Switching Between Selection and Fusion in Combining Classifiers: An Experiment

Ludmila I. Kuncheva, *Member, IEEE*

***Abstract*—This paper presents a combination of classifier selection and fusion by using statistical inference to switch between the two. Selection is applied in those regions of the feature space where one classifier strongly dominates the others from the pool [called clustering-and-selection or (CS)] and fusion is applied in the remaining regions. Decision templates (DT) method is adopted for the classifier fusion part. The proposed combination scheme (called CS+DT) is compared experimentally against its two components, and also against majority vote, naive Bayes, two joint-distribution methods (BKS and a variant due to Wernecke), the dynamic classifier selection (DCS) algorithm DCS_LA based on local accuracy (Woods *et al.*), and simple fusion methods such as maximum, minimum, average, and product. Based on the results with five data sets with homogeneous ensembles [multilayer perceptrons (MLPs)] and ensembles of different classifiers, we offer a discussion on when to combine classifiers and how classifier selection (static or dynamic) can be misled by the differences in the classifier team.**

***Index Terms*—Classifier combination, classifier selection and fusion, confidence intervals (CIs), decision templates (DTs), discriminant analysis, multiple classifier systems, pattern recognition, supervised learning.**

I. Introduction

CLASSIFIER combination is a viable alternative to using a single classifier. This is now an established research area thriving mostly on heuristic solutions. Some theoretical results are also available but only for special cases, usually assuming independent classifier outputs. *Ad hoc* methods, such as the one proposed in this paper, could be useful as a pre-phase toward a more general theory of classifier combination.

In this paper, we assume that a small set of trained classifiers is available and we are interested in combining their outputs aiming at the highest possible accuracy.

Let $\mathcal{D} = \{D_1, D_2, \ldots, D_L\}$ be a set of classifiers and $\Omega = \{\omega_1, \ldots, \omega_c\}$ be a set of class labels. Each classifier gets as its input a feature vector $\mathbf{x} \in \Re^n$ and assigns it to a class label from Ω, i.e., D_i: $\Re^n \rightarrow \Omega$, or equivalently, $D_i(\mathbf{x}) \in \Omega$, $i = 1, \ldots c$. In many cases, the classifier output is a c-dimensional vector with supports to the classes, i.e.,

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

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 [3]). 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 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, we can find a vector with c final degrees of support for the classes as a soft label for $\mathbf{x}$, denoted

$$D(\mathbf{x}) = [\mu_1(\mathbf{x}), \ldots, \mu_c(\mathbf{x})]^T. \tag{2}$$

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

$$\mu_s(\mathbf{x}) \geq \mu_t(\mathbf{x}), \quad \forall t = 1, \ldots, c. \tag{3}$$

Ties are resolved arbitrarily. The minimum-error classifier is recovered from (3) when $\mu_i(\mathbf{x}) = P(\omega_i|\mathbf{x})$.

Two strategies are discussed in the literature on classifier combination: classifier *selection* and classifier *fusion*. The presumption in classifier selection is that each classifier has expertise 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 authority to label $\mathbf{x}$. Classifier fusion assumes that all classifiers are equally "experienced" in the whole feature space and the decisions of all of $\mathcal{D}$ are taken into account for any $\mathbf{x}$. There are many combination models halfway between these two extremes, e.g., where individual competence varies over $\Re^n$.

Contrary to this implicit integration of fusion and selection, in this spaper we propose a scheme using statistical inference to switch between the two. Section II introduces the probabilistic background of classifier selection and explains the clustering-and-selection (CS) model used in the proposed combination. Classifier fusion and the decision templates model (DT) are presented in Section III. Other fusion models used in the experiments for this paper are detailed in the Appendix. Section IV gives the background of using confidence intervals (CIs) in selecting a classifier and the model that results from this, called CS+DT. Section V contains experiments with five datasets, and Section VI offers a conclusion.

II. Classifier Selection

A. Probabilistic Background

In classifier selection, when a feature vector $\mathbf{x} \in \Re^n$ is submitted for classification, the classifier responsible for the vicinity of $\mathbf{x}$ is chosen to decide on the class label. We can nominate exactly one classifier to make the decision, as in [18], or more than one "local expert," as in [2] and [9]. Classifier selection has been proposed in the form of a composite classifier system by Dasarathy and Sheela [4]. They combine

Manuscript received February 23, 2001; revised July 16, 2001 and November 6, 2001. This paper was recommended by Associate Editor D. Goldgof.

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

Publisher Item Identifier S 1083-4419(02)00697-0.

a linear classifier and a k-nearest neighbor (k-nn) classifier. The authors suggest to identify a conflict domain in the feature space and use k-nn in that domain while using the linear classifier elsewhere.

Two types of classifier selection systems can be distinguished.

1) **Static classifier selection.** Selection regions are specified during a training phase, prior to classifying the unlabeled vector $\mathbf{x}$. In the operation phase, the region of $\mathbf{x}$ is first found, e.g., R_j, and processed further by the respective classifier $D_{i(j)}$, responsible for region R_j. Two possible training approaches are: specify the regions and then assign a responsible classifier for each region (e.g., the model in [21]), or given $\mathcal{D}$, find a region (possibly a set of regions) where each classifier is the best one (e.g., the model in [18]). Although probably more efficient, the second approach is difficult to implement.
2) **Dynamic classifier selection (DCS).** The choice of a classifier to label $\mathbf{x}$ is made during the operation phase. This choice is typically based on the certainty of the current decision. Preference is given to more certain classifiers. For example, if the 5-nn rule is being used in a two-class problem, and three of the neighbors vote for class ω_1 and two for class ω_2, we can switch to, say, 3-nn or 1-nn, thereby changing the classification rule *dynamically* [12]. Rastrigin and Erenstein [18] proposed the following dynamic selection scheme. The "competence" of each classifier is estimated *in the vicinity of* $\mathbf{x}$ as the classifier's accuracy. Two methods were suggested for this: the potential functions method and the k-nearest neighbors. The classifier with the highest competence is authorized to label $\mathbf{x}$. Thus, the regions R_j are estimated during the classification process. Woods *et al.* [26] also use local analysis of competence to nominate a classifier from $\mathcal{D}$ to label $\mathbf{x}$. We took as our DCS model their algorithm called *dynamic classifier selection with local accuracy* (DCS_LA, we refer to it in the sequel as DCS). From the two versions of DCS_LA in [26], the "local class accuracy" version has been found to be the better one, so this is what we adopted as well. The algorithm that we used is shown in Fig. 1. It has a minor difference with the original algorithm in that we fixed the depth of the tie breaking procedure. We kept track of the randomly broken ties to make sure that the overall performance of the algorithm is not affected. The dynamic selection idea mimics the decision making in real life situations, e.g., in medical diagnostics, where help is sought if the confidence of the current decision-maker is not high enough.

Why should classifier selection work? Let $\Re^n$ be divided into K regions of competence, $K > 1$. Denote the regions by $R_1, \ldots, R_K$. These regions are not associated with specific classes, nor do they need to be of a certain shape or size.

