Chapter 15

Combining classifiers: Soft computing solutions

Ludmila I. Kuncheva
School of Informatics, University of Wales, Bangor
Bangor, Gwynedd, LL57 1UT, United Kingdom
e-mail: l.i.kuncheva@bangor.ac.uk,
http://www.bangor.ac.uk/*sim*mas00a/

Abstract*

Classifier combination is now an established pattern recognition subdiscipline. Despite the strong aspiration for theoretical studies, classifier combination relies mainly on heuristic and empirical solutions. Assuming that "soft computing" encompasses neural networks, evolutionary computation, and fuzzy sets, we explain how each of the three components has been used in classifier combination.

15.1 Introduction

Let $\mathcal{D} = \{D_1, D_2, \ldots, D_L\}$ be a set of classifiers (we shall also call $\mathcal{D}$ a *team* or *ensemble*), and let $\Omega = \{\omega_1, \ldots, \omega_c\}$ be a set of class labels. Each classifier gets as its input a feature vector $\mathbf{x} = [x_1, \ldots, x_n]^T$, $\mathbf{x} \in \Re^n$ and assigns it to a class label from Ω, i.e., $D_i : \Re^n \rightarrow \Omega$. Alternatively, we may define the classifier output to be a c-dimensional vector with supports to the classes, i.e.,

$$D_i(\mathbf{x}) = [d_{i,1}(\mathbf{x}), \ldots, d_{i,c}(\mathbf{x})]^T. \tag{15.1}$$

* Published in: S.K.Pal and A. Pal (Eds), *Pattern Recognition: From Classical to Modern Approaches, World Scientific*, 2001, 427–451.

Without loss of generality we can restrict $d_{i,j}(\mathbf{x})$ within the interval $[0, 1]$, $i = 1, \ldots, L$, $j = 1, \ldots, c$, and call the classifier outputs "soft labels" (see [7]). Thus, $d_{i,j}(\mathbf{x})$ is the degree of "support" given by classifier D_i to the hypothesis that $\mathbf{x}$ comes from class ω_j (most often $d_{i,j}(\mathbf{x})$ is an estimate of the posterior probability $P(\omega_i|\mathbf{x})$). Combining classifiers means to find a class label for $\mathbf{x}$ based on the L classifier outputs $D_1(\mathbf{x}), \ldots, D_L(\mathbf{x})$. Again, instead of a single label, we can find a vector with c final degrees of support for the classes as a soft label for $\mathbf{x}$, denoted

$$D(\mathbf{x}) = [\mu_1(\mathbf{x}), \ldots, \mu_c(\mathbf{x})]^T. \tag{15.2}$$

If a crisp class label of $\mathbf{x}$ is needed, we can use the maximum membership rule: Assign $\mathbf{x}$ to class ω_s iff,

$$\mu_s(\mathbf{x}) \geq \mu_t(\mathbf{x}), \forall t = 1, \ldots, c. \tag{15.3}$$

Ties are resolved arbitrarily.

We assume that a labeled data set $\mathbf{Z}$ is available, $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, $\mathbf{z}_j \in \Re^n$, which is used to design the classifier combination system: both the individual classifiers and the combiner.

Classifier combination aims at a higher accuracy than that of a single member of the team $\mathcal{D}$. In the past few years, a lot of work has been done towards developing a rigorous theoretical background of classifier combination. Yet, the useful heuristics are a step ahead the theory, and collective effort is being devoted to understanding and explaining why these heuristics work so well.

Classifier combination is called different names in the literature as shown in Table 15.1. It is therefore important to recognize the pressing need for tidying up the shelf by grouping and arranging the existing solutions in a taxonomy.

This chapter surveys soft computing methods in classifier combination. Whatever reservations I might have to the term, we shall assume that *soft computing* covers neural and evolutionary computation, and fuzzy sets. Section 2 explains classifier combination. Section 3 identifies the place of the three components of soft computing within classifier combination tools and techniques, and Section 4 offers a concluding remark.

Table 15.1 Classifier combination "aliases" in the literature

1	combination of multiple classifiers [54; 61; 72; 73; 42];
2	classifier fusion [16; 25; 28; 41; 9];
3	mixture of experts [39; 38; 40; 58];
4	committees of neural networks [8; 21];
5	consensus aggregation [6; 57; 5];
6	voting pool of classifiers [3];
7	dynamic classifier selection [72];
8	composite classifier systems [18];
9	classifier ensembles [21; 22; 62];
10	bagging, boosting, arcing, wagging [62];
11	modular systems [62];
12	collective recognition [60; 2]
13	stacked generalization [71];
14	divide-and-conquer classifiers [13];
15	pandemonium system of reflective agents [64];
16	change-glasses approach to classifier selection [45], etc.

15.2 Classifier combination

15.2.1 *Four approaches*

Table 15.2 shows four approaches to designing a classifier combination system.

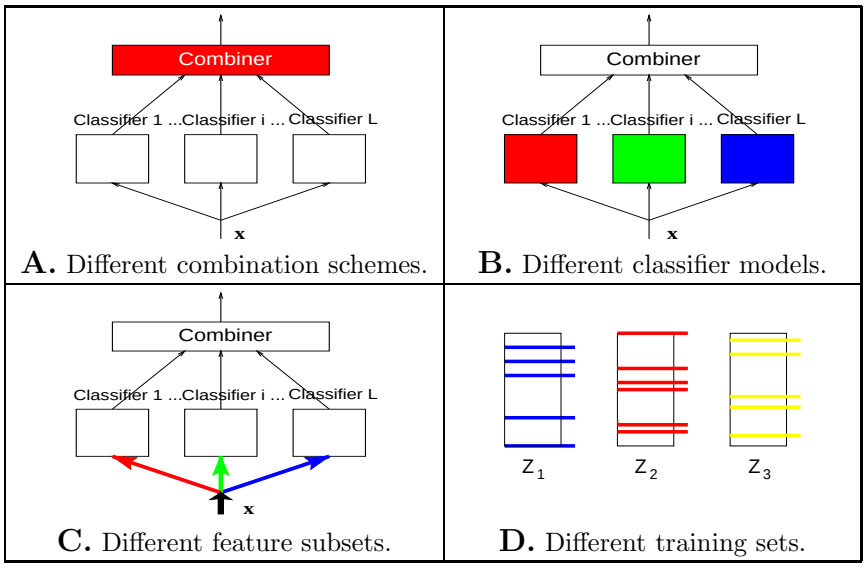
Approach A. We assume that $D_1, \ldots, D_L$ are given (trained in advance), and the problem is to pick a combination scheme and train it if necessary.

