Presupervised and Postsupervised Prototype Classifier Design

Ludmila I. Kuncheva and James C. Bezdek, *Fellow, IEEE*

***Abstract*—We extend the nearest prototype classifier to a *generalized nearest prototype classifier* (GNPC). The GNPC uses "soft" labeling of the prototypes in the classes, thereby encompassing a variety of classifiers. Based on how the prototypes are found we distinguish between *presupervised* and *postsupervised* GNPC designs. We derive the conditions for optimality (relative to the standard Bayes error rate) of two designs where prototypes represent: 1) the components of class-conditional mixture densities (presupervised design) or 2) the components of the unconditional mixture density (postsupervised design). An artificial data set and the "satimage" data set from the database ELENA are used to experimentally study the two approaches. A radial basis function (RBF) network is used as a representative of each GNPC type. Neither the theoretical nor the experimental results indicate clear reasons to prefer one of the approaches. The postsupervised GNPC design tends to be more robust and less accurate than the presupervised one.**

***Index Terms*— Mixture modeling, prototype classifiers, prototype selection, RBF neural networks, supervised and unsupervised designs.**

I. Introduction

LET $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset \Re^d$ be a set of labeled training data. There are two approaches to prototype classifier design with X. We can use the labels of the data *during* training to guide an algorithm toward "good" (labeled) prototypes; or we can ignore the labels during training, and use them *a posteriori* to label the prototypes. We call these two schemes *presupervised* and *postsupervised* learnining. The use of labels during training (presupervision) seems intuitively more reasonable than postsupervised training but there is little evidence that this is the case.

The notion of prototype classification is not limited to finding the nearest prototype and assigning its class label to the object to be classified. Here we consider a broader framework called a *generalized nearest prototype classifier (GNPC)* which encompasses a variety of classifiers. It assigns a label to a new data point on the basis of some subest of the prototypes and their labels in the classes. The way of *finding* prototypes is not specified by the GNPC definition.

Among classifiers that can be represented as a GNPC are the classical nearest mean and nearest neighbor designs [8], [22], some types of *radial basis function (RBF)* networks [5], [19], [23], a class of fuzzy if–then systems [15], *learning vector quantization (LVQ)* classifiers [11], [12], [14], edited nearest neighbor rules [6], fuzzy nearest neighbor rules [1], [13], [26], multiple prototype classifiers [2], and a number of neural-network implementations of the nearest neighbor design [7], [17], [26]. Each of these has specific strategies and algorithms for finding the prototypes. The diagram in Fig. 1 groups the GNPC's into presupervised and postsupervised designs with respect to the way the prototypes are found and labeled. In each group we distinguish GNPC's with crisp and noncrisp (soft) labels for the prototypes. The connection between models and groups of models that fit within the GNPC framework has been addressed many times [3], [20]. Ripley [20] points out that some neural network models are just renamed kernel or Parzen classifiers. Asymptotically (when the number of prototypes/hidden nodes tends to infinity) the models coincide. Here we are interested in the finite-sample case; then different designs can offer significantly different performance.

We study two semiparametric designs (one from each GNPC group) and show that presupervised mixture modeling needs one type of assumption (*decomposition*) while the dual postsupervised design needs two types of assumption (*decomposition* and *homogeniety*) for optimality relative to the standard Bayes error rate (Bayes-optimality [8], p. 16). Two RBF networks are chosen as representatives: the RBF trained by orthogonal least squares (OLS RBF) [4], a presupervised GNPC; and the RBF trained by clustering followed by nonnegative least squares (NNLS RBF) [19], a postsupervised GNPC.

Section II gives our definition of the GNPC. In Sections III and IV we define the mixture models and derive conditions for Bayes-optimality of the GNPC. Experimental results with OLS RBF and NNLS RBF on two data sets (one synthetic and one real) are given in Section V. Section VI contains our conclusions.

II. The Generalized Nearest Prototype Classifier (GNPC)

We consider c mutually exclusive classes with crisp labels $I_{\text{crisp}} = \{\mathbf{e}_1, \cdots, \mathbf{e}_c\}$, where I_{crisp} is the canonical basis of $\Re^c$. Objects associated with class i are labeled by the vector $\mathbf{e}_i = [e_{1i}, \cdots, e_{ci}]^T$, $e_{ji} = 1$ if $j = i$, and 0, otherwise. Let $\Re^d$ be the feature space. Any function $D : \Re^d \rightarrow I_{\text{crisp}}$ is called *a crisp classifier*. Let $V = \{\mathbf{v}_1, \cdots, \mathbf{v}_{n_p}\}$, $\mathbf{v}_i \in \Re^d$ be a set of point-prototypes. Our GNPC is based on the following principles.

Manuscript received March 3, 1998; revised September 29, 1999. This work was supported in part by the NRC COBASE program and ONR under Grant N 00014-96-1-0642.

L. I. Kuncheva is with the School of Mathematics, University of Wales, Bangor, Bangor, Gwynedd LL57 1UT, U.K.

J. C. Bezdek is with the Department of Computer Science, University of West Florida, Pensacola, FL 32514 USA.

Publisher Item Identifier S 1045-9227(99)05981-0.

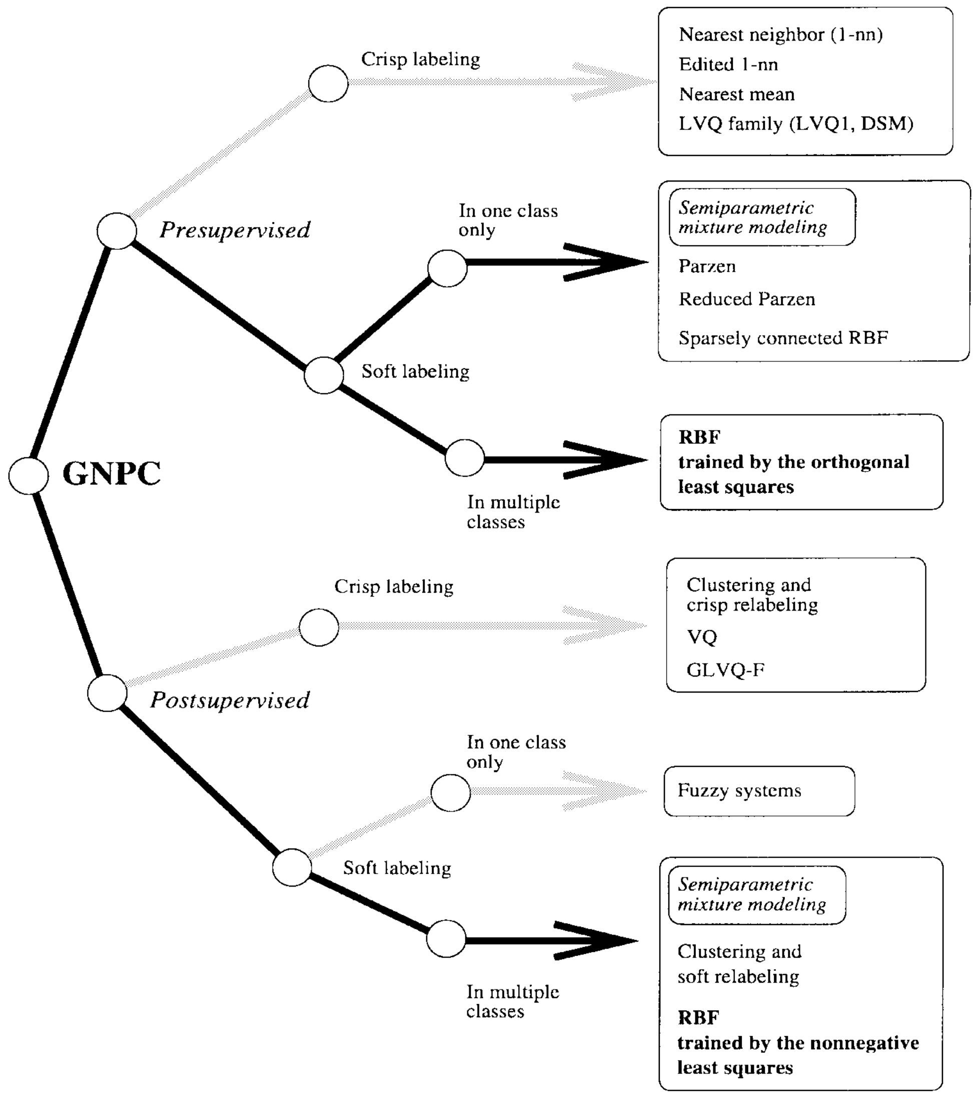

Fig. 1. Families of generalized nearest prototype classifiers.

