Designing Classifier Fusion Systems by Genetic Algorithms

Ludmila I. Kuncheva and Lakhmi C. Jain

***Abstract*—We suggest two simple ways to use a genetic algorithm (GA) to design a multiple-classifier system. The first GA version selects disjoint feature subsets to be used by the individual classifiers, whereas the second version selects (possibly) overlapping feature subsets, and also the types of the individual classifiers. The two GAs have been tested with four real data sets: Heart, Satimage, Letters, and Forensic glasses (tenfold cross validation, except for Satimage where we used only two splits). We used three-classifier systems and basic types of individual classifiers (the linear and quadratic discriminant classifiers and the logistic classifier). The multiple-classifier systems designed with the two GAs were compared against classifiers using: 1) all features; 2) the best feature subset found by the sequential backward selection (SBS) method; and 3) the best feature subset found by a GA (individual classifier!). We found that: 1) the multiple-classifier system derived through the GA, Version 2, yielded the smallest training error rate in all experiments; and 2) with Satimage and Forensic glasses data, it also produced the smallest test error rate. Generalizing on the basis of these experiments is not straightforward because the differences between the error rates in the comparison appeared to be too small. GA design can be made less prone to overtraining by including penalty terms in the fitness function accounting for the number of features used.**

***Index Terms*—Classifier fusion, feature selection, genetic algorithms, multiple classifiers.**

I. Introduction

LET $\Omega = \{\omega_1, \cdots, \omega_c\}$ be a set of c class labels, e.g, $\{$*gray-cell tissue, white-cell tissue, tumor mass, skull, background*$\}$. Let $\boldsymbol{x} = [x_1, \cdots, x_n]^T \in \Re^n$ be an n-dimensional column-vector[1] describing an object which, for the example above, can be a particular sample of magnetic resonance image (MRI) of a brain. Each component of $\boldsymbol{x}$ expresses the value of a *feature*, such as length, temperature, number of prickle per cm^2, concentration of cadmium, etc. A *classifier* is any mapping

$$D: \quad \Re^n \to [0,1]^c, \tag{1}$$

i.e., the output $D(\boldsymbol{x})$ is a c-dimensional vector whose ith component denotes the "support" for the hypothesis that $\boldsymbol{x}$ comes from class ω_i.

In multiple-classifier systems (Fig. 1) the outputs of a set of classifiers are combined to produce the final classification decision. The combination is sought as an *aggregation* of the outputs of L individual classifiers, i.e.,

$$D(\boldsymbol{x}) = \mathcal{F}(D_1(\boldsymbol{x}), \cdots, D_L(\boldsymbol{x})) \tag{2}$$

where $\mathcal{F}$ is an aggregation operator. If a single class label is needed for $\boldsymbol{x}$, the class with the highest support is chosen.

In the long run, the combined decision is supposed to be better (more accurate, more reliable) than the classification decision of the best individual classifier. This idea appears under a variety of names in the literature: classifier fusion [1], [2], classifier combination [3]–[7], mixture of experts [8]–[11], committees of neural networks [12], [13], consensus aggregation [14]–[16], voting pool of classifiers [17], classifier ensembles [13], [18], etc.

As the gray lines and ellipses in Fig. 1 show, multiclassifier systems differ by the following.

1) The *number L of individual classifiers* used.
2) The *type of the individual classifiers*. Some combination scheme use classifiers of the same types, e.g., neural networks [17], [9], linear classifiers [19], nearest neighbor classifiers [20], and other schemes use sets of different classifier models [6].
3) The *feature subsets* used by the individual classifiers [21]–[24] (denoted by the gray ellipses in Fig. 1).
4) The *aggregation* of the individual decisions [25]. Examples of these are majority vote [26]; naive Bayes [7]; behavior-knowledge space (BKS) [27]; simple aggregation connectives like average, product, minimum, maximum [4], [24]; fuzzy integral [28], [1], [2]; trained linear combinations [29]–[31]; neural network aggregation [32], [10]; Dempster–Shafer aggregation [33], [34], [7], etc.
5) The *training data sets* for the individual classifiers [20], [13].
6) The *type of training* of the two-level scheme (cf. [25]): a) training of the individual classifiers and applying aggregation that does not require further training (e.g., aggregation techniques like average, minimum, product, maximum, etc.); b) training of the individual classifiers followed by training the aggregation; c) simultaneous training of the whole scheme.

Usually the individual classifiers are chosen ad hoc on the basis of their accuracy (the higher the better). When the paradigm is trained as a whole (e.g., [19]), the parameters of the individual classifiers are varied along with the parameters of the aggregation scheme.

Manuscript received May 9, 1999; revised October 14, 1999.

L. I. Kuncheva is with the School of Informatics, University of Wales—Bangor, Bangor, Gwynedd LL57 1UT, Wales, U.K. (e-mail: l.i.kuncheva@bangor.ac.uk).

L. C. Jain is with the University of South Australia—Adelaide, Mawson Lakes, SA 5095, Australia (e-mail: lakhmi.jain@unisa.edu.au).

Publisher Item Identifier S 1089-778X(00)04466-0.

[1]Superscript T stands for "transposed."