An example of partitioning into regions is shown in Fig. 2. Depicted is a 15-point dataset in $\Re^2$ with two class labels: squares and snowflakes. The *two classification regions* defined by the nearest neighbor classifier are overlayed using Voronoi diagrams. Shaded is the classification region for

DCS_LA WITH LOCAL CLASS ACCURACY

1. Design the individual classifiers $D_1, \ldots, D_L$ using the labeled data set $\mathbf{Z}$. Pick the value of the parameter K (recommended is $K = 10$).
2. Upon obtaining an input $\mathbf{x}$, label it by $D_1, \ldots, D_L$. If all classifiers agree on the label, then assign this label to $\mathbf{x}$ and return.
3. If there is a disagreement, then estimate the local accuracy for each D_i, $i = 1, \ldots, L$. To do this, take the class label offered for $\mathbf{x}$ by D_i, say $s \in \Omega$, and find the K points closest to $\mathbf{x}$ for which D_i has issued the same label. Calculate the proportion of the points whose true labels were s to be an estimate of the local accuracy of D_i with respect to class s.
4. If there is a unique winner of the local accuracy contest, let it label $\mathbf{x}$ and return. Otherwise, check if the tied winners have the same labels for $\mathbf{x}$. In this case, accept the label and return. If a unique class label could be selected by plurality among the tied classifiers, then assign this label to $\mathbf{x}$ and return.
5. Otherwise, there is a class label tie between the most locally competent classifiers. The classifier with the next highest local competence is identified to break the tie. If all classifiers are tied (there is no classifier left to break the tie) and the class labels are still tied, then pick a random class label among the tied *labels* and return. If there is a unique winner of the (second) local competence contest, and it can resolve the tie, then use the winning label for $\mathbf{x}$ end return.
6. If none of the clauses in the previous point apply, break the class label tie randomly and return a label for $\mathbf{x}$. (Further analysis would have produced an overcomplicated code because there could be another tie on the second highest competence, where the tied classifiers disagree on the class label, etc.)

Fig. 1. Operation of DCS_LA with local class accuracy.

class "squares." Four *selection* regions are set up arbitrarily, R_1, R_2, R_3, and R_4, to be used in classifier selection.

During training of the multiple classifier system we decide which classifier from $\mathcal{D} = \{D_1, \ldots, D_L\}$ we should nominate for each region R_j. Thus, the number of classifiers L is not necessarily equal to the number of regions K. Some classifiers might never be nominated and therefore they are not needed in the operation of the combination scheme. Even the classifier with the highest accuracy over the whole feature space might be dropped from the final set of classifiers. On the other hand, one classifier might be nominated for more than one region.

Let D^* be the classifier with the highest average accuracy amongst the elements of $\mathcal{D}$ over the whole feature space $\Re^n$. Denote by $P(D_i|R_j)$ the probability of correct classification by D_i in region R_j. Let $D_{i(j)} \in \mathcal{D}$ be the classifier responsible for region R_j, $j = 1, \ldots, K$. The overall probability of correct classification of our classifier selection system is

$$P(\text{correct}) = \sum_{j=1}^{K} P(R_j) P(D_{i(j)}|R_j) \tag{4}$$

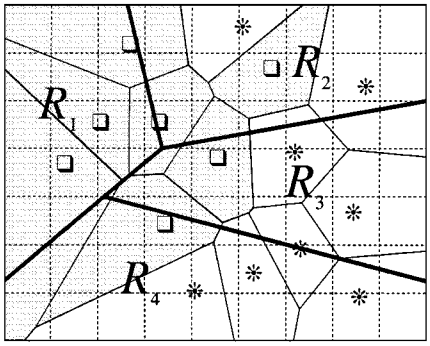

Fig. 2. Example of partitioning the feature space with two classification regions into four selection regions.

where $P(R_j)$ is the probability that an input $\mathbf{x}$ drawn from the distribution of the problem falls in R_j. To maximize $P(\text{correct})$, we assign $D_{i(j)}$ so that

$$P(D_{i(j)}|R_j) \geq P(D_t|R_j), \quad \forall t = 1, \ldots, L. \tag{5}$$

Ties are broken randomly. From (4) and (5)

$$P(\text{correct}) \geq \sum_{j=1}^{K} P(R_j)P(D^*|R_j) = P(D^*). \tag{6}$$

The above equation shows that the combined scheme performs at least as good as the best classifier D^* in the pool $\mathcal{D}$, *regardless of the way the feature space has been partitioned* into selection regions. The only condition (and, of course, the trickiest one) is to ensure that $D_{i(j)}$ is indeed the best amongst the L classifiers from $\mathcal{D}$ for region R_j. The extent to which this is satisfied determines the success of the classifier selection model.

B. The Clustering-and-Selection Model

Based on the above is the simple (static) classifier selection method called *clustering-and-selection (CS)* [13]. Fig. 3 shows the training, and Fig. 4, the operation algorithms.

III. Classifier Fusion

A. A General Model for Classifier Fusion

Classifier fusion assumes that all classifiers are trained over the whole feature space, and are thereby considered as *competitive* rather than *complementary* [17], [27]. We can treat the classifier outputs as the input to a second-level classifier in some *intermediate feature space*, and design a new classifier for the second (combination) level.[1] The classifier outputs can be organized in a *decision profile* [15] as the matrix in

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

The entries in $DP(\mathbf{x})$ are the intermediate features. To build a minimum-error classifier, we replace the problem of estimating

[1]This is called "stacked generalization" in [25].

Clustering and selection (training)

1. Design the individual classifiers $D_1, \ldots, D_L$ using the labeled data set $\mathbf{Z}$. Pick the number of regions K.
2. Disregarding the class labels, cluster $\mathbf{Z}$ into K clusters, $C_1, \ldots, C_K$, using, e.g., the K-means clustering procedure [5]. Find the cluster centroids $\mathbf{v}_1, \ldots, \mathbf{v}_K$ as the arithmetic means of the points in the respective clusters.
3. For each cluster C_j, (defining region R_j), estimate the classification accuracy of $D_1, \ldots, D_L$ using only those elements of $\mathbf{Z}$ which are in C_j.[a] Nominate the classifier with the highest accuracy as $D_{i(j)}$.
4. Return $\mathbf{v}_1, \ldots, \mathbf{v}_K$ and $D_{i(1)}, \ldots, D_{i(K)}$.

[a]This estimate could be calculated through resubstitution, or by some pseudo-testing, e.g., by cross-validation or bootstrapping. For the purposes of our experiment described later we used resubstitution.

Fig. 3. Training of the CS method.

Clustering and selection (operation)

1. Given the input $\mathbf{x} \in \Re^n$, find the nearest cluster center from $\mathbf{v}_1, \ldots, \mathbf{v}_K$, say, $\mathbf{v}_j$.
2. Use $D_{i(j)}$ to label $\mathbf{x}$, i.e., $\mu_k(\mathbf{x}) = d_{i(j),k}(\mathbf{x}), \quad k = 1, \ldots, c$.

Fig. 4. Operation of the CS method.

$P(\omega_i|\mathbf{x})$ with one of estimating $P(\omega_i|D_1(\mathbf{x}), \ldots, D_L(\mathbf{x}))$, or more compactly, $P(\omega_i|DP(\mathbf{x}))$. Thus, the initial feature space with n features, $\Re^n$, is transformed into a new space with $L \times c$ features. This treatment of the combination problem underpins the schemes in [1], [7], [8], [10], [23], and [24]. In a way, this idea is akin to support vector machines approach where the initial feature space is transformed in a new (generally higher dimensional) space and the classifier is built in that new space [20]. However, in the model here, the intermediate feature space has a special context-related structure on which we can base our combination model [15].

