Available online at www.sciencedirect.com

ScienceDirect

Neural Networks 19 (2006) 1506–1516

Neural Networks

www.elsevier.com/locate/neunet

# Trainable fusion rules. I. Large sample size case

Šarūnas Raudys*

*Institute of Mathematics and Informatics, Akademijos 4, Vilnius 08633, Lithuania*

Received 30 January 2003; accepted 2 January 2006

**Abstract**

A wide selection of standard statistical pattern classification algorithms can be applied as trainable fusion rules while designing neural network ensembles. A focus of the present two-part paper is finite sample effects: the complexity of base classifiers and fusion rules; the type of outputs provided by experts to the fusion rule; non-linearity of the fusion rule; degradation of experts and the fusion rule due to the lack of information in the design set; the adaptation of base classifiers to training set size, etc. In the first part of this paper, we consider arguments for utilizing continuous outputs of base classifiers versus categorical outputs and conclude: if one succeeds in having a small number of expert networks working perfectly in different parts of the input feature space, then crisp outputs may be preferable over continuous outputs. Afterwards, we oppose fixed fusion rules versus trainable ones and demonstrate situations where weighted average fusion can outperform simple average fusion. We present a review of statistical classification rules, paying special attention to these linear and non-linear rules, which are employed rarely but, according to our opinion, could be useful in neural network ensembles. We consider ideal and sample-based oracle decision rules and illustrate characteristic features of diverse fusion rules by considering an artificial two-dimensional (2D) example where the base classifiers perform well in different regions of input feature space.
© 2006 Elsevier Ltd. All rights reserved.

*Keywords:* Generalization error; Complexity; Learning set; Neural network ensembles; Multiple classifiers systems; Classifier combination; Fusion

## 1. Introduction

In the last decade, the theory of combination of ensembles of neural networks, cooperative neural-networks, *multiple classifier systems* (**MCS**) and related methods has been developed within many diverse research communities, including neural networks, machine learning, pattern recognition, and statistics (Kittler & Roli, 2000). In this paradigm, a number of simpler neural networks (classification algorithms) are designed at first. A "boss" decision rule aggregates outputs of the first-level decisions (base classifiers) and makes a final classification. I use the term "boss" (governor) in memory of Professor Leonard Rastrigin, who initiated the research direction "a collective of decision rules in pattern recognition" (Rastrigin & Erenstein, 1973, 1981). He used this term to name the fusion (combining) rule.

The neural network ensembles and multiple classifier systems (MCS) suggest an approach to design classification and prediction rules with a possibility of splitting the design process into separate parts. For different data sets, diverse subsets of features or types of algorithms, various computers could be used. Different designers' teams could make their independent contributions. At times, MCS convey decision making processes that are easier to interpret. Accordingly, MCS are helpful mathematical models while investigating collective decision making in human collectives. In addition, MCS, now and again, improve sample size/complexity relations. It is important if the sample size that is available for designing the classification rule is not large.

The first attempts to use MCS in order to organize the decision-making process in two stages began four decades ago (see e.g. Nilsson (1965, chap. 6), Rastrigin and Erenstein (1973), Telksnys (1968) and, for a forecasting task, a review by Clemen (1989)). The number of publications for the decision-making algorithm combination problem is approaching 1000. Useful reviews of different fusion rules can be found in Diettrich (2000), Gosh (2002), Hashem (1997), Haykin (1999), Kittler (1998), Kittler, Hatef, Duin, and Matas (1998), Kuncheva, Bezdek, and Duin (2001), Kuncheva (2004), Xu,

* Tel.: +370 5 2109 341; fax: +370 5 2729 209.
*E-mail address:* raudys@das.mii.lt.

0893-6080/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2006.01.018

Krzyzak, and Suen (1992) and a series of proceedings (Kittler & Roli, 2000, 2001, 2002, 2003, 2004). An impressive, almost encyclopaedic recent survey is Rahman and Fairhurst (2003), where almost 350 literature sources are reviewed.

Rahman and Fairhurst (2003) categorize MCS according to inherent design approach. They set apart: analytical methods, pseudo-analytical methods, empirical methods, neural network based methods, and support methods supporting decision combination from multiple independent sources. According to topologic information exchange pathways, they distinguish vertical, horizontal and hybrid combination schemes. According to the type of outputs provided by the base classifiers, they and Ho (2001) make a distinction between crisp, rank and confidence value (continuous) outputs. These *information management issues* are vital in classifying various multiple expert decision combination approaches.