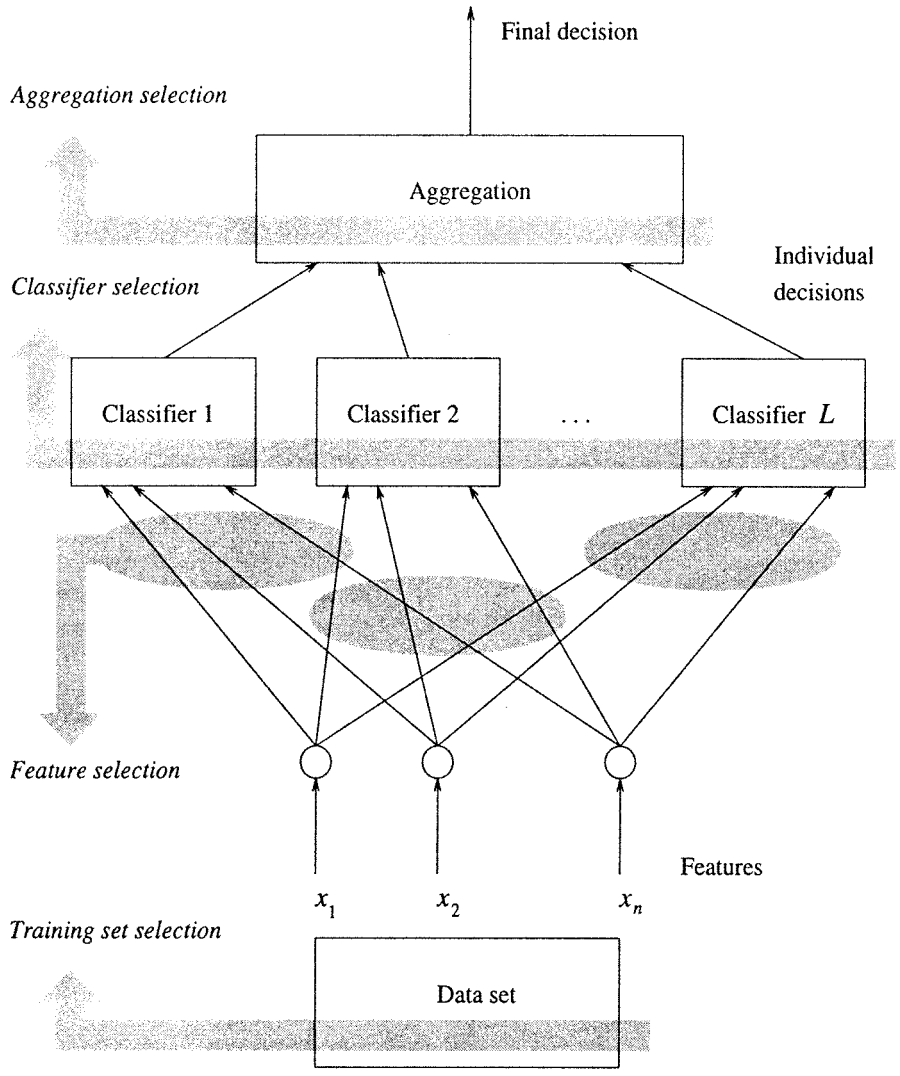

Fig. 1. Combination of multiple classifiers (classifier fusion). Gray lines and ellipses indicate possible ways of altering the classifier fusion system.

In this paper, we use a genetic algorithm (GA)[2] for selecting the feature subsets for the individual classifiers. In the first GA version, we select disjoint feature subsets for a fixed set of classifiers [see item 3) above]. The second version does not restrict the overlap between the feature subsets. Embedded in the chromosome in the second version is also a substring for selecting the type of each individual classifier from a set of prespecified models [items 2) and 3) from the above list].

Section II describes the classifier fusion system in detail. Section III provides our reasons for choosing a genetic algorithm for designing multiple-classifier systems, and presents the two GA versions. Section IV contains our experimental results, and Section V, our conclusions.

II. Combination of Classifiers

The details of the classifier fusion system used here are as follows.

1) *Number of Individual Classifiers:* In this study, we limited the number of individual classifiers to three, but the GAs that we propose can be used for any number of classifiers L. The chromosome length of GA Version 1 is n (number of features), and does not depend on L. In GA Version 2, the chromosome length is $L + n$, and since usually $n \gg L$, the length is not affected much by the increase of L. However, the number of integers that a position in a chromosome can assume grows exponentially (2^L), thereby determining an even higher rate of expansion of the search space. This shifts the (limited-) *integer-valued* GA algorithm used in this study toward real-valued GAs. For this large search space, extensive experiments have to be carried out in order to obtain consistent results, e.g., multiple runs of the GA with different initialization and parameter values. This could be a computationally demanding task. Also, small values of L are used by many authors because a larger number of classifiers is likely to contain highly correlated classifiers which might deteriorate the performance of the combination.
2) *Type of the Individual Classifiers:* In Version 1 of the GA, all individual classifiers are of the same (preselected) type. The choice of classifier type is among the *linear,*

[2]For an explanation of GAs the reader is referred to [35], [36].

quadratic, and *logistic* models. The individual classifiers differ only by the features they use. In Version 2, the classifier type is encoded into the GA. Each individual classifier can be picked from the three options above.

3) *Feature Selection:* In Version 1, the GA selects disjoint feature subsets for the three classifiers. In Version 2, the feature subsets for the individual classifiers are not restricted, i.e., each of the three classifiers can use any feature subset.
4) *Aggregation:* We use either *majority vote* or *decision templates* (DT's) [25], both detailed below.
5) *Training Data Sets:* The same training sets are used for all individual classifiers.
6) *Training:* Both GA versions train the classifier fusion system as a whole [subitem 6c) above].

Majority vote is a popular and easy-to-implement method. The individual classifiers "vote" with class labels for $\boldsymbol{x}$, and the class label with most votes is assigned to $\boldsymbol{x}$. In some studies, the class label is assigned if the majority of L classifiers, i.e., at least $\lfloor L/2 \rfloor + 1$ classifiers, vote for that class. In such cases, if there is no winner, the combined classifier makes no decision. Let $D_i(\boldsymbol{x}) = [d_{i,1}(\boldsymbol{x}), \cdots, d_{i,c}(\boldsymbol{x})]^T \in [0,1]^c$ be the output of classifier D_i for input $\boldsymbol{x}$. The entry $d_{i,j}(\boldsymbol{x}) \in [0,1]$ denotes the support that D_i renders for the hypothesis that $\boldsymbol{x}$ comes from class ω_j. To find the class vote of classifier D_i, we *harden* the classification decision by the *maximum membership* formula

$$\text{choose class } \omega_k \Longleftrightarrow d_{i,k}(\boldsymbol{x}) = \max_j \{d_{i,j}(\boldsymbol{x})\}. \tag{3}$$

By this rule, we can formulate the hardened classification decision of each D_i as the binary vector D_i^h (h stands for "hardened") containing 1 at position k and 0 elsewhere, i.e.,

$$d_{i,j}^h(\boldsymbol{x}) = \begin{cases} 1, & \text{if } j = k \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The *majority vote* aggregation $\mathcal{F}_{\text{maj}}$ is given by

$$\mathcal{F}_{\text{maj}} \equiv D(\boldsymbol{x}) = [d_1(\boldsymbol{x}), \cdots, d_c(\boldsymbol{x})]^T, \quad d_j(\boldsymbol{x}) \in \{0,1\} \tag{5}$$

and

$$d_j(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \sum_{i=1}^{L} d_{i,j}^h(\boldsymbol{x}) = \max_k \sum_{i=1}^{L} d_{i,k}^h(\boldsymbol{x}) \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The result is a binary vector with element 1 corresponding to the most supported class, and 0 elsewhere. More than one element with value 1 means a tie. To find a single class label for $\boldsymbol{x}$, the ties are broken randomly. In two-class problems ($c = 2$), an odd number of classifiers L precludes any ties.

Another approach to aggregation of classification decisions is *decision templates* (DT's). It is a simple, intuitive, and robust aggregation idea that evolved from the *fuzzy templates* [25], [37], [38]. Let $\{D_1, \cdots, D_L\}$ be the set of L classifiers. The classifier outputs can be organized in a *decision profile* (DP) as the matrix shown in (7) at the bottom of the page.