Some fusion methods calculate the support for class ω_i using only the ith column of $DP(\mathbf{x})$, i.e., the individual support for ω_i given by $D_1, \ldots, D_L$, regardless of what the support for the other classes is. Simple and widely used members of this group are the minimum, maximum, average, and product, taken columnwise on the decision profile $DP(\mathbf{x})$. For example, let $L = 3$, $c = 2$, and the decision profile obtained for $\mathbf{x}$ be

$$DP(\mathbf{x}) = \begin{bmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.5 & 0.5 \end{bmatrix}. \tag{8}$$

The first column of $DP(\mathbf{x})$ is the support for ω_1, and the second column is the support for ω_2. The maximum aggregation rule will give $\mu(\mathbf{x})$ with $\mu_1(\mathbf{x}) = \max\{0.3, 0.6, 0.5\} = 0.6$ and $\mu_2(\mathbf{x}) = \max\{0.7, 0.4, 0.5\} = 0.7$, and subsequently label $\mathbf{x}$ in ω_2. For the minimum rule $\mu(\mathbf{x}) = [0.3, 0.4]$, for the product rule $\mu(\mathbf{x}) = [0.09, 0.14]$ and for the average rule $\mu(\mathbf{x}) = [0.47, 0.53]$. Here, all the aggregation rules agree on labeling $\mathbf{x}$ in ω_2 but this need not be the case for a different

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

$$DT_i = \frac{1}{N_i} \sum_{\substack{\mathbf{z}_j \in \omega_i \\ \mathbf{z}_j \in \mathbf{Z}}} DP(\mathbf{z}_j), \tag{9}$$

where N_i is the number of elements of $\mathbf{Z}$ from ω_i.
2. Return $DT_1, \ldots, DT_c$.

Fig. 5. Training of the DT method.

DECISION TEMPLATES (OPERATION)

1. Given the input $\mathbf{x} \in \Re^n$, construct $DP(\mathbf{x})$ as in (7).
2. Calculate the squared Euclidean distance between $DP(\mathbf{x})$ and each DT_i, $i = 1, \ldots, c$

$$d_E(DP(\mathbf{x}), DT_i) = \sum_{j=1}^{c} \sum_{k=1}^{L} (d_{k,j}(\mathbf{x}) - dt_i(k,j))^2, \tag{10}$$

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

$$\mu_i(\mathbf{x}) = 1 - \frac{1}{L \cdot c} d_E(DP(\mathbf{x}), DT_i). \tag{11}$$

Fig. 6. Operation of the DT method.

example. The above models are derived in [11] as different estimates of $P(\omega_i|\mathbf{x})$ under the assumption of conditionally independent $D_1, \ldots, D_L$.

B. Decision Templates (DTs)

Using DT for combining classifiers is proposed in [15]. Given L (trained) classifiers in $\mathcal{D}$, c DT are calculated from the data, one per class. The decision template for class ω_i, denoted DT_i is the centroid of class ω_i in the intermediate feature space. DT_i can be regarded as the expected $DP(\mathbf{x})$ for class ω_i. The support for class ω_i offered by the combination of the L classifiers, $\mu_i(\mathbf{x})$, is then found by measuring the *similarity* between the current $DP(\mathbf{x})$ and DT_i. We use Euclidean distance for calculating the similarity but other measures can also be applied. In [15], we view $DP(\mathbf{x})$ and DT_i as two fuzzy sets defined over the set of intermediate features and use measures of similarity from fuzzy set theory. Fig. 5 describes the training and Fig. 6, the operation of the DT model.

We illustrate the DT model with a numerical example. Let $c = 2$ and $L = 3$, and let the DT_i, $i = 1, 2$, calculated from the data be

$$DT_1 = \begin{bmatrix} 0.6 & 0.4 \\ 0.8 & 0.2 \\ 0.5 & 0.5 \end{bmatrix} \quad \text{and} \quad DT_2 = \begin{bmatrix} 0.3 & 0.7 \\ 0.4 & 0.6 \\ 0.1 & 0.9 \end{bmatrix}. \tag{12}$$

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

$$DP(\mathbf{x}) = \begin{bmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \\ 0.5 & 0.5 \end{bmatrix}. \tag{13}$$

Using (11) and (10), the soft label of $\mathbf{x}$ is

$$\mu_1(\mathbf{x}) = 0.96, \qquad \mu_2(\mathbf{x}) = 0.93. \tag{14}$$

If the classifier outputs are some estimates of the posterior probabilities $P(\omega_k|\mathbf{x})$, $k = 1, \ldots, c$, the decision template DT_i is an unbiased estimate of the expectation of the $L \times c$-dimensional random variable $DP(\mathbf{x})$ given that the true class is ω_i. Therefore, assessing the similarity between the actually occurred matrix of outputs $DP(\mathbf{x})$ and the expected one for ω_i is a reasonable classification strategy. DT have been found to rate among the best combination methods and show stable performance over a range of experimental setups [15]. This was the reason to pick this model as a component in the proposed combination.

Four rival classifier fusion techniques used in the experimental study are described in the Appendix: majority vote, naive Bayes, and two joint-distribution methods.

IV. Combining Clustering and Selection With Decision Templates

A. Using Confidence Intervals for Selecting a Classifier

Clustering and selection is guaranteed by design to give at least the same training accuracy as the best individual classifier D^*. However, the model might overtrain, giving a deceptively low training error. Hopefully, using CIs and nominating a classifier only when it is significantly better than the others, will be a basis of a combination scheme less prone to overfitting and spurious errors.

A way to reduce the possible overtraining is to perform a statistical test for determining whether the best classifier in R_j, $D_{i(j)}$, is significantly different from the remaining lot. Looney proposed a statistical method for comparing L classifiers based on an adjusted F-test [16]. Since we are interested only in a difference between the best classifier and the rest, we can perform pairwise tests such as the paired t-test. It is enough to eliminate the second best classifier. If $D_{i(j)}$ is significantly better than the second best, then $D_{i(j)}$ can be nominated as the classifier responsible for region R_j. Otherwise, a scheme involving more than one classifier might pay off.

As an example, assume that five classifiers have been designed on a dataset with 100 elements. Define

$$\mathbf{y}_i = [y_{1,i}, \ldots, y_{100,i}]^T \tag{15}$$

to be a vector with classification outcome of classifier D_i on the dataset, such that $y_{j,i} = 1$, if D_i recognizes correctly the jth element of the dataset, and 0, otherwise. Table I shows the distribution of $\{\mathbf{y}_1, \ldots, \mathbf{y}_5\}$ for the 100 elements. The total number of correctly recognized objects is shown in the bottom row for each classifier. We could be tempted to nominate D_1 for region R_j as its classification accuracy is 76%, by 5% higher than the

TABLE I
DISTRIBUTION OF CORRECT/INCORRECT CLASSIFICATION DECISIONS FOR FIVE CLASSIFIERS FOR A DATA SET WITH 100 ELEMENTS (NOTICE THAT NOT ALL POSSIBLE COMBINATIONS HAVE OCCURRED). THE BOTTOM ROW CONTAINS THE TOTAL NUMBER OF CORRECTLY RECOGNIZED OBJECTS FOR EACH CLASSIFIER

y_1	y_2	y_3	y_4	y_5	Number
1	1	1	1	1	42
0	0	0	1	1	18
1	1	0	0	0	13
1	0	0	1	0	11
1	0	1	0	0	10
0	0	0	0	1	6
76	55	52	71	66	–

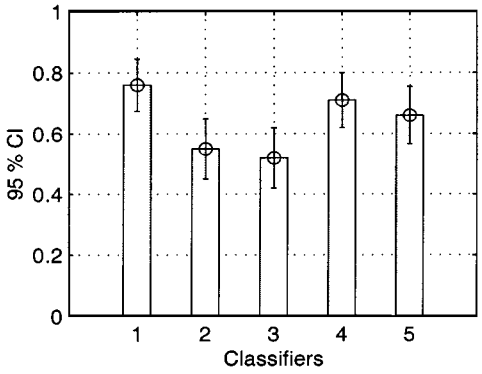

Fig. 7. 95% CI for the five classifiers.

second best. However, the paired t-test analysis suggests that D_1 is not significantly different from D_4 ($p = 0.438$), not even from D_5 ($p = 0.191$). Therefore, it is probably better to consider a fusion scheme of the decisions of more than one classifier in R_j.