One of the principal characteristics of any *pattern recognition* (**PR**) system is *generalization error*. Generalization error of the PR system depends on the complexity of the decision-making rule, the training set size, and a priori information incorporated into the algorithm. The MCS allow an organized complex two-stage decision-making process. Similarly to feature extraction, MCS simplify the decision-making process and allow us to introduce additional non-formalized information (Raudys, 2002). Therefore, the time came to look at MCS design from the generalization error point of view, and to elucidate complexity, prior assumptions and sample size relationships. Alas, except for the monograph of Kuncheva (2004), no broad systematic analysis of this sort has been performed in the literature. A useful mathematical technique for investigating the relationships mentioned is multivariate statistical analysis.

The size of data set available for designing the pattern recognition system is a main factor considered in this paper. For this reason, the complexity of the decision-making algorithm becomes a crucial factor. So we have split the paper into two parts. In the first part, we consider the complexity of algorithms and their characteristic features, assuming that sample size is large. We analyse arguments for utilizing continuous outputs of base classifiers versus categorical outputs at first (Section 2). In Section 3, we compare fixed fusion rules versus trainable rules, and present a review of trainable rules. We pay special attention to these linear and non-linear rules, which are employed rarely but, according to our opinion, could be useful in MCS design. In Section 4, we consider ideal and sample-based oracle decision rules. Section 5 illustrates characteristic features of diverse fusion rules by examining artificial an two-dimensional (2D) example where base classifiers perform well in different regions of input feature space. In the second part of the paper, small sample effects are analysed. Part of these effects arises only in two-stage decision-making algorithms.

Before starting, necessary *mathematical formalism* is introduced. We consider $K$ categories (pattern classes) and $L$ base experts. The outputs provided by the $j$th expert we denote by $o_{j1}, \dots, o_{jK}$. To make the final allocation of $p$-dimensional input vector $\boldsymbol{x} = (x_1, x_2, \dots, x_p)^{\mathrm{T}}$ to one of the categories, the fusion rule utilizes the $D_{\mathrm{MCS}} = L \times K$-dimensional vector $\boldsymbol{o} = (o_{11}, o_{12}, \dots, o_{1K}, o_{21}, o_{22}, \dots, o_{LK})^{\mathrm{T}}$. In some algorithms, the outputs of the base classifiers are normalized to satisfy the condition $\sum_{i=1}^{K} o_{ji} = 1$. Then dimensionality is $D_{\mathrm{MCS}} = L \times (K-1)$. Most often, we will concentrate on the two-category case ($K = 2$). If $K = 2$, the experts produce $L$ components, $o_1, o_2, \dots, o_L$, whether crisp (0 or 1) or continuous.

If one knows the distribution densities of the input vectors exactly, one may design an optimal classifier with minimal, *Bayes error*, $P_{\mathrm{B}}$. If one uses some simplified statistical model (say A) of input data, one obtains a classifier with asymptotic error, $P_\infty^{\mathrm{A}} \geq P_{\mathrm{B}}$. *Asymptotic classification error*, $P_\infty^{\mathrm{A}}$, is a hypothetical notion obtained under an assumption that training set sizes are infinitely large. If training data is used to design classifier A, its classification error, $P_n^{\mathrm{A}}$, is conditioned to a given training set of $N_1$ and $N_2$ samples. This is *conditional generalization error*. *Mean generalization error*, $EP_n^{\mathrm{A}}$, is an expectation of $P_n^{\mathrm{A}}$, taken over all possible training sets of $N_1$ samples from the first class and $N_2$ samples from the second one. The training set based estimate of the classification error is called an *apparent error rate*, $P_{\mathrm{app}}^{\mathrm{A}}$. Usually, $P_{\mathrm{app}}^{\mathrm{A}} < P_n^{\mathrm{A}}$ and $P_{\mathrm{app}}^{\mathrm{A}} < EP_n^{\mathrm{A}}$. For that reason, the experts' outputs calculated for the training set data are optimistically biased.