Let $Z = \{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N\}$, $\boldsymbol{z}_j \in \Re^n$, be the crisply labeled training data set. The *decision template* $\text{DT}_i(Z)$ of class ω_i is the $L \times c$ matrix with a (k,s)th element $dt_i(k,s)(Z)$[3] computed by

$$dt_i(k,s)(Z) = \frac{\sum_{j=1}^{N} \text{Ind}(\boldsymbol{z}_j, \omega_i)\, d_{k,s}(\boldsymbol{z}_j)}{\sum_{j=1}^{N} \text{Ind}(\boldsymbol{z}_j, \omega_i)}, \quad k = 1, \cdots, L,\ s = 1, \cdots, c \tag{8}$$

where $\text{Ind}(\boldsymbol{z}_j, \omega_i)$ is an indicator function with value 1 if $\boldsymbol{z}_j$ has a crisp label ω_i, and 0 otherwise [38]. To simplify the notation, $\text{DT}_i(Z)$ will be denoted by DT_i.

The decision template DT_i for class ω_i is the average of the decision profiles of the elements of the training set Z labeled in class ω_i. When $\boldsymbol{x} \in \Re^n$ is submitted for classification, the DT scheme matches $\text{DP}(\boldsymbol{x})$ to DT_i, $i = 1, \cdots, c$, and produces the soft class labels

$$\mu_D^i(\boldsymbol{x}) = \mathcal{S}(\text{DT}_i, \text{DP}(\boldsymbol{x})), \qquad i = 1, \cdots, c \tag{9}$$

where $\mathcal{S}$ is interpreted as a *similarity* measure. The higher the similarity between the decision profile of the current $\boldsymbol{x}$ ($\text{DP}(\boldsymbol{x})$) and the decision template for class ω_i (DT_i), the higher the support for that class ($\mu_D^i(\boldsymbol{x})$). Notice that we use the word "similarity" in a broad sense, meaning "degree of match" or "likeness," etc. Regarding the arguments of $\mathcal{S}$ as fuzzy sets on some universal set with $L \cdot c$ elements, various fuzzy measures of similarity can be used. Let A and B be fuzzy sets on

[3] dt is used to denote the entries in the matrix DT. Although composed of two letters, dt designates a single element.

$$\text{DP}(\boldsymbol{x}) = \begin{bmatrix} d_{1,1}(\boldsymbol{x}) & \cdots & d_{1,j}(\boldsymbol{x}) & \cdots & d_{1,c}(\boldsymbol{x}) \\ \cdots & & & & \\ d_{i,1}(\boldsymbol{x}) & \cdots & d_{i,j}(\boldsymbol{x}) & \cdots & d_{i,c}(\boldsymbol{x}) \\ \cdots & & & & \\ d_{L,1}(\boldsymbol{x}) & \cdots & d_{L,j}(\boldsymbol{x}) & \cdots & d_{L,c}(\boldsymbol{x}) \end{bmatrix} \tag{7}$$

output of classifier $D_i(\boldsymbol{x})$ (row i)

support from classifiers $D_1 \cdots D_L$ for class ω_j (column j)

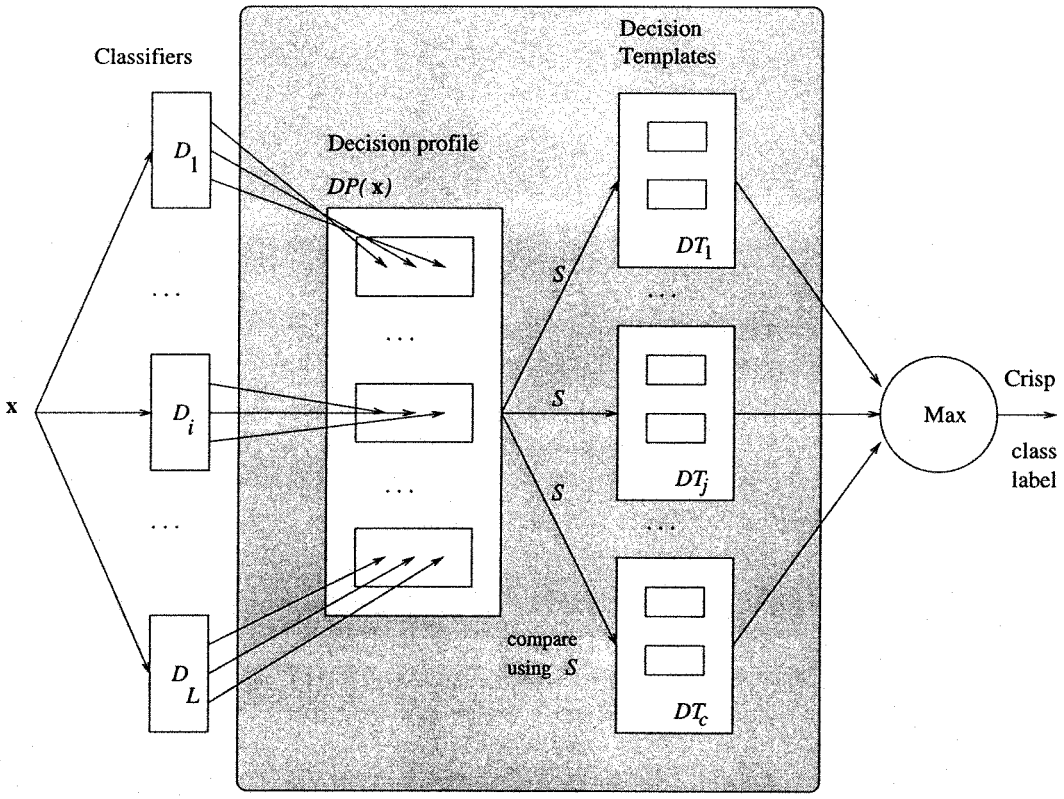

Fig. 2. Architecture of the decision templates classifier fusion scheme. D_i is the ith classifier in the pool, DP($\boldsymbol{x}$) is the decision profile for the current input $\boldsymbol{x}$, DT$_j$ is the decision template for class ω_j, and S is a measure of similarity between two fuzzy sets.

$U = \{u_1, \cdots, u_q\}$. In this study, we used the S_1 measure of similarity based on union and intersection [39]:

$$S_1(A, B) \equiv \frac{||A \cap B||}{||A \cup B||} \tag{10}$$

where $||\zeta||$ denotes the relative cardinality of the fuzzy set ζ on U:

$$||\zeta|| = \frac{1}{q} \sum_{i=1}^{q} \mu_\zeta(u_i). \tag{11}$$

Fig. 2 illustrates how the DT scheme operates. The input vector $\boldsymbol{x}$ is submitted to all L classifiers, and each one produces a c-dimensional vector with support for the classes from Ω. Pooling the outputs, the decision profile DP for that $\boldsymbol{x}$ is constituted as an $L \times c$ matrix. The DP is then compared via similarity measure S_1 with the c decision templates $\mathrm{DT}_1, \cdots, \mathrm{DT}_c$ which are matrices of the same size, calculated in advance using Z in (8). The result of the comparison, i.e., the values of S, are submitted to the Max collector, and the class label of the closest decision template is assigned to $\boldsymbol{x}$.