Fig. 7 shows the 95% CIs of the classification accuracies of the five classifiers. The intervals have been calculated through the standard formula

$$\left[\hat{P}_D - t_{(0.05,\,N-1)}\left(\sqrt{\frac{\hat{P}_D(1-\hat{P}_D)}{N}}\right),\ \hat{P}_D + t_{(0.05,\,N-1)}\sqrt{\frac{\hat{P}_D(1-\hat{P}_D)}{N}}\right] \quad (16)$$

where N is the sample size, $t_{(0.05,\,N-1)}$ is the t value for 95% significance level ($\alpha = 0.05$) and $N-1$ degrees of freedom, and $\hat{P}_D$ is the estimate of the classification accuracy of the respective D_i in region R_j. Equation (16) is an easy (but not very accurate) way to check whether classifiers are significantly different (for $N > 100$ we can use $t_{(0.05,\,N-1)} = 1.96$).

The above calculations are based on the assumption that $P(D_i|R_j)$ is the same for all $\mathbf{x} \in R_j$ and has therefore a Binomial distribution. We can derive an equation for calculating the "gap" needed so that the 95% CI of the best and the second best classifiers do not overlap. In this case, the best classifier

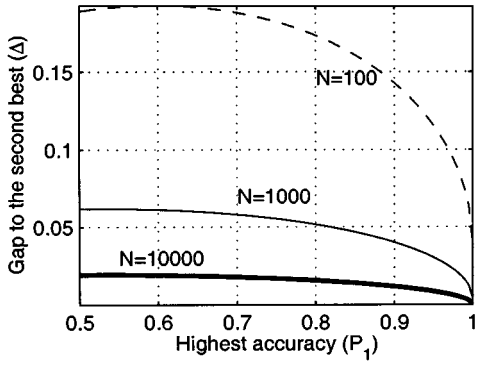

Fig. 8. Difference Δ between the best and the second best classification accuracies in region R_j guaranteeing that the 95% CI of the two do not overlap.

can be nominated as $D_{i(j)}$. Let $\Delta = P_1 - P_2$, $\Delta > 0$, where P_1 is the highest accuracy and P_2 is the second highest in R_j. The critical value for the gap Δ is derived from

$$1.96\sqrt{\frac{P_1(1-P_1)}{N}} + 1.96\sqrt{\frac{P_2(1-P_2)}{N}} = \Delta. \quad (17)$$

Substituting $P_2 = P_1 - \Delta$ and solving for Δ, we obtain

$$\Delta = \frac{7.6832\,P_1 - 3.8416 + 3.92\sqrt{NP_1(1-P_1)}}{N + 3.8416}. \quad (18)$$

Fig. 8 plots Δ against P_1 for three values of N: 100, 1000, and 10 000. For larger N the required gap for the best and the second best classifiers to be significantly different is smaller. In other words, even a 2% difference in the classification accuracy could be enough background to nominate $D_{i(j)}$ as the classifier responsible for region R_j.

B. The CS+DT Classifier Combination Model

The combination is a straightforward application of the two models. First, the CS model is applied. Then, CIs on the accuracy of the L classifier are calculated for cluster C_j (using only the data points in C_j) through (16). Based on the overlap, either one classifier $D_{i(j)}$ is selected in charge of region R_j, or the DT model is used with all L classifiers.

There are various ways to implement the combination. Here, we used the following one. We nominated a classifier for region R_j if the 50% CI (t value 0.675) of the best accuracy did not contain the second best accuracy. The reason for this choice was that since classifiers have similar (presumably high) accuracies, and the number of data points is moderate in each cluster, it is unlikely to obtain nonoverlapping CIs. Therefore we will have to use DT in all regions, so the combination of the two approaches becomes meaningless. Second, we used a single DT calculation, using all data points in $\mathbf{Z}$, and not an individual DT model for each cluster. This was adopted as the simplest possible solution.

We have to be cautious though because although each of the constituent methods have their reasonable probabilistic background, there is no guarantee that the combined model will combine their strengths.

TABLE II
SUMMARY OF THE DATASETS USED

Database	n	c	N	$\hat{P}_{max}$	Past usage[a]	Availability
Cone-torus	2	3	800	50.00 %	85-90 %	Private[b]
Phoneme	5	2	5404	70.65 %	86-91 %	ELENA[c]
Pima Indians Diabetes	8	2	768	65.10 %	76-80 %	UCI[d]
Cleveland Heart Disease	13	2	303	54.48 %	75-79 %	UCI[d]
Wisconsin Diagnostic Breast Cancer	30	2	569	62.74 %	97.5 %	UCI[d]

Notations:
n: number of features
c: number of classes
N: number of cases in the database
$\hat{P}_{max}$: the largest class proportion

[a] Details are given in the text
[b] http://www.bangor.ac.uk/~mas00a/Z.txt and Zte.txt
[c] ftp ftp.dice.ucl.ac.be, directory pub/neural/ELENA, follow the instructions of README.
[d] http://www.ics.uci.edu/ mlearn/MLRepository.html

V. EXPERIMENTS

A. Experimental Setup

A series of experiments has been carried out to check whether CS+DT is better than either CS or DT applied individually, and also how CS+DT rates among other classifier combination methods.

We used five datasets as summarized in Table II

1) **Cone-torus** data. This is a three-class dataset with 400 two-dimensional points generated from three differently shaped distributions: 1) a cone, 2) half a torus, and 3) a normal distribution with prior probabilities 0.25, 0.25, and 0.5, respectively. This dataset is available on http://www.bangor.ac.uk/~mas00a/Z.txt. A separate dataset for testing with 400 more points generated from the same distribution is also available as the file Zte.txt. Experiments with various classification methods on this dataset are reported in [14]. The accuracy of the nearest neighbor method (1-nn) was 84.85%, Parzen classifier gave 87.75%, different configurations and parameter settings of multilayer perceptron (MLP) and RBF neural networks (NNs) resulted in 84.25–89.50% accuracy, and fuzzy classifiers exhibited a large range of performances, the best being a version of a Wang–Mendel model with 90.25% accuracy.
2) **Phoneme** data. This set is from the ELENA database It consists of 5404 five-dimensional vectors characterizing two classes of phonemes: 1) nasals (70.65%) and 2) orals (29.35%). As reported in the database, the set has been classified using 1-nn and 20-nn with results 91.03% and 85.80%, respectively. In both cases, the leave-one-out method was applied.
3) **Pima Indians Diabetes** data. A population of women living near Phoenix, AZ, of Pima Indian heritage were tested for diabetes according to World Health Organization criteria. The dataset is available within the UCI Machine Learning Repository. Reported in the database are results from past usage of data, giving 76% accuracy in a hold-up experiment with 576 training and 192 testing data points. Because of missing values, Ripley uses a subset of 532 cases in his book [19], and reports accuracy of 80%. The version offered at the UCI site contains 768 complete cases (no missing values).
4) **Cleveland Heart Disease** data.[2] The presence or absence of heart disease is predicted based on 13 features. The dataset consists of 303 patient records, of which 164 did not have the disease. There are a few missing values in the data. In our experiments, these were replaced by the average of the column (feature) regardless of the class labels. The classification accuracy on this data, reported in the UCI database, varies between 75% and 79%.
5) **Wisconsin Diagnostic Breast Cancer** data. Features are computed from a digitized image of a fine needle aspirate of a breast mass. The mass is classed as benign or malignant. The results of past usage of the dataset are summarized in the UCI database: the sets are linearly separable using all 30 input features. Using just three of the 30 features, a linear classifier model was built with accuracy of 97.5% estimated by repeated tenfold crossvalidation.