Approach B. Any pattern classifier can be used as a team member. Thus, $\mathcal{D}$ can be homogeneous, i.e., formed using identical classifier models (e.g., Multi-Layer Perceptron (MLP) neural networks) with different structure, parameters, initialization protocols, etc. Alternatively, a heterogeneous $\mathcal{D}$ can be designed, as for example in [72].

Approach C. Sometimes it is suitable to build each D_i on an individual subset of features (subspace of $\Re^n$). This is useful when n if large (e.g., a few hundred), and groups of features come from different sources or different data pre-processing. Examples can be found in image and speaker recognition, etc. [12; 43].

Approach D. Many authors are of the opinion that training set alteration is the most powerful of the four approaches as it can lead to a team of *diverse* classifiers [19; 66] whereas none of the other three approaches is suited for that. Diversity among the classifiers in $\mathcal{D}$ means that the individual classifiers D_i's misclassify *different* objects, having at the same time high individual accuracy. This property alone can guarantee a good potential of the team even with the simplest combination schemes. Exploiting this idea, several methods have been proposed to select training subsets of the data set $\mathbf{Z}$.

Table 15.2 Four approaches to designing a classifier combination system



(1) *Partition the data randomly* into L parts and use a different part to train each classifier.
(2) *Boosting*: Start with a classifier D_1 on the whole of $\mathbf{Z}$, filter out "difficult" objects and build D_2 on them. Continue with the cascade until D_L is built (e.g.,[21]).
(3) *Bagging*: design bootstrap samples by resampling from $\mathbf{Z}$ with a uniform distribution and train one D_i on each sample [10].
(4) *Adaptive resampling*: design bootstrap samples by resampling from $\mathbf{Z}$ with a non-uniform distribution. Update the distribution with respect

to previous successes. Thus, more "difficult" data points will appear more often in the subsequent training samples [4; 10; 19].

Any integration of the four approaches can be applied too. For now, soft computing methods have been used in the context of approaches A,B and C.

15.2.2 *Combination paradigms*

There are generally two types of combinations: **classifier selection** and **classifier fusion** as named in [72]. The presumption in classifier selection is that each classifier is "an expert" in some local area of the feature space. When a feature vector $\mathbf{x} \in \Re^n$ is submitted for classification, the classifier responsible for the vicinity of $\mathbf{x}$ is given the highest credit to label $\mathbf{x}$. We can nominate exactly one classifier to make the decision, as in [60], or more than one "local expert", as in [1; 39; 67]. Classifier fusion assumes that all classifiers are trained over the whole feature space, and are thereby considered as *competitive* rather than *complementary* [57; 73].†

Fusion and selection are often merged. Instead of nominating one "expert" we can nominate a small group of them. We can then take their judgements and weight them by the level of expertise they have on $\mathbf{x}$. Thus, the classifier with the highest individual accuracy could be made the "leading expert" in the team. When many classifiers become involved, the scheme is shifted from classifier selection towards classifier fusion. This suggests that we rarely use the two strategies in their pure forms.

Two major types of multiple classifier outputs are

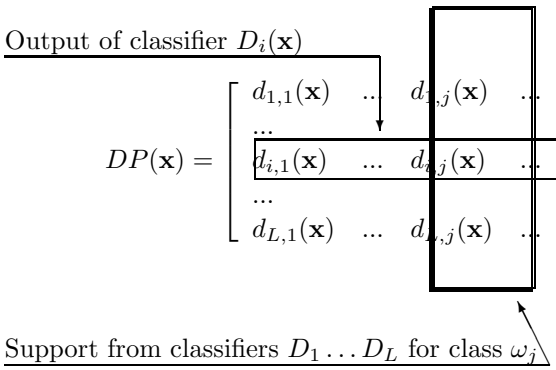
(1) A set of **class labels** (votes), $s_1, \ldots s_L$,

$$D_i(\mathbf{x}) = s_i \in \Omega. \tag{15.4}$$

For example, let $c = 3$, $L = 5$. The output can be $\boxed{\omega_3}\boxed{\omega_2}\boxed{\omega_2}\boxed{\omega_1}\boxed{\omega_2}$.

(2) A matrix of soft labels, called the **decision profile** [52]

†In [62], classifier fusion is named *ensemble approach* and classifier selection is named *modular approach*.



(15.5)

Some fusion methods calculate the support for class ω_j using only the jth column of $DP(\mathbf{x})$, regardless of what the support for the other classes is. Fusion methods that use the DP class-by-class will be called **class-conscious (CC)** [52]. We refer to the alternative group as **class-indifferent (CI)** methods, i.e., methods that use the whole of the decision profile in calculating each $\mu_i(\mathbf{x})$. Notice the difference between the two groups. The former use the *context* of the DP, i.e., recognizing that a column corresponds to a class, but disregard part of the information. Class-indifferent methods use the whole DP but disregard its context.

The diagram in Figure 15.1 depicts one possible grouping of classifier combination methods. The methods are placed in boxes at the leaves of the tree with a few corresponding references. Some of the methods will be described later while others are mentioned only for completeness. Among the class-conscious methods, the weighted linear combination is one of the most popular aggregation formulas. The support for class ω_i is calculated as the weighted average of the supports given by the L classifiers. Based on how the coefficients are obtained, we can distinguish between fixed-coefficient models and data-dependent coefficient models where the coefficients are recalculated for every input $\mathbf{x}$. It is interesting to observe that data-dependent coefficients can be so designed that the combination paradigm (starting off as a classifier fusion model) turns into a classifier selection model. For example, a linear classifier fusion model with data-dependent coefficient so that only one coefficient is 1 and the remaining coefficients are 0s is in fact selecting the classifier corresponding to the 1, to label $\mathbf{x}$.