- Every prototype $\mathbf{v}_j$ may "vote" for all c classes, and the strength of $\mathbf{v}_j$'s vote for class i, $1 \le i \le c$ is defined by a constant $l_{ij} \in [0,1]$.
- The closer (more similar) $\mathbf{x}$ is to the prototype $\mathbf{v}_j$, the higher the "relevance" of the $\mathbf{v}_j$-th vote is to the label for $\mathbf{x}$.

Assume that each prototype $\mathbf{v}_j$ is labeled by a column vector $l(\mathbf{v}_i) \in [0,1]^c - \{\mathbf{0}\}$, thereby constituting a $c \times n_p$ label matrix $\mathbf{L}_V = [l(\mathbf{v}_1), \cdots, l(\mathbf{v}_{n_p})]$. In our model the number of prototypes n_p may be less than, equal to, or greater than the number of classes c; and the label vector $l(\mathbf{v}_i)$ may be crisp or soft (fuzzy, probabilistic, possibilistic).

Definition 1: A norm-induced similarity function $s(\Delta(\mathbf{x},\mathbf{v});\theta)$, where θ is a set of parameters for s, is any monotonically decreasing function $s : \Re^+ \to [0,1]$ of any norm metric Δ on $\Re^d$. For example, if Δ_E denotes the Euclidean norm metric, s could be

$$s(\Delta_E(\mathbf{x},\mathbf{v});\eta) = \eta \exp\left(-\frac{1}{2}\Delta_E(\mathbf{x},\mathbf{v})^2\right), \qquad \eta \le 1. \quad (1)$$

Let $\mathcal{A}$ be an aggregation function defined as generalized matrix multiplication using an $\mathcal{S}$ operator instead of summation and a $\mathcal{T}$ operator instead of multiplication. Let $Q = [q_{ij}]$ and $R = [r_{ij}]$ be two matrices of size $n \times m$ and $m \times k$, respectively. The matrix $\mathcal{A}(Q,R) = [a_{ij}]_{n\times k}$, $i = 1,\cdots,n$; $j = 1,\cdots,k$ is defined as

$$a_{ij} = \underset{t=1}{\overset{m}{\mathcal{S}}}(\mathcal{T}(q_{it}, r_{tj})). \quad (2)$$

Definition 2: The GNPC is implemented with the 5-tuple $(V, \mathbf{L}_V, s, \mathcal{T}, \mathcal{S})$ where

- $V = \{\mathbf{v}_1, \cdots, \mathbf{v}_{n_p}\}$, $\mathbf{v}_i \in \Re^d$ is the set of prototypes;
- $\mathbf{L}_V = [l(\mathbf{v}_1), \cdots, l(\mathbf{v}_{n_p})]$, $l(\mathbf{v}_i) \in [0,1]^c - \{\mathbf{0}\}$, $i = 1, \cdots, n_p$ is the label matrix for the prototypes in c classes;
- $s(\Delta_E(\mathbf{x},\mathbf{v});\theta)$ is a similarity function, where θ is a set of parameters;
- $\mathcal{T}$ is any t-norm and $\mathcal{S}$ is an aggregation operator [24].

For an unlabeled vector $\mathbf{x} \in \Re^d$, the GNPC calculates the vector of similarities $\mathbf{s} = [s_1, \cdots, s_{n_p}]^T$ between $\mathbf{x}$ and $\{\mathbf{v}_i\}$

which is then used to compute the label vector $\mu(\mathbf{x}) = \mathcal{A}(\mathbf{L}_V, \mathbf{s}) = [\mu_1(\mathbf{x}), \cdots, \mu_c(\mathbf{x})]^T$. As a special case the *crisp* GNPC assigns the crisp class label $\mathbf{e}_l \in I_{\text{crisp}}$ to $\mathbf{x}$ if

$$\mu_l(\mathbf{x}) = \max_{i=1,\cdots,c} \{\mu_i(\mathbf{x})\}. \tag{3}$$

Ties are broken randomly.

Thus any unlabeled $\mathbf{x}$ is labeled on the basis of its proximity (or similarity) to the prototypes V combined with their class label information. Note that the GNPC at (3) assigns $\mathbf{x}$ a *crisp* class label. The GNPC is completely specified by $\{V, \mathbf{L}_V, s, \mathcal{T}$ and $\mathcal{S}\}$. We can try different combinations of these GNPC choices and select the most successful GNPC design by optimizing its classification accuracy.

In the next section we derive two models of the GNPC based on postsupervised and presupervised mixture modeling [18], [21] where the components are kernel-type functions (e.g., Gaussians). Each component corresponds to a prototype whose *probability density function* (PDF) gives the similarity of $\mathbf{x}$ to the prototype $\mathbf{v}_i$. The mixing coefficients are used to compute the "soft" class labels of the prototypes.

III. Postsupervised GNPC Design

Let $\mathbf{x} \in \Re^d$ be a random vector coming from one of c classes. Let $P(i)$ denote the prior probability; $p(\mathbf{x} \mid i)$ the class-conditional PDF; and $P(i \mid \mathbf{x})$ the posterior probability for class i, $i = 1, \cdots, c$. Let $p(\mathbf{x})$ be the unconditional PDF

$$p(\mathbf{x}) = \sum_{i=1}^{c} P(i) p(\mathbf{x} \mid i). \tag{4}$$

The objective is to build a GNPC that produces a Bayes-optimal classification decision (assuming a zero-one loss matrix), i.e., the class label assigned to $\mathbf{x}$ is $\mathbf{e}_l \in I_{\text{crisp}}$ when

$$P(l \mid \mathbf{x}) = \max_{1 \le i \le c} P(i \mid \mathbf{x}). \tag{5}$$

Hence, a sufficient condition for optimality of the GNPC is that, if sorted by magnitude, $\{\mu_i(\mathbf{x})\}_{i=1}^c$ have the same order as the sorted $\{P(i \mid \mathbf{x})\}_{i=1}^c \; \forall \mathbf{x} \in \Re^d$.

We consider mixture modeling for the GNPC design. Mixture modeling is used to identify the priors, the class-conditional PDF's and the unconditional PDF, which can be used with Bayes rule to calculate $\{P(i \mid \boldsymbol{x})\}$. Kernel mixture models are nonparametric. Typically, all training data points are used, each one generating a kernel (e.g., Parzen's window classifier). Other methods (e.g., neural networks) try to reduce the number of kernels without much degradation in classification performance. Shrinking the number of kernels shifts the mixture paradigm from nonparametric toward "semiparametric" [23].