With the Cone-torus and Phoneme datasets we performed twofold cross validation so that the results were comparable with those in [14] and [26]. With the other three datasets, tenfold cross validation experiments were carried out. With the exception of the Phoneme data, the sample size is not sufficiently large to allow for a reasonable part of the training to be taken aside for validation. Therefore, we did not consider splitting the training set into two. All of the choices of parameters and the classifier tuning was done on the training sets only.

In the first series of experiments, we trained $L = 3, 7,$ and 11 MLP NNs as the classifiers D_i. Each MLP had one hidden layer with M nodes in it. We carried out three sets of experiments with $M = 5, 10,$ and 20. The hidden and the output nodes had sigmoidal activation functions, so the outputs $d_{i,j}(\mathbf{x})$ were in the interval $[0, 1]$. Fast backpropagation was used to train the NNs (Matlab Neural Network Toolbox). To preserve difference between classifiers, we deliberately left the NNs undertrained, applying only 100 training epochs in each experiment.

In the second series of experiments, we trained five *different* classifier models: the linear discriminant classifier (LDC); the quadratic discriminant classifier (QDC); k-nn (1-nn for the Phoneme data; and the best k chosen on the training set for the remaining datasets); an MLP NN, and a tree classifier. We tried to reproduce the experiment in [26] to compare the proposed method with the DCS_LA proposed there. However, the details of the MLP structure and training were not available, so we trained our own version with $M = 10$ hidden nodes and 300

[2] Dr. Robert Detrano was responsible for the data collection of the database, V.A. Medical Center, Long Beach and Cleveland Clinic Foundation.

TABLE III
RESULTS FROM THE FIRST SERIES OF EXPERIMENTS: $L = 11$ MLP CLASSIFIERS, $M = 20$ HIDDEN NODES. "A" STANDS FOR ACCURACY (GIVEN IN %), AND "R" STANDS FOR RANK. NOTE THAT THE RANKS ARE CALCULATED ACROSS NINE EXPERIMENTS ($L = 3, 7, 11$ AND $M = 5, 10, 20$)

Classifier	Cone-torus		Phoneme		Pima Diabetes		Cleveland		Breast Cancer	
/combiner	A	R	A	R	A	R	A	R	A	R
SB	85.1	85.0	78.5	96.0	77.1	98.5	82.5	53.5	97.4	92.5
MAJ	74.1	58.5	70.7	46.0	**77.5**	116.5	81.8	89.5	97.5	82.5
NB	82.5	92.0	71.7	33.5	77.4	**124.0**	82.1	105.0	**97.7**	100.5
BKS	-	73.5	78.6	107.5	75.4	87.0	80.2	25.5	97.2	60.0
WER	-	67.5	78.6	108.5	75.3	97.0	79.9	39.5	97.0	64.0
MAX	85.4	58.5	70.7	28.5	73.1	14.5	82.1	83.0	83.9	17.0
MIN	49.4	23.5	70.7	24.5	71.1	19.0	83.7	97.5	92.4	56.0
AVR	81.1	79.0	70.7	33.0	**77.5**	101.0	82.8	113.5	97.5	112.5
PRO	54.3	30.0	70.7	24.5	73.1	35.0	83.1	106.0	97.2	89.5
DT	85.8	60.0	78.1	84.5	75.8	89.0	82.1	**121.0**	97.4	91.0
ORA	96.6	162.0	89.9	162.0	89.5	162.0	91.3	162.0	98.6	162.0
DCS	**86.5**	**150.5**	**81.9**	**145.0**	75.4	68.5	79.8	33.5	97.5	**135.0**
CS(3)	84.6	108.0	78.5	111.5	77.2	100.0	81.1	53.5	97.4	58.5
CS+DT(3)	85.8	65.0	77.5	88.0	75.8	94.5	82.1	100.0	97.4	83.0
CS(5)	85.6	117.5	78.6	117.5	77.1	77.5	82.2	80.0	97.2	69.0
CS+DT(5)	85.8	89.0	77.9	95.5	75.8	97.5	83.4	108.0	**97.7**	92.0
CS(8)	86.0	128.0	78.8	125.5	76.7	69.5	81.2	64.0	**97.7**	85.5
CS+DT(8)	86.4	91.5	78.3	107.5	75.8	88.0	**84.1**	104.0	97.4	88.5

training epochs. The same protocol was used for the Pima Indians Diabetes data and the Wisconsin Breast Cancer data. For the Cone-torus data, we trained an MLP with $M = 20$ hidden nodes for 100 epochs, and for the Cleveland Heart Disease data, an MLP with $M = 10$ hidden nodes, for 50 epochs.

The code that was used for LDC, QDC, k-nn (including the tuning of k), and the classification tree was from the Matlab Toolbox "PRTools" [6]. The MLP code was from the Neural Network Toolbox for Matlab as in the first series of experiments.

All datasets were normalized in the following way. A linear transformation was used, separately for each feature, to bring its values in the interval $[0, 1]$. The *training set* was used to find the minimum and the maximum of the feature values. The testing set was transformed using these same constants, so presumably there would be values lower than 0 and higher than 1.

The following classifier combination methods were coded in Matlab:

1) SB: Single best. We take the classifier from the pool that exhibits the smallest *training* error rate.
2) OR: Oracle. The oracle works as follows: assign the *correct* class label to $\mathbf{x}$ iff at least one individual classifier produces the correct class label of $\mathbf{x}$.
3) Fusion methods
   a) MAJ: Majority vote (see Appendix).
   b) NB: Naive Bayes (see Appendix).
   c) BKS (see Appendix).
   d) WER: Wernecke (see Appendix).
   e) MAX: Maximum (Section III-A).
   f) MIN: Minimum (Section III-A).
   g) AVR: Average (Section III-A).
   h) PRO: Product (Section III-A).
   i) DT: Decision templates (Section III-B).
4) Dynamic classifier selection (DCS) methods: with $K = 9$ (Woods *et al.* [26]).
5) Static classifier selection methods: CS(K): Clustering and selection into $K = 3, 5,$ and 8 clusters (Section II-B). For each K, ten runs were carried out with different initializations of the k-means clustering routine. The best (training) CS result was taken forward.
6) Switch between static selection and fusion: CS+DT (K): Clustering and selection combined with DT (Section V-B, again three, five, and eight clusters). As with the static selection, ten runs were carried out with different initializations of the k-means clustering routine for each K, and the best (training) CS+DT result was taken forward.

B. Results

In the first series, there were nine experiments with each combination method for a given dataset: using the three MLP configurations with $M = 5, 10,$ and 20 hidden nodes and ensembles of $L = 3, 7,$ and 11 MLPs. To save space, Table III shows the testing accuracies (columns "A") only for $M = 20$ and $L = 11$.[3] The oracle line separates the classifier fusion from classifier selection (selection+fusion) methods. Marked in boldface are the best accuracies in each column. To facilitate the comparison we calculated the relative performance of each method with respect to the others. For each dataset, the nine columns with the accuracies were sorted individually, and each combination model was assigned a rank with respect to its place among the others. The highest possible rank was 18 (assigned to the best model) and the lowest was 1 (assigned to the worst model). Expectedly, the best combination model was the oracle but since it is an abstraction rather than a real combination model, it is excluded from the comparison. The nine ranks for each combination model were then summed to give a measure of the overall dominance among the models for the particular dataset. The columns marked with "R" in Table III show the ranks of the models for the respective datasets.