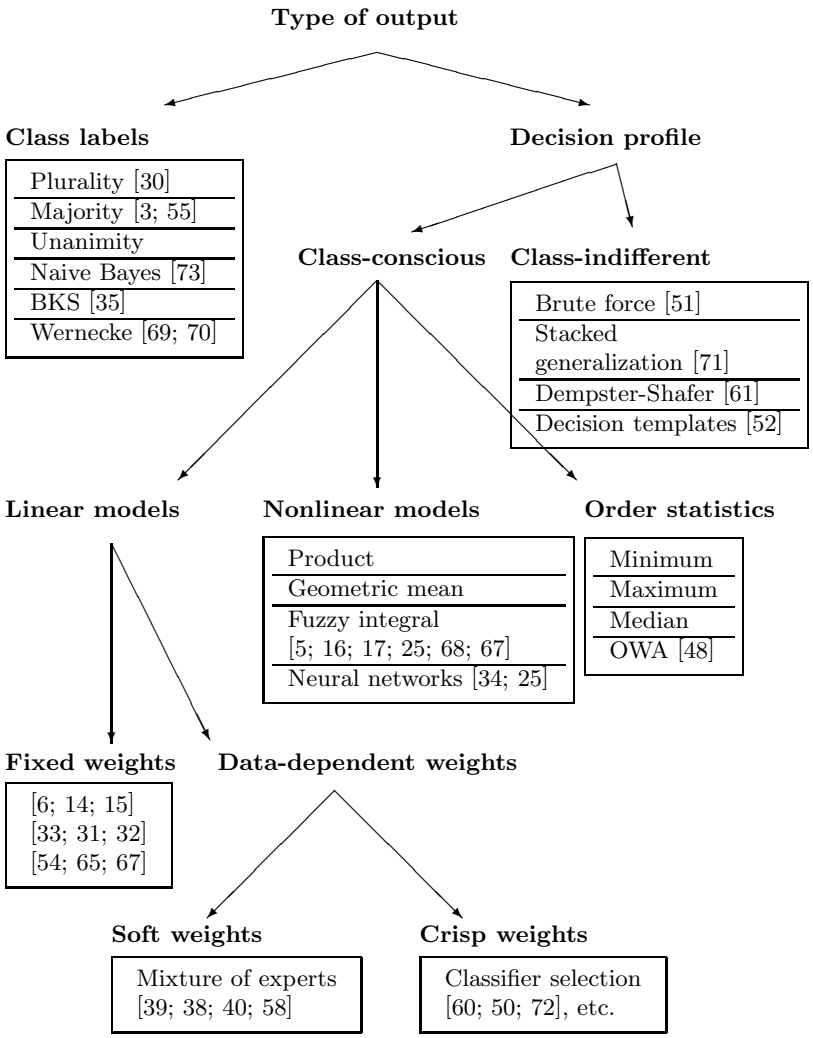


Fig. 15.1 One possible grouping of classifier combination methods

15.3 Soft computing in classifier combination

15.3.1 *Neural networks*

Neural networks (NN) are the most popular choice for the individual classifiers in the team (**approach B**). Most of the studies on combining classifiers

appears in the neural network literature, e.g., the journals *IEEE Transactions on Neural Networks*, *Neural Networks*, *Neural Computation*, *Communication Science*, etc. This choice, initially made by intuition, has now been justified theoretically. The classification error can be decomposed by algebraic manipulation into two terms: bias and variance with respect to individual classifier outputs (see for reference [62]). Ideally, both terms should be small which is hardly possible for a single classifier model. Simple classifiers such as linear discriminant analysis have low variance and high bias. This means that these models are not very sensitive to small changes in the training data set (the calculation of the discriminant functions will not be much affected by small alterations in the training data) but at the same time are unable to reach low error rates. Conversely, neural networks have been shown to be ultimately versatile, i.e., they can approximate any classification boundary with arbitrary precision. The price to pay for the low error rate is that neural classifiers may overtrain. Thus, neural classifiers have low bias (any classification boundary can be modeled) and high variance (small changes in the data set might lead to a very different neural network). Assume that we combine classifiers of the same bias and the same variance V by averaging the classifier outputs, e.g.,

$$\mu_i(\mathbf{x}) = \frac{1}{L} \sum_{k=1,L} d_{k,i}(\mathbf{x}).$$

Then the combination bias will be the same as that of the individual classifiers but the variance can be smaller than V, thereby reducing the error of the combination.

If $\mathcal{D}$ consists of *identical* classifiers, then no improvement will be gained by the combination as the variance of the team estimate will be V. If $\mathcal{D}$ consists of *statistically independent* classifiers, then the combination variance is $\frac{V}{L}$ and the error is subsequently reduced. Even better team can be constituted if the classifiers are *negatively dependent*, i.e., they misclassify different objects. To be able to construct *diverse* classifiers of high accuracy, we need a versatile model. Neural networks are therefore an ideal choice for individual members of the team. The high variance should not be a concern as there are combination mechanisms that will reduce it.

Typically, MLP and Radial Basis Function (RBF) networks are used but variants thereof are also considered [59]. Training of the individual neural classifiers may preceed the design of the combination or be carried out with regard to the team performance. Some authors consider training an excessive

amount of NNs and subsequently selecting the members of $\mathcal{D}$ [23; 24; 26]. If **approach D** is adopted, neural classifiers are trained on the subsequently generated training sets. Drucker [20] compares experimentally neural networks and classification trees (another classifier model found to be very suitable for classifier ensembles) and finds NNs to be superior.

Neural networks can be used as a class-indifferent (brute force or stacked generalization [71]) model and also as a class-conscious model for classifier combination (approach A) [34].

In summary, NNs are undoubtedly the most important intercept between soft computing and classifier combination.

15.3.2 *Evolutionary computation*

Evolutionary computation and mainly *genetic algorithms* (GAs) have been used at different stages of the design of classifier combination systems.

Approach A. Tuning the combiner. Genetic algorithms have been used to find a set of weights for combination through weighted sum [15; 54]. Lam and Suen [54] discuss GAs for finding L weights, one per classifier. Binary encoding of the weights is used with 10-bit representation of each weight. Similarly, Cho [15] uses a GA to find a matrix of $L \times c$ weights, one per classifier-class pair. Thus, $\mu_i(\mathbf{x})$, $i = 1, \ldots, c$ are obtained by c different linear combinations. Each weight is encoded by 8 bits. There are many publications on finding combination weights, both heuristic and more rigorous, e.g., [6; 14; 33; 31; 32; 65; 67]. Whether GAs lead to better results is unknown.