We assume that $p(\mathbf{x})$ can be approximated by another mixture of new PDF's $\{\hat{p}(\mathbf{x} \mid C_j)\}$ using $\{P(C_j)\}$ as priors

$$p(\mathbf{x}) \approx \hat{p}(\mathbf{x}) = \sum_{j=1}^{n_p} P(C_j) \hat{p}(\mathbf{x} \mid C_j), \quad \textbf{(A1)} \tag{6}$$

where $\{C_1, \cdots, C_{n_p}\}$ will be referred to as "*hidden categories*" [23]. We will try to use as few mixture components (prototypes) as possible, so the approximation at (6) will differ from the true value $p(\mathbf{x})$ at (4). We call (6) the *decomposition assumption*, and shall refer to it as (**A1**). We require that the mixture components of (6) have kernel-type PDF's. That is, besides the condition

$$\int_{\Re^d} \hat{p}(\mathbf{x} \mid C_j)\, d\mathbf{x} = 1, \quad 1 \le j \le n_p \tag{7}$$

each $\hat{p}(\mathbf{x} \mid C_j)$ is based on a kernel-type function [10], $\hat{p}(\mathbf{x} \mid C_j) = 1/h_j^d \; K(\Delta(\mathbf{x}, \mathbf{v}_j)/h_j)$, where $h_j \in \Re^+$ is a smoothing parameter and $\mathbf{v}_j \in \Re^d$ is a prototype. A typical choice is the Gaussian kernel

$$\hat{p}(\mathbf{x} \mid C_j) = \frac{1}{(2\pi)^{d/2} h_j^d \sqrt{|S|}} \times \exp\left(-\frac{1}{2h_j^2} \sqrt{(\mathbf{x} - \mathbf{v}_j)^T S^{-1} (\mathbf{x} - \mathbf{v}_j)}\right) \tag{8}$$

where S is a covariance matrix. Often S is chosen to be the same for all prototypes, or at least common for all the prototypes of each class. Equation (6) provides Parzen's estimate of the PDF at (4) [8], [10] if each kernel is centered at a data point and if the number of data points n approaches infinity.

In the sequel we assume that we have a satisfactory algorithm to estimate all the parameters of the mixture (6), the *a priori* probabilities for the classes $\{P(i)\}$, and the conditional probabilities $\{\Pr(i \mid C_j)\}$. We do not restrict the class-conditional PDF's $p(\mathbf{x} \mid i)$. The classes and the hidden categories are related through

$$P(i) = \sum_{j=1}^{n_p} P(C_j) \Pr(i \mid C_j), \qquad i = 1, \cdots, c \tag{9}$$

where $\Pr(i \mid C_j)$ is the probability of class i if hidden category C_j occurs. By conditioning (9) on $\mathbf{x}$ we obtain the posterior probability for class i as

$$P(i \mid \mathbf{x}) = \sum_{j=1}^{n_p} P(C_j \mid \mathbf{x}) \Pr(i \mid C_j, \mathbf{x}). \tag{10}$$

To build an RBF network Traven [23] assumes that

$$\Pr(i \mid C_j, \mathbf{x}) \approx P(i \mid C_j). \quad \textbf{(A2)} \tag{11}$$

We call (11) the *homogeneity assumption* and shall refer to it as (**A2**). Substituting (11) into (10)

$$P(i \mid \mathbf{x}) \approx \sum_{j=1}^{n_p} P(C_j \mid \mathbf{x}) \Pr(i \mid C_j). \tag{12}$$

Developing (12) further, we have with Bayes rule

$$P(i \mid \mathbf{x}) \approx \sum_{j=1}^{n_p} \frac{\hat{p}(\mathbf{x} \mid C_j) P(C_j)}{\hat{p}(\mathbf{x})} \Pr(i \mid C_j). \tag{13}$$

Let

$$A = \max_{j=1,\cdots,n_p} \left\{ \max_{\mathbf{x} \in \Re^d} \{p(\mathbf{x} \mid C_j)\} \right\}. \tag{14}$$

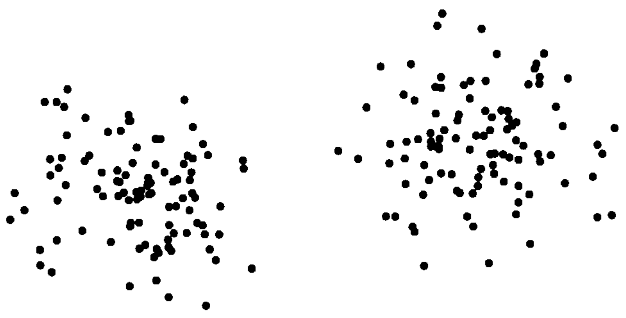
Fig. 2. Scatter plot of X disregarding the class labels.

Dropping the denominator of (13), which is the same for all i, and dividing by A, the Bayes-optimal classifier under assumptions (**A1**) and (**A2**) can be represented by the following set of discriminant functions:

$$\mathcal{G}_i(\mathbf{x}) = \frac{1}{n_p}\sum_{j=1}^{n_p}\frac{\hat{p}(\mathbf{x}\mid C_j)}{A}\Pr(i,C_j), \qquad i=1,\cdots,c \tag{15}$$

where $\Pr(i,C_j)$ is the probability of simultaneous occurrence of class i and hidden category C_j. Assuming kernel-type PDF's, $\hat{p}(\mathbf{x}\mid C_j)/A$ can be represented as $s(\|\mathbf{x}-\mathbf{v}_j\|;\theta_j)$, where s is a similarity function (Definition 1) and $\|\cdot\|$ is any norm on $\Re^d$. Equation (15) can then be rewritten as

$$\mu_i(\mathbf{x}) \equiv \mathcal{G}_i(\mathbf{x}) = \frac{1}{n_p}\sum_{j=1}^{n_p} s(\|\mathbf{x}-\mathbf{v}_j\|;\theta_j)l_{ij}, \qquad i=1,\cdots,c \tag{16}$$

where l_{ij} stands for the probability $\Pr(i,C_j)$. Therefore, the *postsupervised* GNPC that is a Bayes-optimal classifier under assumptions (**A1**) and (**A2**) is

$$\begin{aligned}
&\mathrm{GNPC}_{\mathrm{post}} = \\
&\qquad V=\{\mathbf{v}_j\};\quad j=1,\cdots,n_p\\
&\qquad \mathbf{L}_V=\left[l(\mathbf{v}_1),\cdots,l(\mathbf{v}_{n_p})\right];\quad l_{ij}=\Pr(i,C_j)\\
&\qquad s(\|\mathbf{x}-\mathbf{v}_j\|;\theta_j)=\hat{p}(\mathbf{x}\mid C_j)/A\\
&\qquad \mathcal{T}=\textit{product}\\
&\qquad \mathcal{S}=\textit{average}.
\end{aligned}$$

The following examples illustrate the situation with respect to Bayes optimality of $\mathrm{GNPC}_{\mathrm{post}}$. Plotted in Fig. 2 is data set X consisting of 200 two-dimensional (2-D) points. The vectors come from two classes with labels $\mathbf{e}_1=[1,0]^T;\mathbf{e}_2=[0,1]^T$. The class labels of the points are *not shown* in Fig. 2. X was generated from (4) using a mixture of two Gaussians with identity covariance matrices, viz.,

$$\begin{aligned}
\hat{p}(\mathbf{x}) &= 0.5\hat{p}(\mathbf{x}\mid C_1)+0.5\hat{p}(\mathbf{x}\mid C_2)\\
\hat{p}(\mathbf{x}\mid C_j) &= \frac{1}{2\pi}\exp^{(-\frac{1}{2}\|\mathbf{x}-\mathbf{v}_j\|_E^2)}, \quad j=1,2
\end{aligned} \tag{17}$$

where $\mathbf{v}_1=[1,2]^T$ and $\mathbf{v}_2=[7,3]^T$ are the prototypes of the two mixture components (hidden categories C_1 and C_2).

Below we detail four cases **A**, **B**, **C**, and **D**. The illustrations (Figs. 3, 4, 5, 7, and 9) use *the same* data set X with different labelings corresponding to "ideal" sampling from the respective cases.

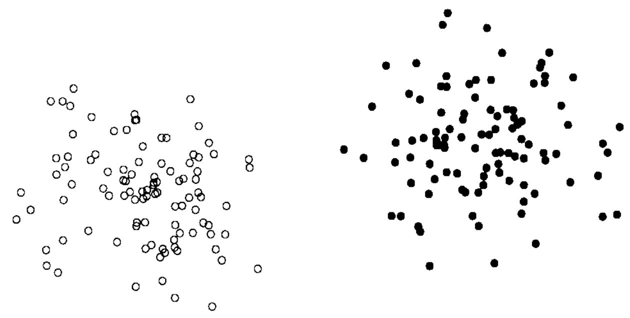
Fig. 3. **Case A:** The most desirable class labeling of $X : p(\mathbf{x}\mid i)=\hat{p}(\mathbf{x}\mid C_i),\ i=1,2$.

