf{x}}{W_l + D_{ik}}. \tag{4}$$

Note that we do not say here, when to merge. This will be discussed later.

Modifying a labeled prototype: A prototype $\mathbf{v}_i$ is modified according to the following equation:

$$\mathbf{v}_i = \frac{\sum_{\mathbf{x} \in X_{iC_i}} \mathbf{x}}{D_{iC_i}}. \tag{5}$$

Splitting a prototype: A prototype $\mathbf{v}_i$ is split into r new prototypes for r different classes according to the following rule. For each of r new prototypes $\mathbf{v}_l$ of class C_l we compute

$$\mathbf{v}_l = \frac{\sum_{\mathbf{x} \in X_{iC_l}} \mathbf{x}}{D_{iC_l}}. \tag{6}$$

The prototype $\mathbf{v}_i$ is deleted. So after the splitting the number of prototypes is increased by $r-1$.

Deleting a prototype: The prototype $\mathbf{v}_i$ is deleted so that the number of prototypes is reduced by one.

Now we are in a position to schematize the evaluation and enhancement strategy for the prototypes as follows.

```
Repeat for all $\mathbf{v}_i \in V_{m-1}$ until termination condition is satisfied.

  If $W_i \neq D_{iC_i}$ and $W_i < \alpha N$ and there is at least another prototype for class $C_i$
     then delete $\mathbf{v}_i$. (Global deletion)
  /*If a prototype is not a pure one (i.e., it represents data from more than one classes) and does not represent a reasonable number of points, it fails to qualify to become a prototype. However if there is no other prototype for class $C_i$ the prototype is retained */

  Else if $W_i > \alpha N$ but $D_{ij} < \beta_{mj} N_j$ for all classes
     then merge $\mathbf{v}_i$ for the classes for which $D_{ij} > 0$ and delete $\mathbf{v}_i$.
     (Merge and delete)
  /* The prototype represents a reasonable number of points, but not a reasonable number of points from any particular class so that it can qualify as a prototype for a particular class. But we cannot ignore the prototype completely. We logically first split $\mathbf{v}_i$ into $s$ prototypes $\mathbf{v}_{i1}, \mathbf{v}_{i2}, \ldots, \mathbf{v}_{is}$, $s \leqslant c$, $s$ is the total number of classes for which $D_{ij} > 0$, and then merge $\mathbf{v}_{ij}$ to its closest prototype from class $j$. $\mathbf{v}_i$ is then deleted. */

  Else if $W_i > \alpha N$ and $D_{iC_i} > \beta_{mC_i} N_{C_i}$ but $D_{ij} < \beta_{mj} N_j$ for all $j \neq C_i$
     then merge $\mathbf{v}_i$ with respect to all the classes other than $C_i$ for which $D_{ij} > 0$ using (4) and modify $\mathbf{v}_i$ using (5). (Merge and modify)
  /* The prototype represents points from more than one classes, however, the points from one class only are well represented by the prototype. According to our labeling scheme the prototype is labeled with the most represented class. Thus we merge $\mathbf{v}_i$ with respect to the classes other than $C_i$ using (4) and then modify $\mathbf{v}_i$ by (5). */

  Else if $W_i > \alpha N$ and $D_{ij} > \beta_{mj} N_j$ for more than one class
     then merge $\mathbf{v}_i$ w.r.t. classes for which $D_{ij} < \beta_{mj} N_j$ by (4) and split $\mathbf{v}_i$ into new prototypes for the classes for which $D_{ij} > \beta_{mj} N_j$ by (5). Add these new prototypes to the new set of prototypes $V_m$.
     (Merge and split)
  /* The prototype represents points reasonably well from more than one classes. So we merge the prototype with respect to the classes whose data are not represented reasonably well and split the prototype into one for each class whose data are reasonably well represented by $\mathbf{v}_i$. */

Let $V_m$ be the union of the unaltered prototypes of $V_{m-1}$ and the modified as well as the new prototypes.
Run the SOFM algorithm on $V_m$ with winner-only update (i.e., no neighbor is updated) strategy using the same training data as input.
/* At this stage we want only to fine tune the prototypes. If the neighbors also are updated the prototypes again might migrate to represent points from more than one class. */
```

Termination conditions

The algorithm may terminate under any one of the following three conditions.