Example: Consider as an example a three-class problem with four individual classifiers (the dimensionality of the feature space n is irrelevant for the example). Let the classifier outputs for some $\boldsymbol{x} \in \Re^n$ be

$$\begin{aligned} D_1(\boldsymbol{x}) &= [0.6, 0.1, 0.8]^T \\ D_2(\boldsymbol{x}) &= [0.1, 0.7, 0.5]^T \\ D_3(\boldsymbol{x}) &= [0.9, 0.0, 0.8]^T \\ D_4(\boldsymbol{x}) &= [0.6, 0.3, 0.3]^T. \end{aligned}$$

Majority Vote: First, we design the hardened vectors as

$$\begin{aligned} D_1^h(\boldsymbol{x}) &= [0, 0, 1]^T \\ D_2^h(\boldsymbol{x}) &= [0, 1, 0]^T \\ D_3^h(\boldsymbol{x}) &= [1, 0, 0]^T \\ D_4^h(\boldsymbol{x}) &= [1, 0, 0]^T \end{aligned}$$

and then the aggregated vector is

$$D(\boldsymbol{x}) = [1, 0, 0]^T$$

according to which $\boldsymbol{x}$ is labeled in class ω_1.

Decision Templates: Prior to classifying $\boldsymbol{x}$ we have to calculate the DT's for the three classes using the training data. Assume that we have obtained the following results:

$$\mathrm{DT}_1 = \begin{bmatrix} 0.6, 0.1, 0.0 \\ 0.7, 0.2, 0.5 \\ 0.8, 0.3, 0.0 \\ 0.4, 0.0, 0.0 \end{bmatrix}, \quad \mathrm{DT}_2 = \begin{bmatrix} 0.0, 0.6, 0.3 \\ 0.2, 0.7, 0.4 \\ 0.1, 0.9, 0.0 \\ 0.0, 0.2, 0.0 \end{bmatrix},$$

$$\mathrm{DT}_3 = \begin{bmatrix} 0.1, 0.1, 0.7 \\ 0.1, 0.2, 0.5 \\ 0.0, 0.3, 0.9 \\ 0.3, 0.0, 0.5 \end{bmatrix}.$$

The decision profile is formed from the classifier outputs as

$$\mathrm{DP}(\boldsymbol{x}) = \begin{bmatrix} 0.6, 0.1, 0.8 \\ 0.1, 0.7, 0.5 \\ 0.9, 0.0, 0.8 \\ 0.6, 0.3, 0.3 \end{bmatrix}.$$

The similarities between DP($\boldsymbol{x}$) and the three DT's are, respec-

tively,

$$\mathcal{S}(\mathrm{DP}(\boldsymbol{x}), \mathrm{DT}_1) = 0.4655$$
$$\mathcal{S}(\mathrm{DP}(\boldsymbol{x}), \mathrm{DT}_2) = 0.2969$$
$$\mathcal{S}(\mathrm{DP}(\boldsymbol{x}), \mathrm{DT}_3) = 0.4921$$

which labels $\boldsymbol{x}$ in class ω_3. ■

As the example shows, the two aggregations can lead to different class labels. Our previous studies found that DT's rate high among a number of aggregation techniques [25]. In the course of this study, we found experimentally that majority vote is a good option for two-class problems with predominantly binary features and logistic model as individual classifiers.

III. Genetic Algorithms for Classifier Fusion Design

GAs have been used in various pattern recognition problems. There is a huge body of literature on neural-network classifiers optimized by GAs and other evolutionary algorithms which is beyond the scope of the current study (see, e.g., [40], [41]). Using GAs for nearest neighbor editing is proposed in [42]–[43]. The most popular pattern recognition niche for GAs has been feature selection because

- the encoding of a feature subset into a chromosome is straightforward
- the function that is optimized does not need to be smooth and can, therefore, be directly the classification accuracy (direct error minimization); classification accuracy/error is a notoriously difficult optimization criterion, making feature selection the most challenging task in pattern recognition [44]–[46].

Although many authors advocate feature selection by GAs [47]–[51], others are skeptical [52], and warn that the results are often not as good as expected, compared with other (simpler!) feature selection algorithms [53].

Kuncheva [22] uses GAs to select features for the individual classifiers in a classifier fusion scheme. The subset of features used by an individual classifier is encoded as a chromosome. The GA evolves the feature subsets so that each population consists of highly accurate *individual* classifiers. The elitist selection procedure was modified slightly so that the classifiers which produce the highest accuracy *as a combination* (by majority vote) are retained together with the offspring that have the highest individual fitness. In other words, individual fitness and group fitness were applied simultaneously. The group fitness was responsible for retaining L members of the population, whereas the individual fitness was applied to select the rest of the population. Notice that, using the standard elitist procedure, the individual classifiers from the best combination could be dropped because they are not necessarily the best individual classifiers. This selection scheme, however, has no mechanism to prevent degeneration of the population toward the same best individual classifier. Therefore, when the GA converges, the combined classifier is most likely to have identical constituents, so no improvement over the individual accuracy is achieved.

In this study, we try two different ideas in which the fitness function is the accuracy of the *combination*, and not that of the individual classifiers. Why do we think that genetic algorithms are good for this problem? It is clear that no optimization algorithm is a panacea [54], and the choice should depend on the type of problem we are solving. We believe that feature selection for multiple-classifier systems is a proper task for GAs. First, the same set of arguments for using GAs for feature selection (listed at the beginning of the section) hold for the multiclassifier model. Second, in feature selection, there are a number of good suboptimal algorithms that avoid extensive search (e.g., sequential forward and backward search and their extensions, floating selection methods, etc.), whereas in selecting features for a *multiple-classifier* scheme, there are no such methods. Presumably, the fitness function (accuracy of the combination) is highly multimodal, which impedes using deterministic optimization methods effectively. Third, encoding of the problem in terms of GAs is straightforward in contrast with, e.g., optimization by GAs of the weights of a neural network, or the membership functions of fuzzy if–then systems.

Version1—SelectionofDisjointFeatureSubsets: Dependence between the individual classifiers in a multiple classifier system can be a good or a bad thing. If we can make use of this dependence, e.g., when the correlation is negative, this can lead to higher accuracy than when using independent classifiers. On the other hand, when the classifiers are dependent so that they misclassify the same vectors, then nothing can be done, and the combination can be even worse than the single best classifier. Often, when we have no information about the individual classifiers, we hope for independent classifiers because this guarantees higher accuracy than the best individual (if we use majority vote) or we can enforce a probabilistic model and use a Bayes-optimal aggregation (e.g., probabilistic product). Version 1 of the GA selects disjoint subsets of features, hoping that this will contribute toward independence of the classifiers.

The chromosome consists of n positions, one for each feature. The ith position can take one of the following values $\{0, 1, \cdots, L\}$, where 0 means that the ith feature is not used, and an integer k, $1 \leq k \leq L$ means that the ith feature is used by classifier D_k. This encoding ensures that classifiers use disjoint feature subsets. Mutation is performed by taking one of the possible $L + 1$ values at random.