Approach B. Tuning the classifier models. All studies in this category use neural networks as individual members of the team. The neural networks are evolved by GAs, with respect to both weights and structure. A standard GA, although evolving a population of networks, will converge to a *single solution*. That is, the last generation is likely to consist of exact clones or very close relatives, meaning almost identical D_is. Despite highly accurate, these classifiers will hardly form a successful team because nothing can be gained from combining exact replicas of the same classifier. Therefore, a mechanism preserving diversity should be incorporated into the GA. One such option is niching.

Benediktsson et al. [5] apply a real-valued GA to train the network weights and a binary-coded GA for pruning weights off a trained network. The networks are evolved with respect to their individual classification performance, and the diversity of the population is enforced by special genetic operators: extinction

and immigration. Friedrich [23; 24] evolves a population of neural networks is evolved and then selects a subset whose members are maximally negatively correlated. While in [5; 23; 24] a standard MLP is considered, Opitz and Shavlik [59] propose an evolutionary algorithm called ADDEMUP for *knowledge-based* neural networks (KBNN). Each such network can be translated into sets of if-then rules.‡ The GA evolves a population of KBNNs to be the team $\mathcal{D}$. To maintain diversity, the fitness function of chromosome S_i (a single KBNN) is

$$Fitness(S_i) = Accuracy(S_i) + \lambda Diversity(S_i). \tag{15.6}$$

The measure of diversity [44] is generally an estimate of the deviation of the output of the ith KBNN from the average of the team. The more diverse the ensemble, the higher the gain in classification accuracy. The parameter $\lambda > 0$ controls the balance between the two criteria. As a rule of thumb, the authors of [59] recommend to set λ to 0.1 and vary it by about 10 % depending on the current accuracy-diversity dynamic. If the accuracy *of the team* is not decreasing over a number of generations but diversity decreases, then diversity is underemphasized and so, increase λ. If the accuracy starts decreasing and diversity is not decreasing, then diversity is overemphasized and so, decrease λ.

A problem with this group of methods is that the chromosomes correspond to the individual D_i's, and the fitness is not directly related to the overall classification accuracy of the team. It is possible to encode $\mathcal{D}$ as a single chromosome and evolve a population of teams. The search space, however, might become too large and the GA will demand a lot of computing resources and expert effort for tuning.

Approach C. Selecting feature subsets. One of the main use of GAs in pattern recognition has been for selection of a subset of features. The aim is to have a space of dimensionality $t < n$, so that the classifier on $\Re^t$ is no worse than the classifier on $\Re^n$ (using all features). This problem is notoriously difficult and its optimal solution is guaranteed only if all feature subsets are checked (exhaustive enumeration). GAs are a natural option for feature selection [11; 63]. A feature selection GA for multiple classifier systems is proposed in [46]. A population of classifiers is evolved aiming at high individual accuracy. The binary chromosome S_i encodes a feature subset, and the respective classifier D_i is built using only this subset. The team $\mathcal{D}$ is then selected as best group of L from the population. Again, the *group* aspiration criterion is not taken into

‡No fuzzy systems connotation has been given by the authors.

account when the individual chromosomes (classifiers) are evaluated by their fitness. The diversity preserving adjustment in this model is that the best team is identified at each generation, and the chromosomes in it are retained for the next generation, regardless of their individual fitness. To overcome the "individualistic" approach, the whole team $\mathcal{D}$ is evolved in [53]. Two GA versions are proposed: Version 1, where D_is use disjoint subsets of features and Version 2, where the subsets of features may overlap. In Version 1, the chromosome has n genes, one for each feature. The values of each gene are in the set $\{0, 1, \ldots, L\}$. A value $i \in \{1, \ldots, L\}$ at position j means that (only) D_i uses feature x_j, and a value 0 means that feature x_j is not used by any classifier in this team.

Example 15.1 Let $n = 10$, and $L = 3$. A possible chromosome is $\boxed{2}\boxed{2}\boxed{1}\boxed{2}\boxed{2}\boxed{3}\boxed{2}\boxed{3}\boxed{0}\boxed{1}$. This chromosome represents a team $\mathcal{D}$ where D_1 uses a 2-dimensional feature vector $\mathbf{x} = [x_3, x_{10}]^T$, D_2 uses a 5-dimensional feature vector $\mathbf{x} = [x_1, x_2, x_4, x_5, x_7]^T$, D_3 uses a 2-dimensional feature vector $\mathbf{x} = [x_6, x_8]^T$, and feature x_9 is not used.

Version 2 GA allows for 2^L values of each gene, accounting for all possible combinations of D_i (or none) that might share feature x_j.

There is an apparent analogy between the problem of evolving one member of the team and the whole team on the one hand, and the Michigan and Pittsburgh approaches for evolving fuzzy if-then systems on the other hand [51]. Within the Michigan approach, the chromosome represents one if-then rule, whereas within the Pittsburgh approach, the chromosome represents the whole fuzzy if-then system. The preferences in the literature are not clear-cut, so both approaches are used.

Approach D. Selecting training sets. The reason why this most promising approach has not been explored so far could be that if the data set $\mathbf{Z}$ is large, the same will be the chromosome, and the GA will be unacceptably slow. Knowing the advantages of approach D, subset selection by GAs seems worth trying (see [47; 49]).

15.3.3 *Fuzzy sets*

Fuzzy set theory has been used predominantly at the combination stage (**approach A**). Detailed below are several fuzzy combination schemes (cf. [51]).