## 2. Continuous versus crisp outputs of the expert classifiers

An elucidation of factors that affect the choice between crisp or continuous outputs will be performed by means of the examination of 2D complex-shaped artificial two-category data. To generate the data, 2D Gaussian vectors were split into two pattern classes with narrow area between them. The classes can be separated by a palm-shaped decision boundary (see Fig. 1(a)). In principle, such a boundary can be realized by a *multilayer perceptron* (**MLP**) with nine hidden units. If the number of hidden units is small (nine to eleven), it is not easy to have faultless classification due to local minima problems.

Fig. 1(a) provides an example where the designer has two expert networks that perform well *in diverse parts* of the feature space separated by line $x_1 = 0.48$. Each decision boundary corresponds to a one-hidden-layer perceptron with $h = 7$ hidden units. The perceptron was trained, using subsets of vectors from diverse regions of the feature space. For such base classifier selection, an ideal "oracle" performs almost faultless classification. Recall that, if at least one of the classifiers produces the correct class label in a certain region of feature space, then the oracle produces the correct class label too.

Denote by $o_j = f(y_j)$, $(j = 1, \dots, L)$ the outputs of the MLP-based expert classifiers, where $y_j = \sum_{i=1}^{h} w_{ji} z_{si} + w_{j0}$ are weighted sums of outputs of the hidden layer neurons $(s = 1, \dots, h)$ and $f(y)$ is the non-linear activation function. In Fig. 2(a), we present a scatter diagram of the distribution of the weighted sums, $y_1$ and $y_2$. In the feature $y_1$ and $y_2$ space, both categories can be classified almost without errors by means of a piecewise linear fusion rule composed of two perpendicular lines. If we use crisp outputs (0 and 1) of the expert networks, a good decision boundary can be obtained by a weighted sum rule that makes decisions according to the sign of a simple sum,

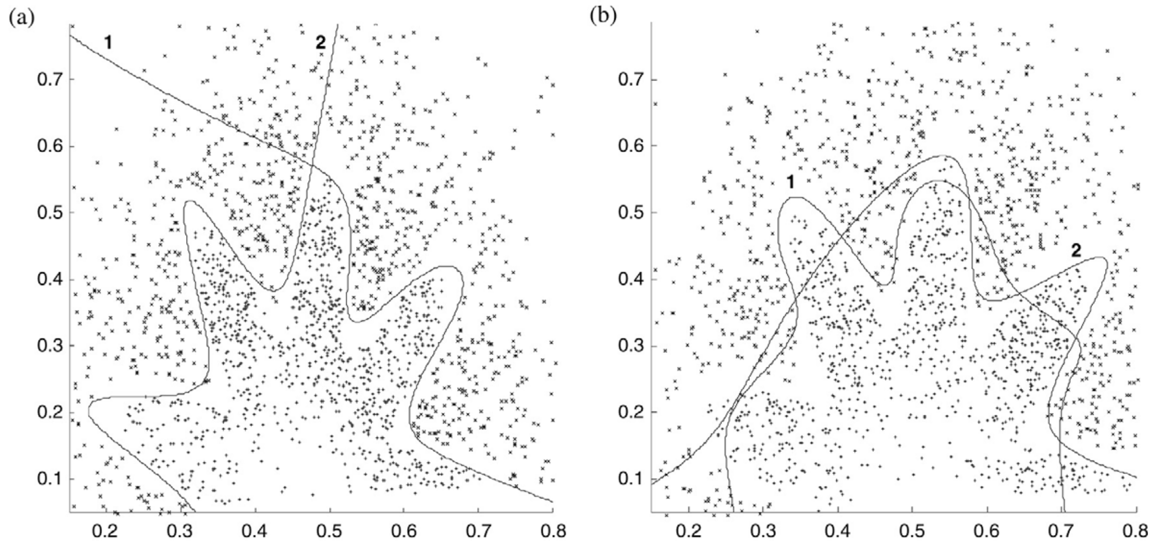

Fig. 1. 2000 2D training points and decision boundaries of two MLP-based base classifiers: (a) experts are perfect for distinct parts of feature space; (b) non-ideal experts.