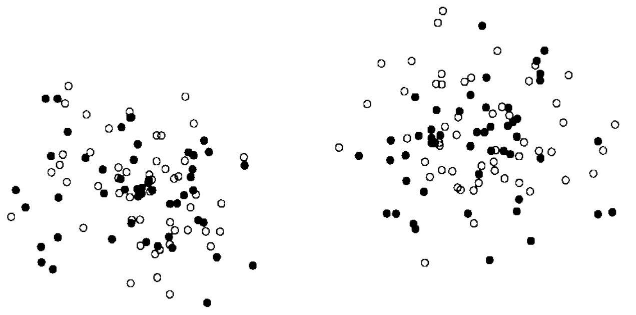
Fig. 4. **Case B:** The worst class labeling of $X : p(\mathbf{x}\mid 1)=p(\mathbf{x}\mid 2)$.

Case A: Ideally, the classes will correspond one-to-one to the hidden categories (e.g., Fig. 3, where class 1, denoted by circles has the same parameters as C_1, and class 2, denoted by filled circles, has the same parameters as C_2). In this case

$$l_{ij}=\begin{cases}\Pr(i,C_j)=P(C_j)=P(i), & \text{if } i=j\\ 0, & \text{otherwise}\end{cases} \quad i,j=1,2. \tag{18}$$

Since $s(\Delta(\mathbf{x},\mathbf{v}_i);\theta_i)$ is proportional to $p(\mathbf{x}\mid i)$, it is easy to show that the classification decision of $\mathrm{GNPC}_{\mathrm{post}}$ is Bayes-optimal.

Case B: The worst possible case of class labeling of X is shown in Fig. 4. The classes are equiprobable and the class-conditional PDF's are identical, i.e., these labels correspond to the case where

$$p(\mathbf{x}\mid 1)=p(\mathbf{x}\mid 2)=0.5\hat{p}(\mathbf{x}\mid C_1)+0.5\hat{p}(\mathbf{x}\mid C_2). \tag{19}$$

In this case the hidden categories C_1 and C_2 cannot be associated with a specific class label. Therefore, it is pointless to attempt to identify C_1 and C_2 because the probability of each class to occur together in either category is 0.5. The error of the $\mathrm{GNPC}_{\mathrm{post}}$ will be 0.5, and, again, this is the Bayes-optimal error rate. The situation in Fig. 4 can occur when some other feature has been used to label the data and this information is not represented in the current feature space $\Re^2$.

Case C: Fig. 5 shows the case where each *class* has a bimodal PDF whose modes are situated at the common prototypes $\mathbf{v}_1$ and $\mathbf{v}_2$. Class-conditional PDF's that will generate

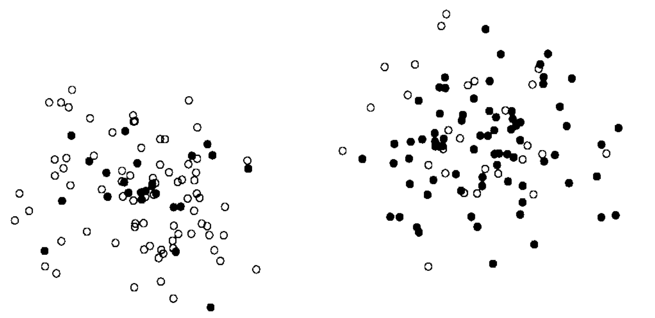
Fig. 5. **Case C:** Decomposable bimodal class-conditional PDF's.

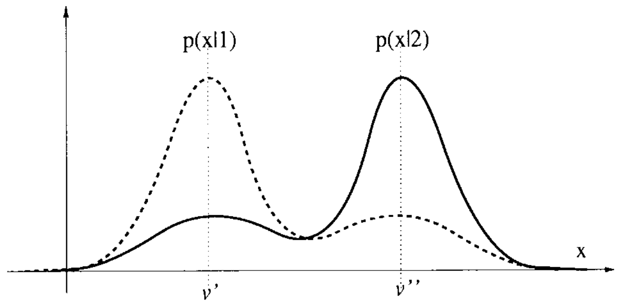

Fig. 6. **Case C:** Bimodal class-conditional PDF's for $x \in \Re$.

labels such as those shown in Fig. 5 are

$$p(\mathbf{x} \mid 1) = 0.25\hat{p}(\mathbf{x} \mid C_1) + 0.75\hat{p}(\mathbf{x} \mid C_2), \tag{20}$$

and

$$p(\mathbf{x} \mid 2) = 0.75\hat{p}(\mathbf{x} \mid C_1) + 0.25\hat{p}(\mathbf{x} \mid C_2). \tag{21}$$

An example of bimodal class-conditional PDF's for $x \in \Re$ is shown in Fig. 6. The modes v' and v'' are the same for both PDF's but the functions are mirror-wise symmetric. If data are drawn from a mixture of the two PDF's shown in Fig. 6 with mixture coefficients 0.5, the result will be two distinct Gaussian clusters in $\Re$, each one containing objects from both classes as illustrated in Fig. 5. As long as the class-conditional PDF's are decomposable on $\hat{p}(\mathbf{x} \mid C_1)$ and $\hat{p}(\mathbf{x} \mid C_2)$, the classification decision of the GNPC$_{\text{post}}$ $\underline{\text{is Bayes-optimal}}$.

Case D: In all cases considered so $\overline{\text{far assumptions}}$ (**A1**) and (**A2**) are satisfied, i.e., $p(\mathbf{x})$ is exactly decomposable on C_1 and C_2, and the probability of class i conditioned by C_j does not depend on $\mathbf{x}$. For example, for the case in Fig. 4, for every $\mathbf{x} \in \Re^2$ that is known to have come from C_1, the probability that $\mathbf{x}$ is from class 1 is 0.25, no matter where $\mathbf{x}$ is located. Sometimes, however, the dependence of $\Pr(i \mid C_j)$ on $\mathbf{x}$ is not negligible. This is illustrated in Fig. 7. The "clusters" corresponding to C_1 and C_2 are *not homogeneous*, i.e., they possess hidden substructure which will be ignored if the GNPC is built as suggested above. Here the homogeneity assumption (**A2**) is not satisfied.

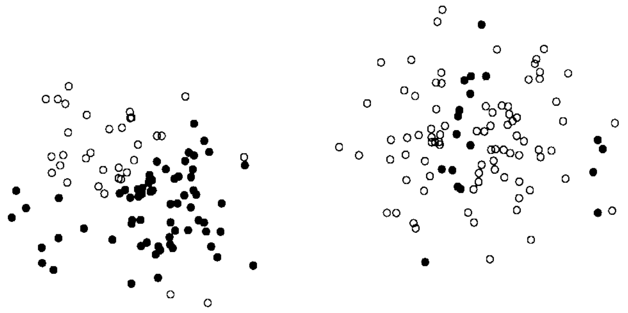
Fig. 7. **Case D:** Nondecomposable class-conditional PDF's.

The class labels in Fig. 7 are artificially assigned to X (Fig. 2) according to the rule: $\mathbf{x}$ is assigned to

Class 1, if $[x_1 \sin(x_1) - 1.6] \le x_2 \le [x_1 \sin(x_1) + 1.6]$;
Class 2, otherwise.