15.3.3.1 *Simple fuzzy aggregation connectives*

These combination designs belong to the *class-conscious* group because each $\mu_i(\mathbf{x})$ is calculated using only the ith column of the decision profile $DP(\mathbf{x})$. We use the L-place operators *minimum, maximum, average* and *product* as the function $\mathcal{F}$ in

$$\mu_i(\mathbf{x}) = \mathcal{F}\ (d_{1,i}(\mathbf{x}), \ldots, d_{L,i}(\mathbf{x}))\,, \quad i = 1, \ldots, c. \tag{15.7}$$

Example 15.2 Let $c = 3$ and $L = 5$. Assume that for a certain $\mathbf{x}$,

$$DP(\mathbf{x}) = \begin{bmatrix} 0.1 & 0.5 & 0.4 \\ 0.0 & 0.0 & 1.0 \\ 0.4 & 0.3 & 0.4 \\ 0.2 & 0.7 & 0.1 \\ 0.1 & 0.8 & 0.2 \end{bmatrix}.$$

Applying each of the operators columnwise, we obtain as the final soft class labels

$$\begin{array}{lcl} \text{Minimum} & = & [\,0.0,\ 0.0,\ 0.1\,]^T; \\ \text{Maximum} & = & [\,0.4,\ 0.8,\ 1.0\,]^T; \\ \text{Average} & = & [\,0.16,\ 0.46,\ 0.42\,]^T; \\ \text{Product} & = & [\,0.0,\ 0.0,\ 0.0032\,]^T. \end{array}$$

If hardened, minimum, maximum, and product will label $\mathbf{x}$ in class ω_3, whereas the average will put $\mathbf{x}$ in class ω_2.

15.3.3.2 *More sophisticated aggregation connectives*

Many such aggregation operations are available in the fuzzy set literature [9]. Ordered Weighted Averaging (OWA) operators can also be applied as $\mathcal{F}$ [48]. The OWA coefficients are not associated with a particular classifier D_i but with the *places* in the ordered outputs. The operation of OWA combination is shown in Figure 15.2

OWA prevents crediting one particular "expert" with the highest competence across $\Re^n$, as it would be the case if we assigned fixed weights to the classifiers. If the favourite expert (classifier) has received the credit because of overfitting the training data, then by praising it, we can face poor generalization. Thus, classifier fusion by OWA seems more robust than the weighted average, where the coefficients are derived on the basis of classifier performance. It is worth noticing that the fuzzy integral for classifier fusion takes this idea

OWA operators for combining classifiers,,

(1) Pick L OWA coefficients such that

$$\mathbf{b} = [b_1, \ldots, b_L]^T, \quad \sum_{i=1}^{L} b_i = 1.$$

(2) For $k = 1, \ldots, c$,

(a) Sort $d_{i,k}(\mathbf{x}), i = 1, \ldots, L$ in descending order, so that

$$a_1 = \max_i d_{i,k}(\mathbf{x}), \quad \text{and} \quad a_L = \min_i d_{i,k}(\mathbf{x}).$$

(b) Calculate the support for class ω_k

$$\mu_k(\mathbf{x}) = \sum_{i=1}^{L} b_i a_i.$$

Fig. 15.2 OWA operators for combining classifiers

further so that OWA aggregation is a special case of it. OWA can model various operations as shown in Table 15.3.

We can either pick the set of OWA coefficients or calculate them from $\mathbf{Z}$ by minimizing the classification error of $\mathcal{D}$.

Verikas et al. [67] consider aggregation by Zimmermann and Zysno's compensatory operator

$$\mu_i(\mathbf{x}) = \left(\prod_{k=1}^{L} [d_{k,i}(\mathbf{x})]^{w_k} \right)^{1-\gamma} \left(1 - \prod_{k=1}^{L} [1 - d_{k,i}(\mathbf{x})]^{w_k} \right)^{\gamma}, \tag{15.8}$$

where $w_k, k = 1, \ldots, L$ are coefficients of global "competence" (across the whole $\Re^n$), $\sum_{k=1}^{L} w_k = L$, and $\gamma \in [0, 1]$ is the *compensation parameter*. Verikas et al. [67] propose also aggregation by BADD defuzzification[§]

$$\mu_i(\mathbf{x}) = \frac{\sum_{k=1}^{L} d_{k,i}(\mathbf{x})[w_k(\mathbf{x})]^{\delta}}{\sum_{k=1}^{L} [w_k(\mathbf{x})]^{\delta}}, \quad i = 1, \ldots, c, \tag{15.9}$$

[§]although used in a slightly different context

Table 15.3 Special cases of OWA operators

Minimum	$[0, 0, \ldots, 1]^T$,
Maximum	$[1, 0, \ldots, 0]^T$,
Median	$[\underbrace{0, \ldots 0}_{\frac{L-1}{2}}, 1, \underbrace{0, \ldots 0}_{\frac{L-1}{2}}]^T$, for odd L, $[\underbrace{0, \ldots 0}_{\frac{L-2}{2}}, \frac{1}{2}, \frac{1}{2}, \underbrace{0, \ldots 0}_{\frac{L-2}{2}}]^T$, for even L,
Average	$[\frac{1}{L}, \ldots, \frac{1}{L}]^T$,
Competition jury	$[0, \frac{1}{L-2}, \ldots, \frac{1}{L-2}, 0]^T$.

where δ is a parameter, and $w_k(\mathbf{x})$ are data-dependent weights calculated to express the "expertise" of classifier D_k for the input $\mathbf{x}$.

15.3.3.3 *Fuzzy integral*

Fuzzy integral can also be used as an aggregation connective [27; 29] and has been applied to classifier combination [5; 16; 17; 25; 68; 67].

We use a fuzzy measure to take into account the *importance of any subset* of classifiers from $\mathcal{D}$ with respect to a given ω_i. Let $\mathcal{P}(\mathcal{D})$ be the power set of $\mathcal{D}$. A *fuzzy measure* on $\mathcal{D}$ is the set function

$$g : \mathcal{P}(\mathcal{D}) \to [0, 1], \tag{15.10}$$

such that

(1) $g(\emptyset) = 0$, $g(\mathcal{D}) = 1$;
(2) For any A and B, subsets of $\mathcal{D}$, $A \subset B \Rightarrow g(A) \leq g(B)$.

g is called a λ-*fuzzy measure* if for any A and B, subsets of $\mathcal{D}$, such that $A \cap B = \emptyset$,

$$g(A \cup B) = g(A) + g(B) + \lambda g(A)g(B), \quad \lambda \in (-1, \infty). \tag{15.11}$$

Two basic types of fuzzy integrals have been proposed: Sugeno type and Choquet type. Let H be a fuzzy set on $\mathcal{D}$. The *Sugeno fuzzy integral* with respect to a fuzzy measure g is obtained by

$$\mathcal{A}_g^{FI} = \max_{\alpha}\{\min(\alpha, g(H_\alpha))\}, \tag{15.12}$$

where H_α is the α-cut of H.