[3]The full set of results is available at http://www.bangor.ac.uk/~mas00a/papers/kuncheva_smc_tables.ps.gz or by request from the author.

TABLE IV
RESULTS FROM THE SECOND SERIES OF EXPERIMENTS: FIVE *DIFFERENT* CLASSIFIERS. "A" STANDS FOR ACCURACY (GIVEN IN %), AND "R" STANDS FOR RANK

Classifier	Cone-torus		Phoneme		Pima Diabetes		Cleveland		Breast Cancer	
/combiner	A	R	A	R	A	R	A	R	A	R
LDC	75.0	1.0	75.8	2.0	**76.8**	21.0	**84.7**	21.0	95.6	2.0
QDC	80.8	3.0	78.7	4.0	74.2	14.0	81.7	8.5	95.6	3.0
kNN	86.4	19.5	**86.8**	19.0	75.1	16.0	79.6	3.0	96.8	13.0
MLP	78.0	2.0	70.7	1.0	75.4	17.0	80.2	4.5	**97.5**	20.5
TRE	85.1	11.5	84.2	6.5	71.6	6.0	80.2	4.5	95.1	1.0
MAJ	84.1	6.5	84.2	6.5	76.7	20.0	82.4	12.5	97.4	18.5
NB	85.6	17.0	85.3	11.5	75.9	18.0	82.4	12.5	**97.5**	20.5
BKS	85.8	18.0	85.3	11.5	71.1	1.0	78.8	1.0	96.5	7.0
WER	85.3	14.0	85.9	13.0	71.4	3.0	79.4	2.0	96.0	4.0
MAX	84.5	8.0	78.1	3.0	73.3	13.0	83.4	20.0	96.5	8.0
MIN	83.6	4.0	84.2	6.5	73.0	12.0	82.4	12.5	96.7	10.0
AVR	84.1	6.5	84.3	9.0	76.3	19.0	83.1	18.0	97.4	18.5
PRO	83.9	5.0	84.2	6.5	72.7	10.0	83.1	18.0	97.0	15.0
DT	**86.5**	21.0	84.7	10.0	72.9	11.0	83.1	18.0	97.2	16.5
ORA	96.1	22.0	98.0	22.0	88.4	22.0	91.9	22.0	99.1	22.0
DCS	86.4	19.5	86.1	14.0	74.4	15.0	81.1	6.0	97.1	16.5
CS(3)	85.1	11.5	**86.8**	19.0	71.6	6.0	82.1	10.0	96.7	9.0
CS+DT(3)	85.1	11.5	**86.8**	19.0	71.5	4.0	82.8	15.5	96.1	5.0
CS(5)	85.1	11.5	86.8	15.5	71.9	8.0	81.5	7.0	97.0	14.0
CS+DT(5)	84.9	9.0	86.8	15.5	71.6	6.0	82.8	15.5	96.8	11.5
CS(8)	85.5	15.5	**86.8**	19.0	71.4	2.0	81.7	8.5	96.5	6.0
CS+DT(8)	85.5	15.5	**86.8**	19.0	72.3	9.0	82.4	12.5	96.8	11.5

Table IV shows the results from the second series of experiments where five *different* classifier models were combined. The top five rows display the accuracies of the individual classifiers, next are the accuracies of the combination, and the oracle line separates the classifier fusion from classifier selection (selection + fusion) methods. Marked in boldface are the best error rates in each column. Next to each accuracy column is the rank column, where the ranks are calculated as in the first series of experiments.

The sums of ranks in the two experiments are visualized in Fig. 9.

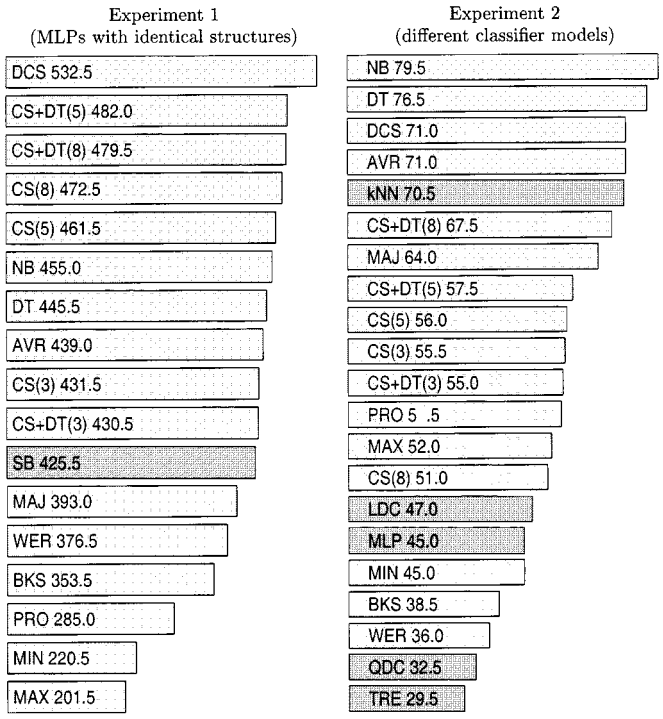

Fig. 9. Bar graphs of the sums of ranks for the two series of experiments. The longer the bar, the better the combination method. Dark gray bars correspond to the individual classifiers.

C. Discussion

The two experiments offer different insights into the combination problem. In the first series, we have identical MLP structures and the only tool to induce diversity into the team is the initialization. Wang *et al.* [22] note that different initializations are possibly the weakest tool for engineering diversity, compared to feature or data selection, data perturbation, or *using different classifier models*. However, the accuracy of the ensembles using MLPs was comparable, and for the Wisconsin Breast Cancer data even better than that in the second series, where different classifier models were combined. Notice that the improvement over the single best classifier is more visible in the first series, as in the second series, there was already a strong model in the team.

It is reasonable to expect that classifier selection methods will perform better when the individual classifiers are of different accuracies. That is, the "selector" will choose the "best" classifier for the respective region or sample, and ignore the less accurate classifiers. Conversely, classifier fusion models will tend to smooth out the differences and will probably be slightly inferior to the most outstanding classifier in the team. As the experiments show, this intuition is not necessarily correct. The reason roots in the training/testing relationship which are different for different classifier models. For example, the linear classifier can be "trusted" to perform approximately as good as on training set. The generalization could be poor but so would be the training accuracy. (Curiously, the linear classifier was amongst the best classifier models in the real datasets, and the top rating classifier on Pima Indian Diabetes data and the Cleveland Heart Disease data.) When the neural net or the tree are trained, it is impossible

to stop at the exact place so that we match the training/testing correspondence that is typical for simple models (LDC, QDC). Thus the training error of the MLP or the classification tree might be deceptively low and the classifier selection models will keep selecting these two classifiers, resulting in high testing error. Using a validation set is of course an option but the sample size not always allows that. Besides, the estimate on the validation set could be misleading as well. Thus, there is always likely to be a discrepancy between the training/testing correspondence if different classifier models are used, and the selection methods might be fooled. This seems to be the reason for the inferior performance of the selection methods in the second series of experiments (little or no improvement on the individual accuracy).