For this labeling the class-conditional PDF's are shown in (22) and (23) at the bottom of the page, where

$$a_1 = \int_{-\infty}^{\infty} \int_{\xi \sin(\xi)-1.6}^{\xi \sin(\xi)+1.6} \hat{p}([\xi, \tau]^T \mid C_1)\, d\tau\, d\xi, \quad a_2 = 1 - a_1 \tag{24}$$

and

$$b_1 = \int_{-\infty}^{\infty} \int_{\xi \sin(\xi)-1.6}^{\xi \sin(\xi)+1.6} \hat{p}([\xi, \tau]^T \mid C_2)\, d\tau\, d\xi, \quad b_2 = 1 - b_1 \tag{25}$$

Fig. 8 shows separately the two classes and the class boundaries for 2000 points generated from the same distribution as X. Note that densities (22) and (23) are nonoverlapping, which means that the Bayes error rate for this case is zero. Neglecting nonhomogeneity of the two hidden components by assigning $l(\mathbf{v}_1) = [a_1, a_2]^T$ and $l(\mathbf{v}_2) = [b_1, b_2]^T$ as in the GNPC description, the error rate will be nonzero.

Fig. 9 shows the difference between the homogeneous and nonhomogeneous groups. The two scatterplots are the same as in Figs. 4 (upper) and 7 (lower). The small "windows" in the upper plot show that the ratio of the number of points from class 1 to class 2 is approximately the same ($\approx$0.5), no matter where the window is placed. In the lower plot the ratio depends on the window location, which suggests that the GNPC$_{\text{post}}$ $\underline{\text{is not Bayes-optimal}}$ because assumption (**A2**) is violated.

Nonoptimality can also arise if the true density $p(\mathbf{x})$ is not decomposable on the hidden categories $\{C_1, \cdots, C_{n_p}\}$ [violation of Assumption (**A1**)]. This can be overcome by

$$p(\mathbf{x} \mid 1) = \begin{cases} \frac{\hat{p}(\mathbf{x}|C_1)}{a_1} + \frac{\hat{p}(\mathbf{x}|C_2)}{b_1}, & \text{if } [x_1 \sin(x_1) - 1.6] \le x_2 \le [x_1 \sin(x_1) + 1.6] \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

$$p(\mathbf{x} \mid 2) = \begin{cases} 0, & \text{if } [x_1 \sin(x_1) - 1.6] \le x_2 \le [x_1 \sin(x_1) + 1.6] \\ \frac{\hat{p}(\mathbf{x}|C_1)}{a_2} + \frac{\hat{p}(\mathbf{x}|C_2)}{b_2}, & \text{otherwise.} \end{cases} \tag{23}$$

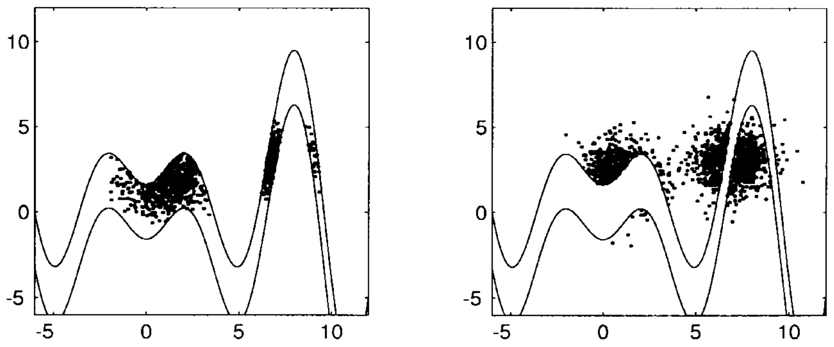

Fig. 8. The classes and class boundaries for samples from (22) and (23).

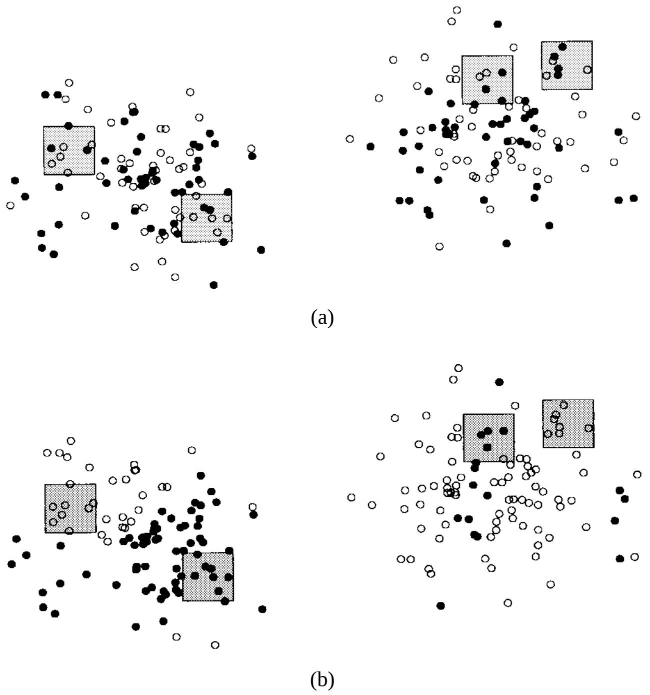

Fig. 9. (a) Homogeneous and (b) nonhomogeneous groups.

increasing n_p. In all the above examples we assumed that $p(\mathbf{x})$ *is* decomposable on $\hat{p}(\mathbf{x} \mid C_1)$ and $\hat{p}(\mathbf{x} \mid C_2)$. Therefore, in the postsupervised model, the sources of "nonoptimality" of the GNPC are potentially two: the inexact representation of $p(\mathbf{x})$ [Assumption (**A1**)], and neglecting the dependency of $\Pr(i \mid C_j, \mathbf{x})$ on $\mathbf{x}$ [Assumption (**A2**)].

IV. Presupervised GNPC Design

Instead of (6) we approximate the class-conditional PDF's on separate sets of prototypes. This approach has been used for constructing a sparsely connected RBF network [5] but in general, the more popular design is the postsupervised one. Let the set of prototypes be arranged as follows:

$$\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{j_1}, \mathbf{v}_{j_1+1}, \cdots, \mathbf{v}_{j_2}, \cdots, \mathbf{v}_{j_{c-1}+1}, \cdots, \mathbf{v}_{j_c}\} \tag{26}$$

where the subset $V[i] = \{\mathbf{v}_{j_{i-1}+1}, \cdots, \mathbf{v}_{j_i}\} \in V$ is used to approximate $p(\mathbf{x} \mid i)$, the ith class-conditional density, $i = 1, \cdots, c$; $j_0 = 0$; $j_c = n_p$. Assumption (**A1**) now takes the form that the conditional PDF in (4) can be approximated by

$$p(\mathbf{x} \mid i) \approx \sum_{k=j_{(i-1)}+1}^{j_i} P(C_k)\hat{p}(\mathbf{x} \mid C_k). \tag{27}$$

Since the prototypes are uniquely connected with the classes, assumption (**A2**) is not needed here. Therefore, in presupervised mixture modeling for GNPC design the only source of nonoptimality is imprecision in approximating the class-conditional densities by decomposition on finite sets of kernel-based components.

Recalling that A in (14) is the maximal PDF value and using the discriminant functions

$$\begin{aligned}\mathcal{G}_i(\mathbf{x}) &= \frac{1}{n_p A} P(i)p(\mathbf{x} \mid i)\\ &= \frac{1}{n_p} \sum_{k=j_{(i-1)}+1}^{j_i} P(i)P(C_k)\frac{\hat{p}(\mathbf{x} \mid C_k)}{A}\\ &\qquad\qquad i = 1, \cdots, c,\end{aligned} \tag{28}$$

one possible Bayes-optimal presupervised GNPC design is

$$\begin{aligned}&\mathrm{GNPC}_{\mathrm{pre}} =\\ &\quad V = \{\mathbf{v}_j\}; \quad j = 1, \cdots, n_p\\ &\quad \mathbf{L}_V = [l(\mathbf{v}_1), \cdots, l(\mathbf{v}_{n_p})]\\ &\quad l_{ik} = \begin{cases} P(i)P(C_k), & \text{if } j_{(i-1)} + 1 \le k \le j_i\\ 0, & \text{otherwise.}\end{cases}\\ &\quad s(\Delta(\mathbf{x}, \mathbf{v}_j); \theta_j) = \hat{p}(\mathbf{x} \mid C_j)/A\\ &\quad \mathcal{T} = \textit{product}\\ &\quad \mathcal{S} = \textit{average}.\end{aligned} \tag{29}$$