Version 2—Selection of Classifiers and (Possibly Overlapping) Feature Subsets: We use the following encoding. The first n positions in the chromosome correspond to the n features. The string is augmented by L positions which specify the type of each individual classifier. A possible encoding for a three-classifier system and ten features is presented in Fig. 3. The Venn diagram shows the semantic of the codes for the first n positions. Each feature can be used in 2^3 ways (not used, used by any single classifier, used by any two, and by all three), the possible values for the example are $\{0, 1, \cdots, 7\}$.

The two GA versions differ only by the encoding. Their implementation follows the traditional pattern given below.

The GA Algorithm:

1) Pick the parameters of the GA:
   a) population size m (even)
   b) terminal number of generations $T_{\max}$
   c) mutation probability P_m.
2) Generate a random population of m chromosomes and calculate their fitness values.

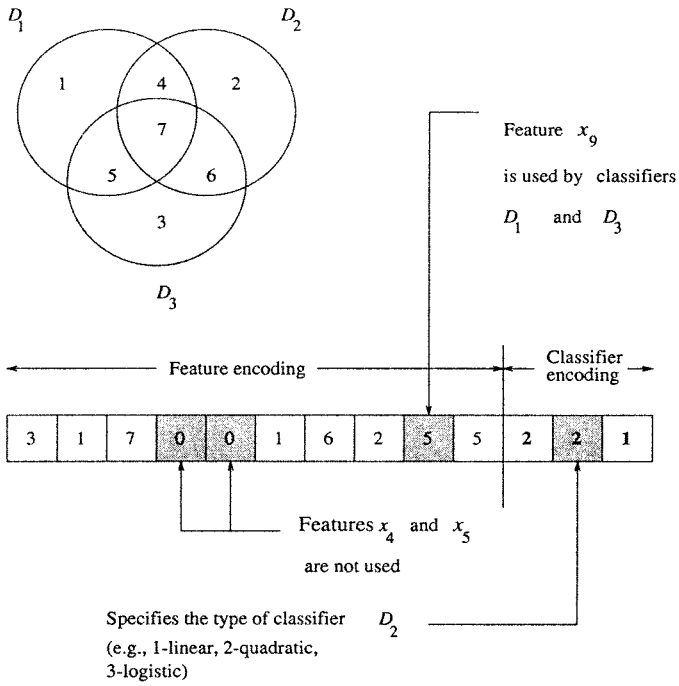

Fig. 3. A possible encoding for a three-classifier system for a ten-dimensional problem.

3) For $i = 1$: $T_{\max}$
   a) assuming that the whole population is the mating set, select $m/2$ couples of parents from the curfrent population (repetitions are allowed)
   b) perform (one-point) crossover to generate m offspring chromosomes
   c) mutate the offspring according to the mutation probability
   d) calculate the fitness values of the mutated offspring
   e) pool the offspring and the current population, and select as the next population the m chromosomes with the highest fitness.
4) End i.

The limit number of generations, population size, and mutation probability are the tunable parameters of this GA model. We assume that the whole population is allowed to reproduce, the crossover probability is set to 1.0, and since elitist selection is used, the generation gap is not fixed. This drives the model closer to a random search, with the main emphasis being on exploration. The choice was guided by the assumption that the fitness function is highly multimodal, and the algorithm should be given a chance to *hit* a good solution rather than *elaborate* one.

IV. Experiments

The hypothesis was that, using GAs for designing the classifier fusion system, we can achieve better results than that produced by the best single classifier of a prespecified type. Notice that our aim was not to show that the combined decision is better than that of any of the *constituents* (this would be easy!). The experiments with GA Version 1 and Version 2 were confined to three basic types of classifiers: 1) linear discriminant classifier (LDC), 2) quadratic discriminant classifier (QDC) [44], and 3) logistic classifier [55].

The following data sets were used:

1) *Heart Data:* This data set was taken from the UCI machine learning database (deposited by Dr. Robert Detrano, V.A. Medical Center, Long Beach and Cleveland Clinic Foundation). The data set is denoted as "Cleveland database" in the technical report by Prechelt [56]. The problem is to distinguish between cases with and without heart disease on the basis of 18 features (13 binary and 5 continuous valued) from clinical and laboratory examination of the patient. The database contains 303 cases, with no missing values. Previous results reported in the database description find the misclassification rate to vary between 21 and 26%.
2) *Satimage Data:* These data have been generated from the Landsat Multi-Spectral Scanner image data. They consist of 6 435 patterns (pixels) with 36 attributes (4 spectral bands $\times$ 9 pixels in a 3 $\times$ 3 neighborhood). Pixels are labeled in one of six classes, and are presented in random order in the database. The classes are: red soil (23.82%), cotton crop (10.92%), gray soil (21.10%), damp gray soil (9.73%), soil with vegetation stubble (10.99%), and very damp gray soil (23.43%). What makes this database attractive is the large sample size; numerical, equally ranged features; no missing values; and compact classes of approximately equal size, shape, and prior probabilities.
3) *Letters Data:* This set was taken from the UCI machine learning database. The objective is to classify distorted black and white images of the 26 letters from English alphabet. Each image is described by 16 continuous-valued features. The total number of images is 20 000. The best error rate reported in the database description is a little lower than 20%. We found a test error (tenfold cross validation) of 11.50 $\pm$ 1.13% with the quadratic discriminant classifier (QDC). Since the majority of the classes

are easy to recognize, we decided to take only the two most often confused classes, which turned out to be the letters "H" and "K."

4) *Forensic Glasses Data:* This data set consists of 214 nine-dimensional vectors, each describing the chemical content of a fragment of glass. The glasses are labeled into six classes [57]: window float glass (32.71%), window nonfloat glass (35.51%), vehicle window glass (7.94%), containers (6.07%), tableware (4.21%), and vehicle headlamps (13.55%). We tried to reproduce the tenfold cross-validation experiments from [57] using the same tenfold partition of the data set. The disadvantage of this data set is the small proportions of three of the six classes, which leads to an error rate between 23.4% for the nearest neighbor classifier (in our experiments, we obtained 27.62%) and 38% for the linear discriminant classifier (our result is 39.56%).

With the Satimage data, we used hold-up experiments splitting the set into two halves, and using the first half (3218 cases) for training and the second half (3217 for test). With the other data sets tenfold cross validation was used.

For comparison with the GA-designed combination scheme, we tried simple classifier models: linear discriminant classifier (LDC) and quadratic discriminant classifier (QDC) assuming normal densities, Parzen classifier, the nearest neighbor classifier (1-nn) (all described in [44]), and the logistic classifier (LOG) [55]. For all classifiers, the PRTOOLS toolbox for Matlab [58] was used. Table I shows the training and test error rates with the four data sets. Showing training results is important here because a comparison on test results only can be misleading. Too versatile classifiers can be overtrained, and if test results are the only testimony, such classifiers will be marked as unsuccessful. An example is the radial-basis function (RBF) neural network. Applied to the Heart data, with at most 200 nodes at the hidden layer, the RBF classifier resulted in an average of 4.11% training error (four times smaller than the error rate with the logistic classifier used here) and 45.21% test error (more than twice the LOG classifier test error). This poor generalization does not invalidate the RBF model, but shows that both training and test results have to be displayed and analyzed, especially when novel models are considered.