The overall impression from the two experiments is that DCS is the best strategy. It requires more computational resources than the other methods but the pay-off is its stable performance and high accuracy.

The switching between selection and fusion has some potential in it. The choice of the DT method as the fusion component was based on the good overall performance of the method in our previous experiments, and was confirmed again here. However, naive Bayes and average have also shown adequate accuracy and robustness, and can replace DT in the CS+DT scheme. The clustering part can be improved as well. The choice of three, five, or eight clusters here was arbitrary. A more systematic approach to the partitioning of the feature space could lead to a better solution.

BKS and Wernecke's methods showed unexpectedly poor performance. In the first series of experiments they were not run for 11 MLPs and three classes (Cone-torus data) due to time reasons, and so their overall ranks were affected by that. In the second series of experiments, the two methods were probably overfitted and also suffered from the differences in the training/testing correspondences explained above.

There were differences in the individual classifier performances in both experiments. In the first series, the training of some of the NNs was stuck in a local extremum, producing a very poor member of the team (40%–50% error). In the second series of experiments, the classifiers did have very different accuracies. Some of the simple fusion methods are sensitive to such anomalies. This led to the unsatisfactory performance of MIN, MAX, and PRO.

Looking at the results from the second series of experiment, the arising question is "to combine or not to combine?" Choosing one individual classifier over the rest will depend on how much we trust our estimate of the generalization of the classifiers. If there is no clear preference, DCS appeared to be a good option when classifiers were of the same type. However, it was not picked as the best solution in any of the experiments in the second series. Trying to reproduce the experiment for the Phoneme data from [26], we found that the combined approach proposed here CS+DT, the best in this experiment, only just reached the performance of the nearest neighbor classifier. So the merit of a combination in the case of different classifier models should be carefully looked at. Using weights (where possible) might help overcome the discrepancy problem.

The potential for using the methods discussed in this paper have to be assessed with respect to the growing amounts of data in real-life problems. The small sample size problems of the past are now being replaced by extremely-large size problems where the user is faced with gigabytes and even terabytes of data. It is likely that the focus of the future research will be dictated by this shift and classifier combination methods will be of a different value in this light. DCS which relies on on-line estimates of the classifier accuracy in the vicinity of $\mathbf{x}$ will take a lot of computational resources and, though highly accurate, might become cumbersome to run. Therefore, theoretical and algorithmic amendments of classifier combination methods (and classifier design methods in the first place) will be needed.

VI. Conclusions

A combination consisting of a classifier selection and a classifier fusion method is proposed: CS and DT. First, the data is clustered to form regions in the feature space. Second, we use CIs to decide whether one classifier should be nominated to make the decision in a certain region or DT should be applied instead. One synthetic and four real datasets were used to find out how the switching between selection and fusion rates amongst the DCS, the static selection and the fusion. DCS was the best alternative for classifiers with the same structure and training protocols (MLP). CS+DT was the second best in this series of experiments. For different classifier models in the team, it is difficult to predict whether a combination can achieve a better accuracy than the best individual classifier. In this series of experiments there was no clear preference of one combination approach over the rest. The result depended on the dataset, the only consistent pattern being that the improvement over the best individual classifier was negligible.

A curious result was the excellent performance of the Naive Bayes combiner which is often overlooked. The average method kept its reputation as a reasonably accurate and stable combiner, whereas the other simple fusion methods were not as good as deemed in the literature.

Appendix

The four methods described below operate on crisp classifier outputs, i.e., where $DP(\mathbf{x})$ contains only values 0 and 1 and each row has exactly one 1 in it.

A. Majority Vote

The class label assigned to $\mathbf{x}$ is the one that is most represented in the set of L class labels $D_1(\mathbf{x}), \ldots, D_L(\mathbf{x})$. Using the (crisp) decision profile $DP(\mathbf{x})$, the majority vote method is implemented by summing up the $DP(\mathbf{x})$ columns and taking the index of the column with the highest score as the class label of $\mathbf{x}$. Ties are broken randomly.

B. Naive Bayes

This scheme assumes that the classifiers are mutually independent (this is the reason we use the name "naive"); for each classifier D_j, a $c \times c$ confusion matrix CM^j is calculated by applying D_j to the training dataset. The (k, s)th entry of this matrix, $cm^j_{k,s}$ is the number of elements of the dataset whose true class label was ω_k, and were assigned by D_j to class ω_s.

By $cm^j_{\cdot,s}$ we denote the total number of elements labeled by D_j into class ω_s (this is calculated as the sum of the sth column of CM^j). Using $cm^j_{\cdot,s}$, a $c \times c$ label matrix LM^j is computed, whose (k, s)th entry $lm^j_{k,s}$ is an estimate of the probability that the true label is ω_k given that D_j assigns crisp class label ω_s

$$lm^j_{k,s} = \hat{P}(\omega_k | D_j(\mathbf{x}) = \omega_s) = \frac{cm^j_{k,s}}{cm^j_{\cdot,s}}. \tag{19}$$

For every $\mathbf{x} \in \Re^n$, D_j yields a crisp label vector $D_j(\mathbf{x})$ pointing at one of the classes, say, $\omega_s \in \Omega$. Considering the label matrix for D_j, LM^j, associated with ω_s is a *soft label vector* $[\hat{P}(\omega_1|D_j(\mathbf{x}) = \omega_s), \ldots, \hat{P}(\omega_c|D_j(\mathbf{x}) = \omega_s)]^T$, which is the sth column of the matrix. Let $s_1, \ldots, s_L$ be the crisp class labels assigned to $\mathbf{x}$ by classifiers $D_1, \ldots, D_L$, respectively. Then, by the independence assumption, the estimate of the probability that the true class label is ω_i, is calculated by

$$\mu_i(\mathbf{x}) \propto \prod_{j=1}^{L} \hat{P}(\omega_i|D_j(\mathbf{x}) = s_j) = \prod_{j=1}^{L} lm^j_{i,s_j} \qquad i = 1, \ldots, c. \tag{20}$$

As an example, consider a problem with $L = 2$ classifiers, D_1 and D_2, and $c = 3$ classes. Let the number of training data points be $N = 20$. From these, let eight be from ω_1, nine from ω_2, and three from ω_3. Suppose the following confusion matrices have been obtained for the two classifiers:

$$CM^1 = \begin{bmatrix} 6 & 2 & 0 \\ 1 & 8 & 0 \\ 1 & 0 & 2 \end{bmatrix} \quad \text{and} \quad CM^2 = \begin{bmatrix} 4 & 3 & 1 \\ 3 & 5 & 1 \\ 0 & 0 & 3 \end{bmatrix}. \tag{21}$$

The two label matrices obtained from CM^1 and CM^2 are

$$LM^1 = \begin{bmatrix} 6/8 & 2/10 & 0 \\ 1/8 & 8/10 & 0 \\ 1/8 & 0 & 1 \end{bmatrix}$$

and

$$LM^2 = \begin{bmatrix} 4/7 & 3/8 & 1/5 \\ 3/7 & 5/8 & 1/5 \\ 0 & 0 & 3/5 \end{bmatrix}. \tag{22}$$

Assume $D_1(\mathbf{x}) = \omega_2$ and $D_2(\mathbf{x}) = \omega_1$ for the input $\mathbf{x} \in \Re^n$. Using the second column of LM^1 and the first column of LM^2, we calculate the output of the Naive Bayes classifier fusion scheme as follows:

$$\mu_1(\mathbf{x}) \propto \hat{P}(\omega_1|D_1(\mathbf{x}) = \omega_2)\ \hat{P}(\omega_1|D_2(\mathbf{x}) = \omega_1) = \tfrac{4}{35}$$
$$\mu_2(\mathbf{x}) \propto \hat{P}(\omega_2|D_1(\mathbf{x}) = \omega_2)\ \hat{P}(\omega_2|D_2(\mathbf{x}) = \omega_1) = \tfrac{12}{35}$$
$$\mu_3(\mathbf{x}) \propto \hat{P}(\omega_3|D_1(\mathbf{x}) = \omega_2)\ \hat{P}(\omega_3|D_2(\mathbf{x}) = \omega_1) = 0.$$

The maximum membership rule will class $\mathbf{x}$ in ω_2.