In the presupervised design we have only *one* type (the decomposition) of assumption, but it must hold for *all* class-conditional PDF's. Thus, we need c assumptions to hold. Therefore, the number of prototypes required can be greater than that for GNPC$_{\mathrm{post}}$. With more prototypes, GNPC$_{\mathrm{post}}$ can also overcome the "nonhomogeneity" problem because the C_js can be restricted to small parts of the feature space where the class-conditional PDF's might be practically homogeneous. By increasing the number of prototypes the decomposition assumption will be less likely to be violated. Asymptotically

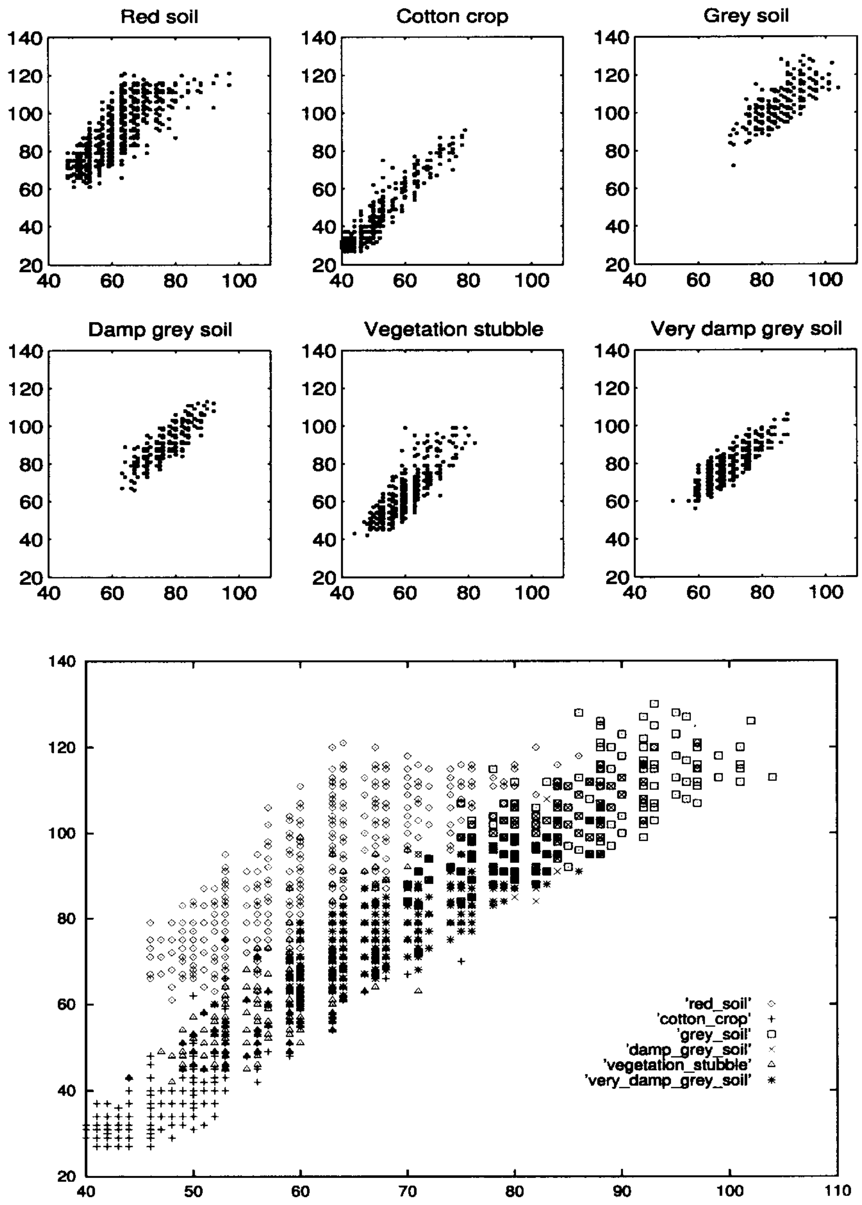

Fig. 10. Scatterplot of the six classes on features # 17 (horizontal) and 18 (vertical).

this leads to approximation of the true class-conditional (and the unconditional) PDF's.

V. Experimental Results

To illustrate the two designs we used the following data sets.

- **The 2-D data (case D)**. The training set consists of 200 points and the test set of 2000 points in $\Re^2$, all drawn from the distribution (22) and (23).
- **The** "satimage" **data from ELENA database**. The data can be obtained by anonymous ftp at ftp.dice.ucl.ac.be, directory pub/neural-nets/ELENA/databases. The `satimage` data is generated from Landsat Multispectral Scanner image data. It consists of $n = 6435$ feature vectors with 36 attributes (four spectral bands $\times$ nine pixel intensities per 3×3 window). The data are classified into six physical classes, and are presented in random order in the database. Here we used only features 17–20, as recommended by the designers of the database. The first 200 vectors were used for training and the remaining 6235 for testing. Fig. 10 shows scatterplots of the six satimage classes on features 17 and 18.

Our objective is to experimentally compare particular presupervised and postsupervised GNPC designs. At the beginning of Section III we assumed that we had a satisfactory algorithm to approximate the mixture densities. Practically, classifiers based on explicit mixture modeling are seldom used. Thus, the imprecision of the approximation algorithm may spoil the comparison. Instead, we chose one representative from each group of GNPC's (in boldface in Fig. 1).

- **OLS RBF (Presupervised GNPC)** [4]. This RBF network is considered because the prototypes to be retained are selected from the training data on the basis of a score

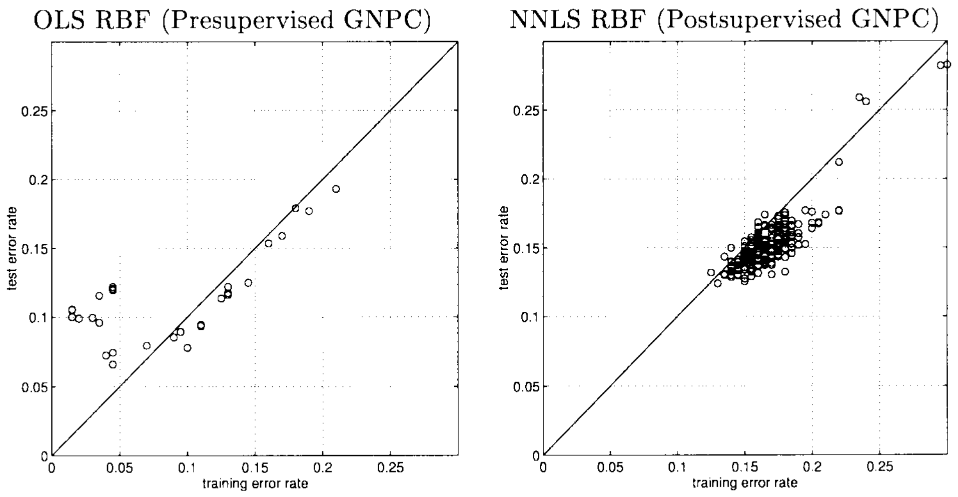

Fig. 11. Test versus training error rates for the presupervised and postsupervised GNPC designs with the 2-D data.

calculated using data labels. A parameter that must be specified in advance is *sc*, the "scaling factor" in the denominator of the power of the kernel exponent. It corresponds to the smoothing parameter h_j in (8) and has the same value for all prototypes. We tried five different values of *sc* with each data set, viz., *sc* $= 0.5, 1.0, 1.5, 2.0$ and 2.5 for the 2-D data and *sc* $= 4, 6, 8, 10$ and 12 for the satimage data.