Example 15.3 Let $L = 3$, and let the fuzzy measure g be defined as follows

Subset	D_1	D_2	D_3	D_1, D_2	D_1, D_3	D_2, D_3	D_1, D_2, D_3
g	0.3	0.1	0.4	0.4	0.5	0.8	1

Let $H = [0.1, 0.7, 0.5]^T$ be a fuzzy set on $\mathcal{D}$ accounting for the support for class ω_i by D_1, D_2, and D_3, respectively (the ith row of $DP(\mathbf{x})$). The α-cuts of H are

$$\begin{array}{lllll} \alpha & = & 0, & H_0 & = & \{D_1, D_2, D_3\}; \\ \alpha & = & 0.1, & H_{0.1} & = & \{D_1, D_2, D_3\}; \\ \alpha & = & 0.5, & H_{0.5} & = & \{D_2, D_3\}; \\ \alpha & = & 0.7, & H_{0.7} & = & \{D_2\}; \\ \alpha & = & 1, & H_0 & = & \emptyset. \end{array}$$

Then

$$\begin{array}{rcl} \mu_i(\mathbf{x}) & = & \mathcal{A}_g^{FI} \\ & = & \max\{\min(0,1), \min(0.1,1), \min(0.5,0.8), \\ & & \min(0.7,0.1), \min(1,0)\} \\ & = & \max\{0, 0.1, 0.5, 0.1, 0\} \\ & = & 0.5. \end{array} \tag{15.13}$$

The fuzzy measure g can be calculated from a set of L values g^j, called *fuzzy densities*, representing the individual importance of $D_1, \ldots, D_L$, respectively. We can find a λ-fuzzy measure which is consistent with these densities. The value of λ is obtained as the unique real root greater than -1 of the polynomial

$$\lambda + 1 = \prod_{j=1}^{L}(1 + \lambda g^j), \quad \lambda \neq 0. \tag{15.14}$$

The operation of fuzzy integral as a classifier combiner is shown in Figure 15.3.

The support for ω_i, $\mu_i(\mathbf{x})$, can be thought of as a "compromise" between the *competence* (represented by the fuzzy measure g) and the *evidence* (represented by the i-h row of the decision profile $DP(\mathbf{x})$. Notice that the fuzzy measure

Fuzzy integral for classifier fusion

(1) Fix the L *fuzzy densities* $g^1, \ldots, g^L$, e.g., by setting g^j to the estimated probability of correct classification of D_j.
(2) Calculate $\lambda > -1$ from (15.14).
(3) For a given $\mathbf{x}$ sort the kth column of $DP(\mathbf{x})$ to obtain $[d_{i_1,k}(\mathbf{x}), d_{i_2,k}(\mathbf{x}), \ldots, d_{i_L,k}(\mathbf{x})]^T$, $d_{i_1,k}(\mathbf{x})$ being the highest degree of support, and $d_{i_L,k}(\mathbf{x})$, the lowest.
(4) Sort the fuzzy densities correspondingly, i.e., $g^{i_1}, \ldots, g^{i_L}$ and set $g(1) = g^{i_1}$.
(5) For $t = 2$ to L, calculate recursively

$$g(t) = g^{i_t} + g(t-1) + \lambda g^{i_t} g(t-1).$$

(6) Calculate the final degree of support for class ω_k by

$$\mu_k(\mathbf{x}) = \max_{t=1}^{L} \left\{ \min\{d_{i_t,k}(\mathbf{x}), g(t)\} \right\}.$$

Fig. 15.3 Fuzzy integral for classifier fusion

vector $[g(1), \ldots, g(L)]^T$ might be different for each class, and is also specific for the current $\mathbf{x}$. Two fuzzy measure vectors will be the same only if the ordering of the classifier support is the same. The algorithm in Figure 15.3 calculates a Sugeno fuzzy integral. For the Choquet fuzzy integral with the same λ-fuzzy measure, the last formula should be replaced by

$$\mu_k(\mathbf{x}) = d_{i_1,k}(\mathbf{x}) + \sum_{j=2}^{L} \left(d_{i_{j-1},k}(\mathbf{x}) - d_{i_j,k}(\mathbf{x}) \right) g(j-1).$$

15.3.3.4 *Decision templates*

The idea of the decision templates model is to "remember" the most typical decision profile for each class, called the *decision template*, DT_i, for that class, and then compare it with the current decision profile $DP(\mathbf{x})$. The closest match will label $\mathbf{x}$. Figure 15.4 describes the training and Figure 15.5, the operation of the decision templates model. The similarity between DT_i, $i = 1, \ldots, c$ on the one hand, and $DP(\mathbf{x})$ on the other hand, is calculated through Euclidean distance between the two.

Decision templates (training)

(1) For $i = 1, \ldots, c$, calculate the mean of the decision profiles $DP(\mathbf{z}_j)$ of all members of ω_i from the data set $\mathbf{Z}$. Call the mean *a decision template* DT_i

$$DT_i = \frac{1}{N_i} \sum_{\substack{\mathbf{z}_j \in \omega_i \\ \mathbf{z}_j \in \mathbf{Z}}} DP(\mathbf{z}_j), \tag{15.15}$$

where N_i is the number of elements of $\mathbf{Z}$ from ω_i.

(2) Return $DT_1, \ldots, DT_c$.

Fig. 15.4 Training of the Decision templates method

Decision templates (operation)

(1) Given the input $\mathbf{x} \in \Re^n$, construct $DP(\mathbf{x})$ as in (15.5).

(2) Calculate the squared Euclidean distance between $DP(\mathbf{x})$ and each DT_i, $i = 1, \ldots, c$

$$d_E(DP(\mathbf{x}), DT_i) = \sum_{j=1}^{c} \sum_{k=1}^{L} (d_{k,j}(\mathbf{x}) - dt_i(k,j))^2, \tag{15.16}$$

where $dt_i(k,j)$ is the k,j-th entry in decision template DT_i (an $L \times c$ matrix).