1) Behavior-Knowledge Space (BKS): Let again $(s_1, \ldots, s_L) \in \Omega^L$ be the crisp class labels assigned to $\mathbf{x}$ by classifiers $D_1, \ldots, D_L$, respectively. Every possible combination of class labels is an index regarded as a cell in a lookup table (BKS table)[8]. The table is designed using the dataset $\mathbf{Z}$ (joint-distribution of the class labels). Each $\mathbf{z}_j$ is placed in the cell indexed by $D_1(\mathbf{z}_j), \ldots, D_L(\mathbf{z}_j)$. The elements in each cell are tallied and the most representative class label is selected for this cell. Ties are broken randomly. The decision for an $\mathbf{x} \in \Re^n$ is made according to the class label of the cell indexed by $D_1(\mathbf{x}), \ldots, D_L(\mathbf{x})$.

2) Wernecke's Method: Wernecke's method [24] is similar to the BKS. The difference is that Wernecke considers the 95% CIs of the frequencies in each cell. If there is overlap between the intervals, a combination formula is applied instead of taking the label of the cell. The L confusion matrices are used to identify the "least wrong" classifier among the L members of the team, and that classifier's decision is taken as the label of the cell. As an illustration, assume that $L = 2$, $c = 3$, the confusion matrices are

$$CM^1 = \begin{bmatrix} 16 & \mathbf{6} & 10 \\ 12 & 20 & 10 \\ 7 & \mathbf{7} & 21 \end{bmatrix} \quad \text{and} \quad CM^2 = \begin{bmatrix} 13 & 9 & 10 \\ \mathbf{8} & 21 & 13 \\ \mathbf{7} & 6 & 22 \end{bmatrix}$$

and

$$DP(\mathbf{x}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Assume that the BKS cell indexed by $s_1 = \omega_2, s_2 = \omega_1$ contains six data points from ω_1, four from ω_2, and six from ω_3. As the CIs of the best two counts are identical, Wernecke selects a classifier from $\mathcal{D}$ to label of the BKS cell. First, each classifier D_i is examined with respect to the probability $P(error, D_i(\mathbf{x}) = s_i)$ by calculating the number of misclassifications when $D_i(\mathbf{x})$ labels $\mathbf{x}$ in s_i [see (15)]

$$M_i = \sum_{\substack{\mathbf{z}_j \in \mathbf{Z} \\ D_i(\mathbf{x}) = s_i}} 1 - y_{ji}. \tag{23}$$

The classifier with the smallest M_i labels the cell.

From the second column of CM^1 (because $s_1 = \omega_\mathbf{2}$) and the first column of CM^2 (because $s_2 = \omega_\mathbf{1}$), $M_1 = 6 + 7 = 13$, $M_2 = 8 + 7 = 15$. Since $M_1 < M_2$ we take $s_1 = \omega_2$ as the class label of the cell. Notice that only four points from ω_2 are present in the cell, compared to six from ω_1 and six from ω_3, and yet, the class label that the cell gets is ω_2.

REFERENCES

[1] M. E. Aladjem, "Combined discriminant analysis with binary features" (in Bulgarian), *Biocybernetics*, vol. 8, pp. 57–62, 1991.

[2] E. Alpaydin and M. I. Jordan, "Local linear perceptrons for classification," *IEEE Trans. Neural Networks*, vol. 7, pp. 788–792, May 1996.

[3] J. C. Bezdek, J. M. Keller, R. Krishnapuram, and N. R. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Norwell, MA: Kluwer, 1999.

[4] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proc. IEEE*, vol. 67, pp. 708–713, 1978.

[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[6] R. P. W. Duin, "A note on comparing classifiers," *Pattern Recognit. Lett.*, vol. 17, pp. 529–536, 1996.

[7] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 66–75, Jan. 1994.

[8] 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, Jan. 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] H.-J. Kang, K. Kim, and J. H. Kim, "A framework for probabilistic combination of multiple classifiers at an abstract level," *Eng. Applicat. Artif. Intell.*, vol. 10, no. 4, pp. 379–385, 1997.
[11] 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.
[12] L. I. Kuncheva, "Change-glasses approach in pattern recognition," *Pattern Recognit. Lett.*, vol. 14, pp. 619–623, 1993.
[13] ——, "Clustering-and-selection model for classifier combination," in *Proc. Knowledge-Based Intelligent Engineering Systems and Allied Technologies*, Brighton, U.K., 2000, pp. 185–188.
[14] ——, *Fuzzy Classifier Design*. Studies in Fuzziness and Soft Computing. Heidelberg, Germany: Springer-Verlag, 2000.
[15] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognit.*, vol. 34, no. 2, pp. 299–314, 2001.
[16] S. W. Looney, "A statistical technique for comparing the accuracies of several classifiers," *Pattern Recognit. Lett.*, vol. 8, pp. 5–9, 1988.
[17] K.-C. Ng and B. Abramson, "Consensus diagnosis: A simulation study," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 916–928, Sept./Oct. 1992.
[18] L. A. Rastrigin and R. H. Erenstein, *Method of Collective Recognition* (in Russian). Moscow, Russia: Energoizdat, 1981.
[19] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[20] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Trans. Neural Networks*, vol. 10, pp. 988–999, Nov. 1999.
[21] A. Verikas, A. Lipnickas, K. Malmqvist, M. Bacauskiene, and A. Gelzinis, "Soft combination of neural classifiers: A comparative study," *Pattern Recognit. Lett.*, vol. 20, pp. 429–444, 1999.
[22] W. Wang, P. Jones, and D. Partridge, "Diversity between neural networks and decision trees for building multiple classifier systems," in *Multiple Classifier Systems*. ser. Lecture Notes in Computer Science, J. Kittler and F. Roli, Eds. Cagliari, Italy: Springer, 2000, vol. 1857, pp. 240–249.
[23] K.-D. Wernecke, "On classification strategies in medical diagnostics (with special preference to mixed models)," in *Classification and Related Methods of Data Analysis*, H. H. Bock, Ed. Amsterdam, The Netherlands: Elsevier, 1988, pp. 299–306.
[24] ——, "A coupling procedure for discrimination of mixed data," *Biometrics*, vol. 48, pp. 497–506, 1992.
[25] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–260, 1992.
[26] 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, Apr. 1997.
[27] 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, May/June 1992.

Ludmila I. Kuncheva (M'99) received the M.Sc. degree from the Technical University, Sofia, Bulgaria, in 1982, and the Ph.D. degree from the Bulgarian Academy of Sciences in 1987.

Until 1997, she was with the Central Laboratory of Biomedical Engineering, Bulgarian Academy of Sciences, as a Senior Research Associate. Since 1997, she has been with the School of Informatics, University of Wales, Bangor, U.K., where she is currently a Senior Lecturer. Her interests include pattern recognition, classifier combination, diversity measures, fuzzy classifiers, and prototype classifiers.