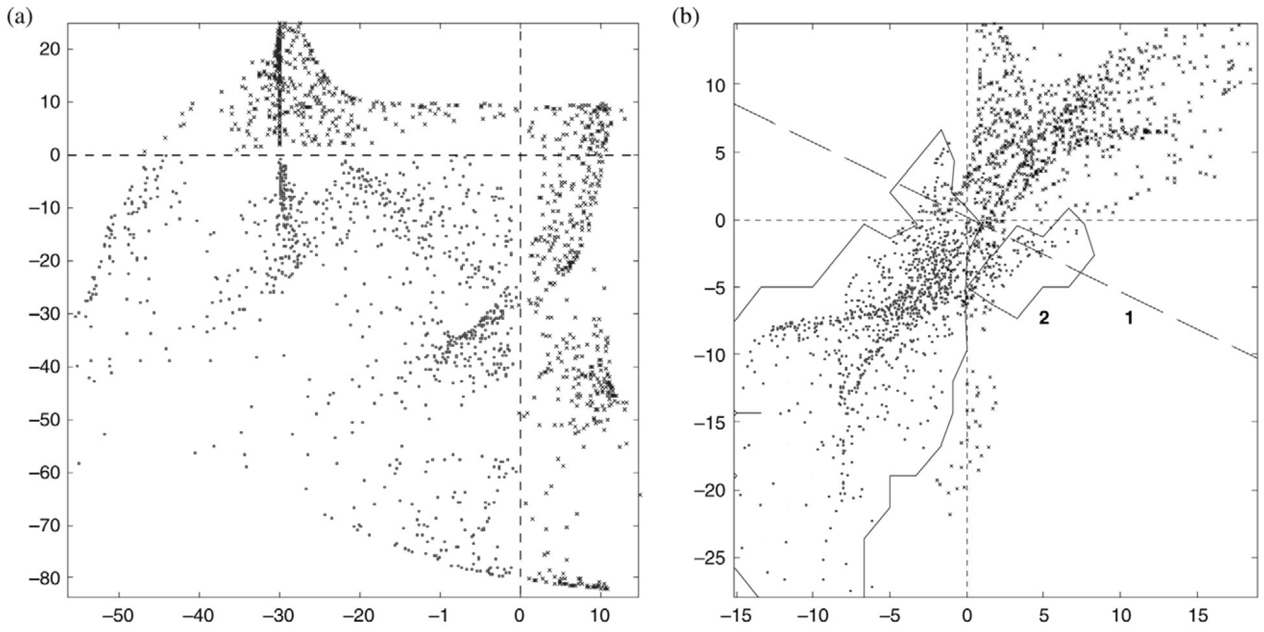

Fig. 2. The distributions of the weighted sums, $y_1$, $y_2$: (a) "perfect" experts; (b) non-ideal experts. Decision boundaries: 1—of the SLP used as a fusion rule; 2—the KDA classifier.

$g(y_1, y_2) = y_1 + y_2 - 0.5$. This example gives a strong reason that, in the case of *perfect experts* in separate areas of the feature space, the utilization of crisp outputs could appear useful. In Part II of this paper, we will dispute that a large number of base classifiers and small sample size could disagree with the utilization of crisp outputs.

In the second example (Fig. 1(b)), we have decision boundaries of two *imperfect* base classifiers (MLP seven hidden units). 1000 vectors of each category were used to train the MLP classifier. In Fig. 2(b), we present a scatter diagram of 2D vectors, $y_1$, $y_2$. In order to obtain good separation in feature $y_1$ and $y_2$ space, we have to create a complex non-linear decision boundary. In Fig. 2(b), we depict such a boundary, which was realized by means of *Kernel Discirminant Analysis* (**KDA**; see e.g. Duda, Hart, and Stork (2000), Fukunaga (1990), or Section 3.2 of this paper). Linear separation realized by *single layer perceptron* (**SLP**) results gave worse discrimination of the pattern classes.

Complexity of the decision boundary in the features $y_1$ and $y_2$ space emphasizes that, in the present example, obtaining the optimal fusion rule is almost as difficult as in the original feature $x_1$ and $x_2$ space. Thus, one can deduce that:

*If we have experts, good in separate parts of the feature space, then we may use crisp outputs*. In such situations, the task of the boss (the fusion rule designer) is rather simple. *If we have weak experts, continuous outputs have to be* *used*. Then all the difficulty of the classification problem is transferred to the boss (fusion rule).