(3) Calculate the components of the soft label of $\mathbf{x}$ by

$$\mu_i(\mathbf{x}) = 1 - \frac{1}{L \cdot c} d_E(DP(\mathbf{x}), DT_i). \tag{15.17}$$

Fig. 15.5 Operation of the Decision templates method

Example 15.4 Let $c = 3$ and $L = 2$, and

$$DT_1 = \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \\ 0.5 & 0.5 \end{bmatrix} \quad \text{and} \quad DT_2 = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \\ 0.1 & 0.9 \end{bmatrix}. \tag{15.18}$$

Assume that for an input $\mathbf{x}$, the following decision profile has been obtained

$$DP(\mathbf{x}) = \begin{bmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.5 & 0.5 \end{bmatrix}. \tag{15.19}$$

Then the soft label of $\mathbf{x}$ is

$$\mu_1(\mathbf{x}) = 0.96, \;\; \mu_2(\mathbf{x}) = 0.93. \tag{15.20}$$

As both $DP(\mathbf{x})$ and DT_i are fuzzy sets on $\mathcal{D} \times \Omega$, any measure of similarity between fuzzy sets can be used.

Ishibuchi et al. [36; 37] propose voting schemes over a set of fuzzy if-then rules or systems of fuzzy if-then rules. Lu and Yamaoka [56] apply fuzzy inference to design the combiner.

Fuzzy set theory offers a great choice of combination ideas but these have not been explored in conjunction with approach **D**. Many authors argue that the combination scheme is not as relevant for the final accuracy as is the diversity of the classifiers. Knowing that fuzzy combinations are capable of improving the accuracy beyond that of the majority vote or the averaging model, integration of fuzzy combination with approach **D** seems very promising.

15.4 Conclusions

This chapter explains classifier combination and some soft computing paradigms within it. The three components attributed to the term "soft computing": neural networks, evolutionary computing and fuzzy sets are considered separately. The purpose was to explain the techniques and methods used, not to advocate a particular example. Brief comments on each soft computing component are offered in the respective subsections. It goes without saying that all soft computing methods cited in this study have been compared experimentally against some rival methods and have been found to be better. However, since there is no accepted standard or "ultimate test", the superiority of some techniques over others cannot be empirically proven. This is the beauty and the curse of the heuristic methods such as these explained here, and the final choice is left to the experience and the intuition of the designer. Sorting and grouping the methods, as done here, might help with the choice, and so might the somewhat contradictory guideline: keep it *simple* and *accurate*.

Bibliography

[1] E. Alpaydin and M. I. Jordan. Local linear perceptrons for classification. *IEEE Transactions on Neural Networks*, 7(3):788–792, 1996.

[2] Y. L. Barabash. *Collective Statistical Decisions in Recognition*. Radio i Sviaz', Moscow, 1983. (In Russian).

[3] R. Battiti and A.M. Colla. Democracy in neural nets: Voting schemes for classification. *Neural Networks*, 7:691–707, 1994.

[4] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–142, 1999.

[5] 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 Analysis, theory, Methods & Applications*, 30(3):1323–1334, 1997.

[6] J.A. Benediktsson and P.H. Swain. Consensus theoretic classification methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:688–704, 1992.

[7] J.C. Bezdek, J.M Keller, R. Krishnapuram, and N.R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer Academic Publishers, 1999.

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

[9] I. Bloch. Information combination operators for data fusion: a comparative review with classification. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 26:52–67, 1996.

[10] L. Breiman. Combining predictors. In A.J.C. Sharkey, editor, *Combining Artificial Neural Nets*, pages 31–50. Springer-Verlag, London, 1999.

[11] E.I. Chang and R.P. Lippmann. Using genetic algorithms to improve pattern classification performance. volume 3 of *Neural Information Processing Systems*, pages 797–803, San Mateo, CA, 1991. Morgan Kaufmann Publishers.

[12] K. Chen, L. Wang, and H. Chi. Methods of combining multiple classifiers with different features and their applications to text-independent speaker identification. *International Journal on Pattern Recognition and Artificial Intelligence*, 11(3):417–445, 1997.

[13] C.-C. Chiang and H.-C. Fu. A divide-and-conquer methodology for modular supervised neural network design. In *IEEE International Conference on Neural Networks*, pages 119–124, Orlando, Florida, 1994.

[14] C.C. Chibelushi, F. Deravi, and J.S.D. Mason. Adaptive classifier integration for robust pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B: Cybernetics, 29(6):902–907, 1999.

[15] S.-B. Cho. Pattern recognition with neural networks combined by genetic algorithm. *Fuzzy Sets and Systems*, 103:339–347, 1999.

[16] S.-B. Cho and J.H. Kim. Combining multiple neural networks by fuzzy integral and robust classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 25:380–384, 1995.

[17] S.B. Cho and J.H. Kim. Multiple network fusion using fuzzy logic. *IEEE Transactions on Neural Networks*, 6:497–501, 1995.

[18] B.V. Dasarathy and B.V. Sheela. A composite classifier system design: concepts and methodology. *Proceedings of IEEE*, 67:708–713, 1978.

[19] T.G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15, Cagliari, Italy, 2000. Springer.

[20] H. Drucker. Boosting using neural networks. In A.J.C. Sharkey, editor, *Combining Artificial Neural Nets*, pages 51–78. Springer-Verlag, London, 1999.

[21] H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6:1289–1301, 1994.

[22] E. Filippi, M. Costa, and E. Pasero. Multy-layer perceptron ensembles for increased performance and fault-tolerance in pattern recognition tasks. In *IEEE International Conference on Neural Networks*, pages 2901–2906, Orlando, Florida, 1994.

[23] C.M. Friedrich. Ensembles of evolutionary created artificial neural networks. In *Proc. 5th Int. Workshop Fuzzy-Neuro Systems'98 (FNS'98)*, pages 250–256, Munich, Germany, 1998.

[24] C.M. Friedrich. Ensembles of evolutionary created artificial neural networks and nearest neighbour classifiers. In *Proc. 3rd On-line Conference on Soft Computing in Engineering Design and Manufacturing (WSC3)*, pages 288–298, 1998.

[25] P.D. Gader, M.A. Mohamed, and J.M. Keller. Fusion of handwritten word classifiers. *Pattern Recognition Letters*, 17:577–584, 1996.

[26] G. Giacinto and F. Roli. Design of effective neural network ensembles for image classification processes. *Image Vision and Computing Journal*, 19(9-10):699–707, 2001.

[27] M. Grabisch. On equivalence classes of fuzzy connectives - the case of fuzzy integrals. *IEEE Transactions on Fuzzy Systems*, 3(1):96–109, 1995.