(i) Satisfactory recognition score defined in terms of percentage of correct classifications (ε).
(ii) Stability of prototypes.
(iii) A maximum number of iterations ($I_{\max}$) reached.

Proper use of condition (i) requires some knowledge of the data. However, even if we do not have the same, we can always set a high (conservative) percentage for (ε), say 95%.

Condition (ii) can be checked by a parameter δ_p using the following condition:

$$\frac{||V_{m-1}| - |V_m||}{|V_{m-1}|} < \delta_p, \tag{7}$$

where $|V_{m-1}|$ is the number of prototypes in V_m.

Thus the algorithm terminates when between two successive iterations the number of prototypes do not change significantly.

Condition (iii) is used to protect against infinite looping of the algorithm for some data with highly overlapped structures for which the chosen values of ε and δ_p may not be reachable.

4. Results

Several data sets have been used to judge the performance of the algorithm. But we report here results for five data sets: *Iris*, *Glass*, *Breast Cancer*, *Vowel* and *Norm4*. Iris data [10] have 150 points in four dimensions that are from three classes each with 50 points. Glass data [11] consist of 214 samples with nine attributes from six classes. Breast cancer [12] data have 569 points in 30 dimensions from two classes. The vowel [15] data set consists of 871 samples of discrete phonetically balanced speech samples for the Telugu vowels in consonant–vowel nucleus–consonant (CNC) form. These samples are generated from three male informants (in the age group of 25–30 yr) on an AKAI-type recorder. The spectrographic analysis was done on a Kay Sonograph Model 7029-A. The data have three features as the first three formant frequencies. The data set Norm4 [13] is a sample of 800 points consisting of 200 points each from the four components of a mixture of 4 class 4-variate normals. All our reported results are obtained on the entire data sets.

Table 1 summarizes the classification performances. We used the values $K_1 = 3$, $K_2 = 6$, $\delta_p = 0.2$, $\varepsilon = 95\%$ and $I_{\max} = 10$.

It is well known that classes 2 and 3 of Iris have some overlap and the typical re-substitution error with a nearest-prototype classifier defined by three prototypes obtained by some clustering algorithm is 15–16 (i.e., about 10% error with three prototypes). Our algorithm terminated with seven prototypes in three iterations. The performance of the proposed system with seven prototypes is quite good resulting only in 2.66% error.

Breast cancer data have been used in Ref. [12] to train a linear programming-based diagnostic system by a variant of multisurface method (MSM) called MSM-tree and about 97.5% accuracy was obtained. Breast cancer data of a similar kind have also been used in a recent study [14] with 74.0% accuracy with **100** rules. Our classifier could achieve as low as 11.07% error with only five prototypes and it is quite good.

Table 1
Performance of the classifier for different data sets

Data set	Size	No. of prototypes		No. of iterations	% of error
		Initial	Final		
Iris	150	3	7	4	2.66%
Glass	214	6	30	7	21.29%
Breast cancer	569	2	5	6	11.07%
Vowel	871	6	15	5	21.01%
Norm4	800	4	4	1	3.75%

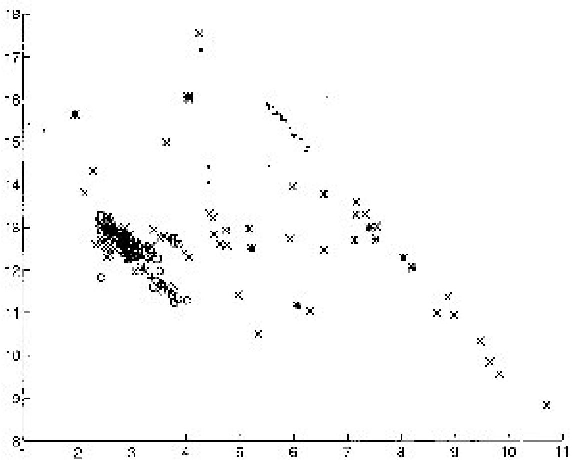
Fig. 2. Scatterplot of the glass data along two most significant principal components.