With each data set, we picked one of the simple classifiers, and proceeded with the following further experiments.

1) A feature selection by the sequential backward selection (SBS). The error of the classifier was used as the criterion function J. The procedure starts with the whole set of features (of cardinality n), and discards one feature at each step, as shown in Fig. 4.
2) A GA for feature selection [53], [55]. The GA described in Section III is used, where the fitness function is the classification accuracy using the feature subset represented as a chromosome. The chromosome is a binary string of length n. The ith bit takes values 0 if the feature is not in the subset or 1 if the feature is in the subset.
3) Version 1 of GA with the fitness function being the classification accuracy of the *combination*, either by majority vote or by decision templates $(\mathrm{DT}(S_1))$.
4) Version 2 of GA with the same fitness function.

TABLE I
AVERAGED ERROR RATES (IN%) AND STANDARD DEVIATIONS FROM TENFOLD CROSS-VALIDATION EXPERIMENTS AND FIVE BASIC CLASSIFIER MODELS. [1] COVARIANCE MATRIX CLOSE TO SINGULAR (EITHER TOO FEW DATA OR TOO LITTLE INTERCLASS VARIANCE. [2] TOO MANY BINARY VARIABLES. [3] THE DATA SET IS TOO LARGE AND COMPUTATIONALLY DEMANDING

Data set		LDC	QDC	LOG	Parzen	1-nn	Method
Heart	Training	28.45 ±12.47	N/A[1]	17.53 ±1.08	N/A[2]	N/A[2]	10-fold cross-validation
	Test	24.12 ±14.58	N/A[1]	20.12 ±7.04	N/A[2]	N/A[2]	
Satimage	Training	15.60	9.94	15.23	N/A[3]	N/A[3]	Hold-on
	Test	16.23	15.05	16.97	N/A[3]	N/A[3]	
Letters	Training	6.81 ±0.65	5.02 ±0.68	4.53 ±0.70	0 ±0	8.96 ±1.01	10-fold cross-validation
	Test	9.11 ±5.18	8.39 ±5.36	8.55 ±3.81	9.21 ±5.90	8.86 ±6.50	
Forensic glasses	Training	33.12 ±1.34	N/A[1]	28.14 ±1.67	24.48 ±2.66	27.43 ±1.39	10-fold cross-validation
	Test	35.56 ±15.46	N/A[1]	39.54 ±15.65	35.40 ±8.24	27.62 ±7.84	

The results with the individual data sets are given in Tables II–V. The lowest error is marked in boldface.

All GAs were run with the following set of parameters:

- population size $m = 10$
- terminal number of generations $T_{\max} = 100$
- mutation probability $P_m = 0.2$.

We initialized the ten runs in the cross-validation experiment in the same way, so that each GA started from the same configuration (feature subset or set of classifiers), and used different training and testing data. We used decision templates for aggregation for the Satimage, Letters, and Forensic glasses data, and majority vote for the Heart data. This was decided on the basis of the improvement of the combination accuracy over the best individual accuracy in a series of preliminary experiments with the training data.

V. DISCUSSION AND CONCLUSIONS

1) *Training Error Rate:* With all four data sets, GA Version 2 yielded the smallest training error rate. This shows that *the GA has done the job for which we have been using it:* based on the training data, the multiple-classifier system designed by the GA appeared to be more accurate than the best individual classifier (not necessarily used as one of the constituents of the system).
2) *Test Error Rate:* A good training rate is often at the expense of some deterioration of generalization (overtraining). The experiment with the RBF network with 200 hidden nodes and the Heart data is an example of this phenomenon. Ideally, the classifier should have both small training and test errors. With the Satimage and Forensic glasses data, the system designed by the GA (Version 2) gave better test results than the other classifiers. With the other two sets, the GA appeared

The SBS algorithm

1. Set $X = \{x_1, \ldots, x_n\}$.
2. For $i = n$: downto : 2,
   (a) Take one element of X out and calculate the criterion value J for the remaining subset of cardinality $i - 1$.
   (b) Return the element back in X and proceed in this way with all elements of X.
   (c) Assign as the new X the subset of cardinality $i - 1$ with the lowest error rate (lowest J).
   (d) Store the results found at step i: the new X and the respective value $J(X)$.
3. End i.
4. Choose the smallest of the stored $n - 1$ values of J and retrieve the corresponding X.
5. Return X and J.

Fig. 4. Sequential backward selection (SBS) algorithm for feature selection.

TABLE II
AVERAGE RESULTS (IN%) FROM TENFOLD CROSS-VALIDATION EXPERIMENTS WITH HEART DATA

Method	Training error	Testing error	Comment
LOG	17.53	20.12	18 features
SBS	16.28	**19.12**	9.5 features
GA for feature selection	15.91	23.12	11.4 features, LOG classifier as fitness function
GA for multiple classifiers	16.50	21.73	Version 1, majority vote for aggregation
GA for multiple classifiers	**14.74**	22.12	Version 2, majority vote, classifier choices LOG, LDC

TABLE III
AVERAGE RESULTS (IN%) FROM TWOFOLD CROSS-VALIDATION EXPERIMENTS WITH SATIMAGE DATA

Method	Training error	Testing error	Comment
QDC	9.68	14.60	36 features
SBS	9.54	14.52	33.5 features
GA for feature selection	10.24	14.14	25.5 features, QDC classifier as fitness function
GA for multiple classifiers	12.66	14.54	Version 1, $DT(S_1)$ for aggregation
GA for multiple classifiers	**9.52**	**12.76**	Version 2, $DT(S_1)$, classifier choices QDC,LDC

to be slightly overtrained. Since the classifiers that were used both individually and in combination were simple models, the overtraining is not very high (see, for comparison, the RBF example). GAs offer an option to reduce overtraining. We can include penalty terms in the fitness function accounting for the dependency between the individual classifiers (assuming that independent classifiers will form a group that does not overtrain much) and for the number of features used (smaller subsets should be preferred).

3) *Small Differences:* It is a postulate in pattern recognition that we cannot expect *one* model to do dramatically better than all other models on all data sets. Therefore, the more classifiers and the more data sets we bring in the comparison, the blurrier the comparison picture becomes. Multiple-classifier systems have the advantage that, if we reduce all individual classifiers to the *best* individual classifier, then the system will be no better and no worse than the best individual classifier. Therefore, the combination aims at a lower error rate than that of the best individual classifier. Although theoretically proven, this is only a possibility, depending on the success (or luck!) in designing the two-level scheme. By and large, multiple-classifier systems outperform the best individual *constituent* classifier, but if compared with another individual classifier, the relationship could be the reverse. For example, if the multiple-classifier system uses LDC as the individual classifiers, then it is very likely that it will be outperformed by an individual neural network classifier (a multilayer perceptron). This is a possible explanation of the small differences in our experiments between the multiple-classifier systems and the best individual classifiers.