[28] M. Grabisch and F. Dispot. A comparison of some for fuzzy classification on real data. In *2nd International Conference on Fuzzy Logic and Neural Networks*, pages 659–662, Iizuka, Japan, 1992.

[29] M. Grabisch and M. Sugeno. Multi-attribute classification using fuzzy integral. In *IEEE International Conference on Fuzzy Systems*, pages 47–54, San Diego, California, 1992.

[30] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[31] S. Hashem. Optimal linear combinations of neural networks. *Neural Networks*, 10(4):599–614, 1997.

[32] S. Hashem. Treating harmful collinearity in neural network ensembles. In A.J.C. Sharkey, editor, *Combining Artificial Neural Nets*, pages 101–125. Springer-Verlag, London, 1999.

[33] S. Hashem, B. Schmeiser, and Y. Yih. Optimal linear combinations of neural networks: an overview. In *IEEE International Conference on Neural Networks*, pages 1507–1512, Orlando, Florida, 1994.

[34] Y.S. Huang and C.Y. Suen. A method of combining multiple classifiers - a neural network approach. In *12th International Conference on Pattern Recognition*, pages 473–475, Jerusalem, Israel, 1994.

[35] Y.S. Huang and C.Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:90–93, 1995.

[36] H. Ishibuchi, T. Morisawa, and T. Nakashima. Voting schemes for fuzzy rule-based classification systems. In *Proc. FUZZ/IEEE*, 1996.

[37] H. Ishibuchi, T. Nakashima, and T. Morisawa. Voting in fuzzy rule-based systems for pattern classification problems. *Fuzzy Sets and Systems*, 103:223–238, 1999.

[38] R.A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888, 1995.

[39] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[40] M.I. Jordan and L. Xu. Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8:1409–1431, 1995.

[41] J.M. Keller, P. Gader, H. Tahani, J.-H. Chiang, and M. Mohamed. Advances in fuzzy integration for pattern recognition. *Fuzzy Sets and Systems*, 65:273–283, 1994.

[42] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[43] J. Kittler, A. Hojjatoleslami, and T. Windeatt. Strategies for combining classifiers employing shared and distinct representations. *Pattern Recognition Letters*, 18:1373–1377, 1997.

[44] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation and active learning. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, Cambridge, MA, 1995.

[45] L.I. Kuncheva. Change-glasses approach in pattern recognition. *Pattern Recognition Letters*, 14:619–623, 1993.

[46] L.I. Kuncheva. Genetic algorithm for feature selection for parallel classifiers. *Information Processing Letters*, 46:163–168, 1993.

[47] L.I. Kuncheva. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters*, 16:809–814, 1995.

[48] L.I. Kuncheva. An application of OWA operators to the aggregation of multiple classification decisions. In R.R. Yager and J. Kacprzyk, editors, *The Ordered Weighted Averaging operators. Theory and Applications*, pages 330–343. Kluwer Academic Publishers, USA, 1997.

[49] L.I. Kuncheva. Fitness functions in editing k-nn reference set by genetic algorithms. *Pattern Recognition*, 30:1041–1049, 1997.

[50] L.I. Kuncheva. Clustering-and-selection model for classifier combination. In *Proc. Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, pages 185–188, Brighton, UK, 2000.

[51] L.I. Kuncheva. *Fuzzy Classifier Design*. Studies in Fuzziness and Soft Computing. Springer Verlag, Heidelberg, 2000.

[52] L.I. Kuncheva, J.C. Bezdek, and R.P.W. Duin. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern Recognition*, 34(2):299–314, 2001.

[53] L.I Kuncheva and L.C. Jain. Designing classifier fusion systems by genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(4):327–336, 2000.

[54] L. Lam and C.Y. Suen. Optimal combination of pattern classifiers. *Pattern Recognition Letters*, 16:945–954, 1995.

[55] L. Lam and C.Y. Suen. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics*, 27(5):553–568, 1997.

[56] Y. Lu and F. Yamaoka. Fuzzy integration of classification results. *Pattern Recognition*, 30(11):1877–1891, 1997.

[57] K.-C. Ng and B. Abramson. Consensus diagnosis: A simulation study. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:916–928, 1992.

[58] S.J. Nowlan and G.E. Hinton. Evaluation of adaptive mixtures of competing experts. In R.P. Lippmann, J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 774–780, 1991.

[59] D. Opitz and J. Shavlik. A genetic algorithm approach for creating neural network ensembles. In A.J.C. Sharkey, editor, *Combining Artificial Neural Nets*, pages 79–99. Springer-Verlag, London, 1999.

[60] L.A. Rastrigin and R.H. Erenstein. *Method of Collective Recognition*. Energoizdat, Moscow, 1981. (In Russian).

[61] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7:777–781, 1994.

[62] A.J.C. Sharkey, editor. *Combining Artificial Neural Nets. Ensemble and Modular Multi-Net Systems*. Springer-Verlag, London, 1999.

[63] W. Siedlecki and J. Sklansky. A note on genetic algorithms for large-scale feature selection. *Pattern Recognition Letters*, 10:335–347, 1989.

[64] F. Smieja. The pandemonium system of reflective agents. *IEEE Transactions on Neural Networks*, 7:97–106, 1996.

[65] V. Tresp and M. Taniguchi. Combining estimators using non-constant weighting functions. In G. Tesauro, D.S. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, Cambridge, MA, 1995. MIT Press.

[66] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3/4):385–404, 1996.

[67] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis. Soft combination of neural classifiers: A comparative study. *Pattern Recognition Letters*, 20:429–444, 1999.

[68] D. Wang, J. M. Keller, C.A. Carson, K.K. McAdoo-Edwards, and C.W. Bailey. Use of fuzzy-logic-inspired features to improve bacterial recognition through classifier fusion. *IEEE Transactions on Systems, Man, and Cybernetics*, 28B(4):583–591, 1998.

[69] K.-D. Wernecke. On classification strategies in medical diagnostics (with special preference to mixed models). In H.H. Bock, editor, *Classification and Related Methods of Data Analysis*, pages 299–306. Elsevier Science Publisher, 1988.

[70] K.-D. Wernecke. A coupling procedure for discrimination of mixed data. *Biometrics*, 48:497–506, 1992.

[71] D.H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–260, 1992.

[72] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:405–410, 1997.

[73] L. Xu, A. Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:418–435, 1992.