Glass data shows a high percentage of error; this is possibly unavoidable, because a scatterplot (Fig. 2) of the two principal components shows that the data for class 3 are almost randomly distributed among the data points from other classes. In fact the points from class 3 (represented by +) are not visible in the scatterplot. In Ref. [14] the recognition score for the glass data is 64.4%, i.e., about 35% error. Our classifier could realize more than 78% accuracy with 30 prototypes generated in seven iterations of the algorithm.

Although the vowel data set has three features, we used only the first two features. Bayes classifier for this data set [16] gives an overall recognition score of 79.2%. Fig. 3, the scatterplot of vowel data depicts that there are substantial overlap among different classes and hence some misclassification is unavoidable. The proposed classifier could achieve nearly 79% correct classification with 15 prototypes.

The performance on Norm4 [13] with only four prototypes, i.e., one prototype/class is excellent too. In this case the SOFM based classifier could achieve up to 96% accuracy with only four prototypes.

5. Conclusions

We have proposed a simple but powerful approach of finding a set of reliable prototypes for designing

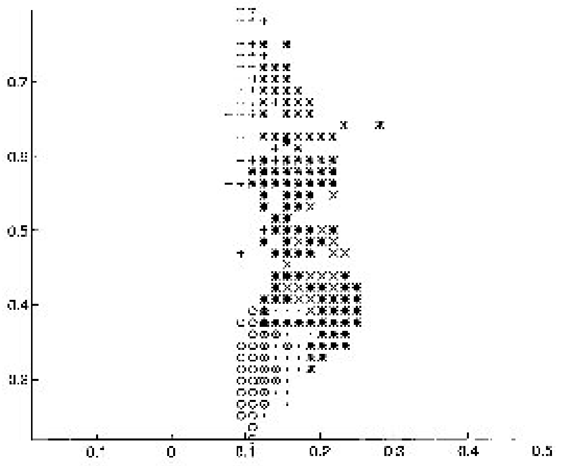

Fig. 3. Scatterplot of the vowel data.

nearest-prototype classifiers. The algorithm first finds a set of representative prototypes from the training data using SOFM disregarding the class information. These prototypes are labeled following a "most-likely class" heuristic. Subsequent stages of tuning cycles fine-tune the prototype set to realize a better class discrimination. Depending on the performance of the classifier prototypes are deleted, merged, modified or split. The retention parameters try to strike a compromise between error rate and the number of the prototypes. Global retention parameter $\alpha(K_1)$ prevents uncontrolled increase in the number of prototypes while the class-wise retention parameters $\beta_k(K_2)$ try to generate prototypes as pure as possible resulting in an increase in the number of prototypes. Proper choice of K_1 and K_2 is needed for balancing the opposing tendencies generated by α and β_ks. It is found that for data with well separated classes (such as Norm4 used here) the process is comparatively insensitive to the change of values of K_1 and K_2, but for data with highly overlapped classes (like Glass and Vowel) the performance of the system varies considerably with the change of value of K_1 and K_2, especially K_2. The process, to some extent, depends on δ_p also. Further investigation is required to provide a guideline for selection of these parameters. Since, the proposed scheme can generate a small number of good prototypes for 1-NP classifier, they can be used to extract fuzzy rules for classifier design also. This is currently under investigation.

References

[1] T. Kohonen, K. Torkkola, M. Shozakai, J. Kangas, O. Venta, Microprocessor implementation of a large vocabulary speech recognizer and phonetic typewriter for Finnish and Japanese, Proceedings of the European Conference of Speech Technology, Edinberg, 1987, pp. 377–380.

[2] D.H. Graf, W.R. Lalonde, A neural controller for collision-free movement of general robot manipulators, Proceedings of the IEEE International Conference on Neural Networks, ICNN-88, San Diego, CA, 1988, pp. I-77–I-80.