4) *Why Was GA Version 1 Not Successful Here?:* Selecting disjoint subsets of features increases the chance of obtaining a group of independent classifiers, but it also can lead to degraded performance of the individual classifiers if too few features are used. In the data sets used here,

TABLE IV
AVERAGE RESULTS (IN%) FROM TENFOLD CROSS-VALIDATION EXPERIMENTS WITH LETTERS DATA

Method	Training error	Testing error	Comment
LOG	6.56	7.12	16 features
SBS	6.17	7.19	12.2 features
GA for feature selection	6.30	**6.99**	13.2 features, LOG classifier as fitness function
GA for multiple classifiers	7.09	8.14	Version 1, $DT(S_1)$ for aggregation
GA for multiple classifiers	**6.11**	7.80	Version 2, $DT(S_1)$ vote, classifier choices LOG, LDC, QDC

TABLE V
AVERAGE RESULTS (IN%) FROM TENFOLD CROSS-VALIDATION EXPERIMENTS WITH FORENSIC GLASSES DATA

Method	Training error	Testing error	Comment
LOG	28.14	39.54	9 features
SBS	27.62	37.39	6.9 features
GA for feature selection	27.52	40.39	7.7 features, LOG classifier as fitness function
GA for multiple classifiers	27.62	39.26	Version 1, $DT(S_1)$ vote for aggregation
GA for multiple classifiers	**25.35**	**34.55**	Version 2, $DT(S_1)$ vote, classifier choices LOG, LDC

there were few or no redundant features, which is demonstrated by the good error rates of classifiers using all of the features or using the selected by SBS subsets. Therefore, almost all features were necessary for achieving a low error rate, and splitting them into three subsets was not a successful line of designing the multiple-classifier scheme. We expect that this version of the GA will be very useful in problems with higher dimensionality (hundreds of features) where using all features together will be infeasible, and selecting a single subset can ignore good alternative solutions which are potentially good constituents of a multiple-classifier system. In any case, GA Version 1 is simpler and more intuitive, and is worth trying first on such problems.

GAs have various tuning parameters (e.g., m, P_m, and $T_{\max}$). We did not explore the effect of changing these parameters on the GAs. The execution times are high, as with any GA, compared to these for the individual classifiers. This, however, is not a serious obstacle if we design a multiple-classifier system off line.

The main message of this study is that GAs and other evolutionary algorithms offer an intuitive *automatic* way to design multiple-classifier systems instead of picking the classifier types and selecting the combination ad hoc by the designer. The GA-designed systems in our experiments were, in most cases, more accurate than the best individual classifier in the system, and also outperformed a number of other classifier designs. For now, we have considered the selection of features for a preliminary fixed (small) number of classifiers, and the selection of classifier types from a (small) set of basic models. This GA-based design scheme can be extended straightforwardly to a higher number of classifiers ($L > 3$) and classifier models. The chromosome length does not increase with L for GA Version 1, and increases as $n+L$ for GA Version 2, but the search space does expand exponentially. Extensive experimentation with the GAs in such cases could prove computationally demanding. Therefore, optimization in terms of execution time of the algorithm and the corresponding software is an important future research topic.

REFERENCES

[1] P. D. Gader, M. A. Mohamed, and J. M. Keller, "Fusion of handwritten word classifiers," *Pattern Recognit. Lett.*, vol. 17, pp. 577–584, 1996.

[2] J. M. Keller, P. Gader, H. Tahani, J.-H. Chiang, and M. Mohamed, "Advances in fuzzy integration for pattern recognition," *Fuzzy Sets Syst.*, vol. 65, pp. 273–283, 1994.

[3] H.-J. Kang, K. Kim, and J. H. Kim, "A framework for probabilistic combination of multiple classifiers at an abstract level," *Eng. Appl. Artif. Intell.*, vol. 10, no. 4, pp. 379–385, 1997.

[4] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 3, pp. 226–239, 1998.

[5] L. Lam and C. Y. Suen, "Optimal combination of pattern classifiers," *Pattern Recognit. Lett.*, vol. 16, pp. 945–954, 1995.

[6] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 405–410, 1997.

[7] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their application to handwriting recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 418–435, 1992.

[8] R. A. Jacobs, "Methods for combining experts' probability assessments," *Neural Comput.*, vol. 7, pp. 867–888, 1995.

[9] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, pp. 79–87, 1991.

[10] M. I. Jordan and L. Xu, "Convergence results for the EM approach to mixtures of experts architectures," *Neural Networks*, vol. 8, pp. 1409–14 312, 1995.

[11] S. J. Nowlan and G. E. Hinton, "Evaluation of adaptive mixtures of competing experts," *Advances in Neural Information Processing Systems 3*, pp. 774–780, 1991.

[12] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon, 1995.

[13] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other ensemble methods," *Neural Comput.*, vol. 6, pp. 1289–1301, 1994.

[14] J. A. Benediktsson, J. R. Sveinsson, J. I. Ingimundarson, H. Sigurdsson, and O. K. Ersoy, "Multistage classifiers optimized by neural networks and genetic algorithms," *Nonlinear Anal., Theory, Methods Appl.*, vol. 30, no. 3, pp. 1323–1334, 1997.

[15] J. A. Benediktsson and P. H. Swain, "Consenus theoretic classification methods," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 688–704, 1992.

[16] K.-C. Ng and B. Abramson, "Consensus diagnosis: A simulation study," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 916–928, 1992.