- **NNLS RBF (Postsupervised GNPC)** [19]. This design is postsupervised because the prototypes are found by clustering the whole data set, disregarding the class labels. To obtain the similarity s we use

$$s(\Delta_E(\mathbf{x}, \mathbf{v})) = \exp\left(-\frac{1}{w}\Delta_E(\mathbf{x}, \mathbf{v})^2\right) \qquad (30)$$

where w is the "nearest neighbor heuristic," as suggested in [19]. The label matrix $\mathbf{L}_V$ is then found by a nonnegative least squares procedure (see the Matlab reference books, MathWorks, Inc., and the reference recommended there [16]). NNLS is a version of the least squares method in which the resultant vector (a row of $\mathbf{L}_V$) is nonnegative. Although less accurate, NNLS was adopted because we wish to keep the interpretation of $\mathbf{L}_V$ as soft class labels for the prototypes (negative labels do not make sense). NNLS yields the best possible match for $\mathbf{L}_V$ under the nonnegativity constraint.

For the OLS RBF we used the Neural-Network Toolbox for Matlab and for the NNLC RBF, the k-means clustering code from the pattern recognition package PRTOOLS for Matlab [9]. With the 2-D data n_p was varied from two to 30 (29 values) and with the Satimage data, from six to 30 (26 values). Since the OLS RBF is a deterministic algorithm, for a specific *sc* we ran it once for each n_p, i.e., 29 (or 26) runs. The NNL RBF depends on the initialization of the hard k-means, and therefore it was run ten times starting from different initializations for each number of prototypes n_p (clusters), i.e., total 290 (260) runs.

For comparison we also used the following classifiers from [9]:

parametric:	Linear discriminant classifier (LDC)
	Quadratic discriminant classifier (QDC)
	Nearest mean classifier (NMC)
semiparametric:	Logistic classifier (LOGC)
nonparametric:	Nearest neighbor (1-nn)
	Parzen windows classifier (Parzen).

We carried out one hold-out experiment with each data set because we observed that the training and the test error rates correlate well. This is shown in Figs. 11 and 12 which plot the test versus training error rates for the runs with one of the OLS RBF's (with *sc* $= 2.0$ for the 2-D data and *sc* $= 12$ with the satimage data) and with the NNLS RBF. A well-designed classifier will have the same training and test error rates and will be a dot on the diagonal. In the best case, these error rates will be zero or close to the left bottom corner. The figures show that the presupervised designs produce smaller error rates with both data sets and that for the smallest error rates they are likely to overtrain. This is indicated by the points on the two left-hand plots that are above the diagonal (i.e., the test error rate is higher than the training error rate), especially with the 2-D data. The postsupervised designs show better match, i.e., the test error rates are even a little bit lower than the training rates (right-hand plots).

Fig. 13 shows the test error rates with the two GNPC designs:

- **Presupervised (OLS RBF)**. We show the error rate on the test data using the *sc* value that produced the smallest resubstitution error on the training data: *sc* $= 2$ with the 2-D data, and *sc* $= 12$ with the satimage data.
- **Postsupervised (NNLS RBF)**. The plain lines in Fig. 13 show the averaged error rates of the runs with different initializations and the same number of prototypes.

The figure shows that the presupervised design provides lower error rates. Looking at the training-test plots (Fig. 12) with the

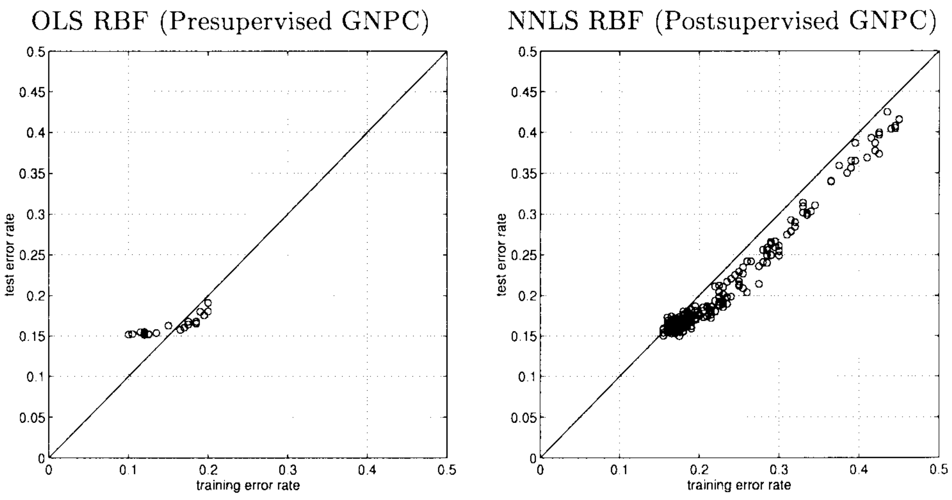

Fig. 12. Test versus training error rates for the presupervised and postsupervised GNPC designs with the `satimage` data.

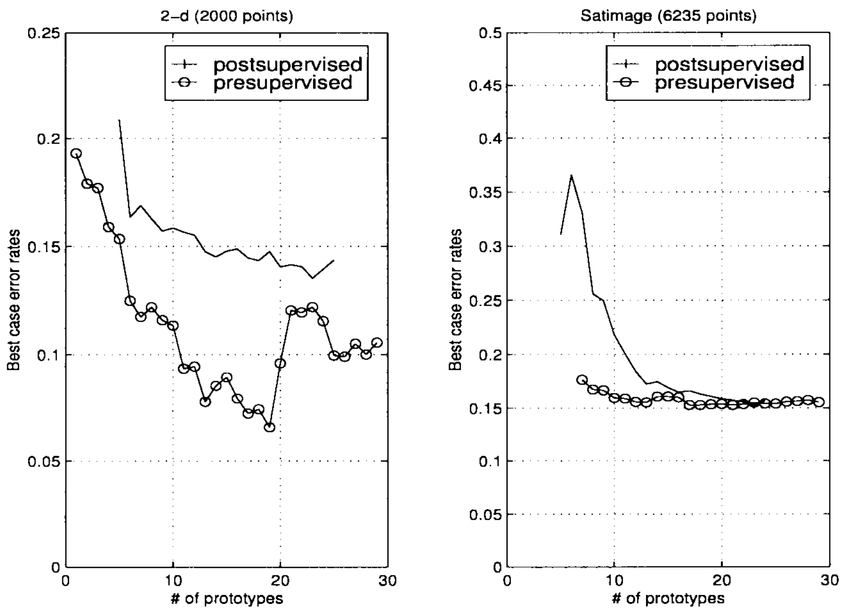

Fig. 13. Best case error rates for the postsupervised and presupervised GNPC's.

satimage data we see that although the presupervised GNPC's reach lower training error, the best-case test error is as high as that of the postsupervised GNPC's. This is observed for high n_p and is caused by overtraining of the presupervised GNPC's. The postsupervised design seems more robust since the training and test error rates correlate well and are of the same magnitude (Figs. 11 and 12). For small numbers of prototypes the supervised design is clearly better. The gap between the accuracies of the two designs is more clear with the 2-D data. Even with the high overtraining for small training errors (high n_p) displayed in Fig. 11 the presupervised design shows better *test* accuracy than the postsupervised design. This is probably due to the specific structure of the data (Case **D**) which makes the supervised design a more reasonable choice.

Tables II and III display the minimal error rates found by the two designs. Here "*minimal*" means that we take the smallest error on the training data and show the *corresponding* test error. The number of prototypes used is shown in parentheses.