[3] D.H. Graf, W.R. Lalonde, Neuroplanners for hand/eye coordination, Proceedings of the Internationl Joint Conference on Neural Networks, IJCNN 89, Washington, DC, 1989, pp. II-543–II-548.

[4] N.M. Nasarabadi, Y. Feng, Vector quantization of images based on Kohonen self-organizing feature maps, Proceedings of the IEEE International Conference on Neural Networks, ICNN-88, San Diego, CA, 1988, pp. I-101–I-108.

[5] K.M. Marks, K.F. Goser, Analysis of VLSI process data based on self-organizing feature maps, Proceedings Neuro-Nimes'88, Nimes, France, pp. 337–347.

[6] S. Mitra, S.K. Pal, Self-organizing neural network as a fuzzy classifier, IEEE Trans. System Man. Cybernet. 24 (3) (1994) 385–398.

[7] Z. Chi, J. Wu, H. Yan, Handwritten numeral character recognition using self-organizing maps and fuzzy rules, Pattern Recognition 28 (1) (1995) 59–66.

[8] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.

[9] J.C. Bezdek, N.R. Pal, A note on self-organizing semantic maps, IEEE Trans. Neural Networks 6 (5) (1995) 1029–1036.

[10] E. Anderson, The IRISes of the Gaspe peninsula, Bull. Amer. IRIS Soc. 59 (1935) 2–5.

[11] R.C. Holte, Very simple classification rules perform well on most commonly used data sets, Mach. Learning 11 (1993) 63–91.

[12] O.L. Mangasarian, W.N. Street, W.H. Wolberg, Breast cancer diagnosis and prognosis via linear programming, Oper. Res. 43 (4) (1995) 570–577.

[13] N.R. Pal, J.C. Bezdek, On cluster validity for the fuzzy c-means model, IEEE Trans. Fuzzy Systems 3 (3) (1995) 370–379.

[14] H. Ishibuchi, T. Nakashima, T. Murata, Performance evaluation of fuzzy classifier systems for multi-dimensional pattern classification problems, IEEE Trans. SMC 29 (5) (1999) 601–618.

[15] S.K. Pal, D. Dutta Majumder, Fuzzy sets and decision making approaches in vowel and speaker recognition, IEEE Trans. System Man. Cybernet SMC-7 (1977) 625–629.

[16] S.K. Pal, S. Mitra, Multilayer perceptron, fuzzy sets, and classification, IEEE Trans. Neural Networks 3 (5) (1992) 683–697.

About the Author—ARIJIT LAHA obtained B.Sc. with honors in physics and M.Sc. in physics from the University of Burdwan in 1991 and 1993, respectively. He obtained his M.Tech. (Comp. Sc.) from the Indian Statistical Institute in 1997. From August 97 to May 98 he has worked in Wipro Infotech Global R& D as a senior software engineer. Currently he is a software engineer in the Electronics and Communication Sciences Unit of the Indian Statistical Institute, Calcutta and involved in a real-time expert system development project. His research interests include pattern recognition, neural networks, fuzzy systems and expert systems.

About the Author—NIKHIL R. PAL obtained B.Sc. with honors in physics and Master of Business Management from the University of Calcutta in 1979 and 1982, respectively. He obtained M. Tech. (Comp. Sc.) and Ph.D. (Comp. Sc.) from the Indian Statistical Institute in 1984 and 1991, respectively. Currently he is a Professor in the Electronics and Communication Sciences Unit of the Indian Statistical Institute, Calcutta. From Sept. 1991 to Feb. 1993, July 1994 to Dec. 1994, Oct 1996 to Dec. 1996 he was with the Computer Science Department of the University of West Florida. He was a guest faculty of the University of Calcutta also. His research interest includes image processing, pattern recognition, fuzzy sets theory, measures of uncertainty, neural networks, genetic algorithms, and fuzzy logic controllers. He has coauthored a book titled *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Kluwer Academic Publishers. He is an associate editor of the *International Journal of Fuzzy Systems*, *International Journal of Approximate Reasoning* and *IEEE Transactions on Fuzzy Systems*.