[17] R. Battiti and A. M. Colla, "Democracy in neural nets: Voting schemes for classification," *Neural Networks*, vol. 7, pp. 691–707, 1994.
[18] E. Filippi, M. Costa, and E. Pasero, "Multy-layer perceptron ensembles for increased performance and fault-tolerance in patern recognition tasks," in *IEEE Int. Conf. Neural Networks*, Orlando, FL, 1994, pp. 2901–2906.
[19] E. Alpaydin and M. I. Jordan, "Local linear perceptrons for classification," *IEEE Trans. Neural Networks*, vol. 7, no. 3, pp. 788–792, 1996.
[20] E. Alpaydin, "Voting over multiple condensed nearest neighbors," *Artif. Intell. Rev.*, vol. 11, pp. 115–132, 1997.
[21] K. Chen, L. Wang, and H. Chi, "Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 11, no. 3, pp. 417–445, 1997.
[22] L. I. Kuncheva, "Genetic algorithm for feature selection for parallel classifiers," *Inform. Processing Lett.*, vol. 46, pp. 163–168, 1993.
[23] D. M. J. Tax, R. P. W. Duin, and M. van Breukelen, "Comparison between product and mean classifier combination rules," in *Proc. Workshop Statist. Pattern Recognit.*, Prague, Czech Republic, 1997.
[24] M. van Breukelen, R. P. W. Duin, D. M. J. Tax, and J. E. den Hartog, "Combining classifiers for the recognition of handwritten digits," in *1st IAPR TC1 Workshop Statist. Techniques in Pattern Recognit.*, Prague, Czech Republic, 1997, pp. 13–18.
[25] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognit.*, 1999.
[26] L. Lam and C. Y. Suen, "Application of majority voting to pattern recognition: An analysis of its behavior and performance," *IEEE Trans. Syst., Man, Cybern.*, vol. 27, no. 5, pp. 553–568, 1997.
[27] Y. S. Huang and C. Y. Suen, "A method of combining multiple experts for the recognition of unconstrained handwritten numerals," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 90–93, 1995.
[28] S.-B. Cho and J. H. Kim, "Combining multiple neural networks by fuzzy integral and robust classification," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 380–384, 1995.
[29] S. Hashem, "Optimal linear combinations of neural networks," *Neural Networks*, vol. 10, no. 4, pp. 599–614, 1997.
[30] S. Hashem, B. Schmeiser, and Y. Yih, "Optimal linear combinations of neural networks: An overview," in *IEEE Int. Conf. Neural Networks*, Orlando, FL, 1994, pp. 1507–1512.
[31] V. Tresp and M. Taniguchi, "Combining estimators using non-constant weighting functions," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: M.I.T. Press, 1995.
[32] Y. S. Huang and C. Y. Suen, "A method of combining multiple classifiers—A neural network approach," in *12th Int. Conf. Pattern Recognit.*, Jerusalem, Israel, 1994, pp. 473–475.
[33] Y. Lu, "Knowledge integration in a multiple classifier system," *Appl. Intell.*, vol. 6, pp. 75–86, 1996.
[34] G. Rogova, "Combining the results of several neural network classifiers," *Neural Networks*, vol. 7, pp. 777–781, 1994.
[35] D. B. Fogel, *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*, 2nd ed. New York: IEEE Press, 2000.
[36] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, PA: Addison-Wesley, 1989.
[37] L. I. Kuncheva, J. C. Bezdek, and M. A. Sutton, "On combining multiple classifiers by fuzzy templates," in *Proc. NAFIPS'98*, Pensacola, FL, 1998, pp. 193–197.
[38] L. I. Kuncheva, R. K. Kounchev, and R. Z. Zlatev, "Aggregation of multiple classification decisions by fuzzy templates," in *3rd European Congr. Intell. Technol. Soft Comput. EUFIT'95*, Aachen, Germany, Aug. 1995, pp. 1470–1474.
[39] D. Dubois and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*. New York: Academic, 1980.
[40] A. J. Van Rooij, L. C. Jain, R. P. Johnson, and A. J. F. Van Rooij, *Neural Network Training Using Genetic Algorithms, Machine Perception and Artificial Intelligence*. Singapore: World Scientific, 1997, vol. 26.
[41] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sept. 1999.
[42] L. I. Kuncheva, "Editing for the k-nearest neighbors rule by a genetic algorithm," *Pattern Recognit. Lett.*, vol. 16, pp. 809–814, 1995.
[43] L. I. Kuncheva and J. C. Bezdek, "On prototype selection: Genetic algorithms or random search?," *IEEE Trans. Syst., Man, Cybern.*, vol. 28, no. 1, pp. 160–164, 1998.
[44] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley , 1973.
[45] E. A. Patrick, *Fundamentals of Pattern Recognition*. Englewood Cliffs, NJ: Prentice-Hall, , 1972.
[46] L. A. Rastrigin and R. H. Erenstein, *Method of Collective Recognition* (in Russian). Moscow: Energoizdat, 1982.
[47] Y. Chtioui, D. Bertrand, and D. Barba, "Feature selection by a genetic algorithm. Application to seed discrimination by artificial vision," *J. Sci. Food Agric.*, vol. 76, pp. 77–86, 1998.
[48] R. Leardi, "Application of a genetic algorithm to feature selection under full validation conditions and to outlier detection," *J. Chemometrics*, vol. 8, pp. 65–79, 1994.
[49] ——, "Genetic algorithms in feature selection," in *Genetic Algorithms in Molecular Modeling*, J. Devillers, Ed. London: Academic, 1996, pp. 67–86.
[50] B. Sahiner, H.-P. Chan, D. Wei, N. Petrick, and M. A. Helvie, "Image feature selection by a genetic algorithm: Application to classification of mass and normal breast tissue," *Med. Phys.*, vol. 23, no. 10, pp. 1671–1684, 1996.
[51] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognit. Lett.*, vol. 10, pp. 335–347, 1989.
[52] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognit. Lett.*, vol. 15, pp. 1119–1125, 1994.
[53] A. K. Jain and J. Mao, "Guest editorial: Special issue on artificial neural networks and statistical pattern recognition," *IEEE Trans. Neural Networks*, vol. 8, no. 1, pp. 1–3, 1997.
[54] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.
[55] J. A. Anderson, "Logistic discrimination," in *Classification, Pattern Recognition and Reduction of Dimensionality*, P. R. Krishnaiah and L. N. Kanal, Eds. Amsterdam: North-Holland, 1982, vol. 2 of Handbook of Statistics, pp. 169–191.
[56] L. Prechelt, "PROBEN1—A set of neural network benchmark problems and benchmarking rules," Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep. 21/94, 1994.
[57] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, England: University Press, 1996.
[58] R. P. W. Duin, "PRTOOLS(Version 2). A Matlab toolbox for pattern recognition," Pattern Recognit. Group, Delft Univ. Technol., Delft, The Netherlands, June 1997.

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.

Until 1997, she worked at the Central Laboratory of Biomedical Engineering, Bulgarian Academy of Sciences, as a Senior Research Associate. She is currently a Lecturer at the School of Informatics, University of Wales, Bangor, U.K. Her interests include pattern recognition, neural networks, fuzzy classifiers, prototype classifiers, and multiple-classifier systems.

L. C. Jain is a Director/Founder of the knowledge-based Intelligent Engineering Systems (KES) Centre, located in University of South Australia. He ia a fellow of the Institution of Engineers Australia. He has initiated a postgraduate stream by research in the Knowledge Based Intelligent Engineering Systems area. He has presented a number of keynote addresses in International Conferences on Knowledge-Based Systems, Neural Networks, Fuzzy Systems and Hybrid Systems.

He is the Founding Editor-in-Chief of the International Journal of Knowledge-Based Intelligent Engineering Systems and served as an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS. Dr Jain was the technical chair of the ETD2000 International conference in 1995, Publications Chair of the Australian and New Zealand Conference on Intelligent Information Systems in 1996 and the Conference Chair of the International Conference on the Knowledge-based Intelligent Electronics Systems in 1997, 1998 and 1999. He served as the Vice President of the Electronics Association of South Australia in 1997. He is the Editor-in-Chief of the *International Book series on Computational Intelligence*, CRC Press USA. His interests focus on the applications of the novel techniques such as knowledge-based systems, artificial neural networks, fuzzy systems and genetic algorithms and the application of these techniques