TABLE I
ERROR RATES IN [%] ON THE TEST DATA SETS

	LDC	QDC	NMC	LOGC	1-nn	Parzen
2-d	24.00	19.15	28.25	23.45	6.85	10.25
Satimage	18.38	16.98	25.12	19.50	20.02	14.74

Where more than one classifier achieves the same lowest *training* error rate all test error rates are shown. We display the results for all five OLS RBF's as a function of *sc*. The numbers in boldface indicate what test error we would have if we chose the classifier with the globally minimal training error. The two GNPC designs compare favorably to the set of conventional classifiers (Table I). For the 2-D data, the *average* error rate for the six classifiers in Table I is 18.66%, whereas the average error in Table II (best case) for the 6 GNPC designs is 11.5%. For the satimage data the Table I average is 19.12% whereas the GNPC average from Table III

TABLE II
BEST CASE (MINIMAL) ERROR RATES OF GNPC IN [%] WITH THE 2-D DATA

	NNLS RBF	OLS RBF				
		$sc = 0.5$	$sc = 1.0$	$sc = 1.5$	$sc = 2.0$	$sc = 2.5$
Training	12.50	5.50	3.00	1.50	1.50	6.50
Test	**13.20** (22)	12.10 (28) 12.05 (29)	11.70 (23) 12.25 (24)	**9.10** (25)	**10.50** (27) **10.00** (28) **10.55** (29)	12.95 (29)

TABLE III
BEST CASE (MINIMAL) ERROR RATES OF GNPC IN [%] WITH THE SATIMAGE DATA

	NNLS RBF	OLS RBF				
		$sc = 4$	$sc = 6$	$sc = 8$	$sc = 10$	$sc = 12$
Training	15.50	24.50	10.50	10.50	10.00	10.00
Test	**15.91** (20) **15.87** (21) **15.37** (22) **15.00** (23)	27.07 (29)	18.36 (27)	15.80 (27) 16.18 (28)	**15.17** (29)	**15.72** (28)

is 17.85%—not as dramatic an improvement as for the 2-D data—but better than the Table I results.

Tables II and III show that the value of the scaling parameter can be crucial for good performance. On the other hand, initialization of crisp k-means clustering did not seem to have much effect on classification performance: scatterplots of the test versus training errors with both data sets for the postsupervised design grouped along the bisectrix of the **1**-st quadrant. This means that the presupervised design is more sensitive to initialization. But this design takes less time, and, since the training algorithm is deterministic, it is easier to check the possible choices and select *sc*.

VI. CONCLUSION

Which is the preferable design for a prototype classifier like the GNPC—the presupervised or the postsupervised scheme?

Considerations in Sections III and IV suggest the following. For Bayes optimality with a fixed number of prototypes the postsupervised design requires two types of assumptions (decomposition and homogeneity) while the presupervised design requires only one (decomposition). With a small number of prototypes both types of assumptions may not hold, thus leading to degraded performance. The rate of degradation depends on how badly the assumptions are violated. Increasing the number of prototypes strengthens the decomposition assumption, leading (asymptotically) to approximation of the true PDF's. Alleviation of the homogeneity assumption is less obvious. Therefore, it is not generally clear which of the two approaches should be preferred. If we have reasons to suspect that the classes have hidden substructures (e.g., compact groups of objects forming clusters of irregular shape within larger compact clusters of data, as in Case **D**) the presupervised design seems to be a better choice.

Our experiments show that the postsupervised GNPC design is robust (to initialization change) and accurate. It can be expected to work well when data sets do not contain peculiar class shapes and where natural clusters in data correspond to classes (or parts of classes). The majority of data sets are of this type. Although there is a tendency toward overtraining, the presupervised design seems a better choice for "difficult" data sets. Splitting the training data into training and validation sets can help prevent overtraining, and is always recommended as a good engineering practice. With respect to the experimental part, we agree that we cannot make firm conclusions and give recommendations based on experiments only. Here we report only what *our experiments* have shown. We are aware that there will always be data sets for which our conclusions will not hold, and we assume that the reader is aware of this too.

REFERENCES

[1] J. C. Bezdek, S. K. Chuah, and D. Leep, "Generalized k-nearest neighbor rules," *Fuzzy Sets Syst.*, vol. 18, pp. 237–256, 1985.
[2] J. C. Bezdek, T. R. Reichherzer, G. S. Lim, and Y. Attikiouzel, "Multiple prototype classifier design," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, pp. 67–79. Feb. 1998.
[3] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.
[4] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.
[5] R. L. Coultrip and R. H. Granger, "Sparse random networks with LTP learning rules approximate Bayes classifiers via Parzen's method," *Neural Networks*, vol. 7, pp. 463–476, 1994.
[6] B. V. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1990.
[7] C. Decaestecker, "NNP: A neural net classifier using prototypes," in *Proc. IEEE Int. Conf. Neural Networks*, San Francisco, CA, 1993, pp. 822–824.
[8] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[9] R. P. W. Duin, *PRTOOLS. A Matlab Toolbox for Pattern Recognition*, Delft Univ. Technol., Delft, The Netherlands: 1997.
[10] K. Fukunaga *Introduction to Statistical Pattern Recognition*. Orlando, FL: Academic, 1972.
[11] N. B. Karayiannis and P.-I. Pai, "Fuzzy algorithms for learning vector quantization," *IEEE Trans. Neural Networks*, vol. 7, pp. 1196–1211, 1996.
[12] N. B. Karayiannis, J. C. Bezdek, N. R. Pal, R. J. Hathaway, and P.-I. Pai, "Repairs to GLVQ: A new family of competitive learning schemes," *IEEE Trans. Neural Networks*, vol. 7, pp. 1062–1071, 1996.
[13] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbors algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. 15, pp. 580–585, 1985.
[14] T. Kohonen, "Improved versions of learning vector quantization," in *Proc. Int. Joint Conf. Neural networks*, San Diego, CA, 1990, pp. I-545–550.

[15] L. I. Kuncheva and J. C. Bezdek, “A fuzzy generalized nearest prototype classifier,” in *Proc. 7th IFSA World Congr.*, Prague, Czech Republic, vol. III, 1997, pp. 217–222.

[16] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Englewood Cliffs, NJ: Prentice-Hall, 1974, ch. 23.

[17] R. P. Lippmann, “Pattern classification using neural networks,” *IEEE Commun. Mag.*, pp. 47–64, 1989.

[18] G. J. McLachlan and K. E. Basford, *Mixture Models. Inference and Applications to Clustering*. New York: Marcel Dekker, 1988.

[19] J. Moody and C. J. Darken, “Fast learning in networks of locally tuned processing units,” *Neural Comput.*, vol. 1, pp. 281–294, 1989.

[20] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[21] D. M. Titterington, A. F. M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. Chichester, U.K.: Wiley, 1985.

[22] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles* Reading, MA: Addison-Wesley, 1974.

[23] H. G. C. Traven, “A neural network approach to statistical pattern classification by “semiparametric” estimation of probability density functions,” *IEEE Trans. Neural Networks*, vol. 2, pp. 366–377, 1991.

[24] R. R. Yager and D. P. Filev, *Essentials of Fuzzy Modeling and Control*. New York: Wiley, 1994.

[25] M.-S. Yang and C.-T. Chen, “On strong consistency of the fuzzy generalized nearest neighbor rule,” *Fuzzy Sets Syst.*, vol. 60, pp. 273–281, 1993.

[26] H.-C. Yau and M. T. Manry, “Iterative improvement of a nearest neighbor classifier,” *Neural Networks*, vol. 4, pp. 517–424, 1991.

Ludmila I. Kuncheva received the M.Sc. degree from the Technical University, Sofia, in 1982, and the Ph.D. degree from the Bulgarian Academy of Sciences in 1987.

In 1993 she was a Visiting Researcher at ELITE Laboratory, Aachen, Germany, and in 1995 to 1996 worked with the Neural Systems Group, EEE Department, Imperial College, London, under a Royal Society Research Fellowship. She was a Visiting Researcher in Pensacola, FL, sponsored by a COBASE research grant in 1996 to 1997. Her interests include pattern recognition, neural networks, fuzzy classifiers, prototype classifiers and multiple classifier systems.

James C. Bezdek (M’80–SM’90–F’92) received the BSCE degree from the University of Nevada, Reno, in 1969, and the Ph.D. degree in Applied Math from Cornell University, Ithaca, NY, in 1973.

He is currently a Professor in the Computer Science Department at the University of West Florida, Pensacola. His interests include optimization, pattern recognition, computer vision and image processing, computational neural networks, and medical applications.

Dr. Bezdek is the founding Editor of the *International Journal of Approximate Reasoning* and the IEEE TRANSACTIONS ON FUZZY SYSTEMS.