The intermediate case between continuous and crisp outputs arises if one utilizes non-linearly transformed variables, e.g. $o_1 = f(y_1) = 1/(1+\exp(-y_1))$, $o_2 = f(y_2) = 1/(1+\exp(-y_2))$. As *a generalization of this situation*, one can consider generalized outputs

$$o_j = 1/(1+\exp(-\gamma y_j)), \tag{1}$$

which are distinguished by different positive $\gamma$. For very small $\gamma$, one has almost linear transformation of the variables $y_1$ and $y_2$. In such a case, final classification is performed in continuous feature space. For very large $\gamma$, one obtains crisp outputs. As a result, *coefficient $\gamma$ controls linearity of the expert output transformation*. Therefore, *coefficient $\gamma$ influences the complexity of the fusion rule and affects its generalization error*.

## 3. Fixed and trainable fusion rules

### 3.1. Linear rules

An attractive feature of MCS is the possibility of using fixed non-trainable fusion rules: sum, product rules, etc. Such rules are based on quite strong assumptions about the similar performance of all expert classifiers, and on the independence of the base classifiers (Fumera & Roli, 2005; Kittler, 1998; Kuncheva, 2004). The simple average (fixed sum) and the product rules have common roots, since the sum rule is a variation of the product (average) rule if logarithms of the experts' outputs were considered. In the two-category case, the "best expert" rule and the majority voting rule could also be considered as the *simple average* (hereafter, **SA**) rules. Simple average rules give the best results in many applications. In situations where experts' outputs are statistically dependent, qualifications of experts are different and sample size is large, trainable linear rules like *weighted average* (**WA**) or weighted voting could become preferable. The weighted average is a generalization of the simple average rule. In the two-category case, allocation of the vector to be classified is performed according to the sign of a *discriminant function* (**DF**):

$$g(\boldsymbol{o}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{o} + w_0 \tag{2}$$

where $\boldsymbol{w} = (w_1, w_1, \ldots, w_L)$ is an $L$-dimensional weight vector.

In the majority of known fusion rules, the weights are positive (see e.g. Ramachandran, Farrell, & Mammone, 2002). Standard statistical classifiers, however, could result in positive and negative weights.

**Euclidean distance (nearest means) classifier (EDC).** This is one of the simplest statistical allocation rules. A membership of vector $\boldsymbol{o}$ is determined according to the distance to sample estimates of mean vectors of the first and second class, $\bar{\boldsymbol{o}}^{(1)}$ and $\bar{\boldsymbol{o}}^{(2)}$. Classification is performed according to the sign of DF (2). In EDC, $\boldsymbol{w} = \bar{\boldsymbol{o}}^{(1)} - \bar{\boldsymbol{o}}^{(2)}$ and $w_0 = -1/2(\bar{\boldsymbol{o}}^{(1)} + \bar{\boldsymbol{o}}^{(2)})^{\mathrm{T}}\boldsymbol{w}$. In the decision templates method, to fuse base classifiers (Kuncheva et al., 2001), standard output patterns (templates) were found for each pattern class. The unknown pattern vector is classified according to its similarity to a particular template. Decision making is very similar to that performed by the Euclidean distance classifier.

**Fisher linear discriminant function.** This rule is used if expert outputs are statistically dependent. Its weight vector is $\boldsymbol{w}_{\mathrm{F}} = S_o^{-1}(\bar{\boldsymbol{o}}^{(1)} - \bar{\boldsymbol{o}}^{(2)})$ and its bias term is $w_0 = -1/2(\bar{\boldsymbol{o}}^{(1)} + \bar{\boldsymbol{o}}^{(2)})^{\mathrm{T}}\boldsymbol{w}_{\mathrm{F}}$, where $\mathbf{S}_o$ is a sample estimate of the $L \times L$ covariance matrix $\boldsymbol{\Sigma}_o = ((\sigma_{ij}))$ supposed to be common for both pattern classes (Duda et al., 2000; Fukunaga, 1990). If the expert outputs are correlated, some of the components of vector $\boldsymbol{w}_{\mathrm{F}}$ could become negative. We will show that ***the introduction of negative weights can reduce the generalization error of MCS dramatically***.

Consider the case where the expert output vectors $\boldsymbol{o}$ are distributed according to the multivariate Gaussian law, both pattern classes share common covariance matrix $\boldsymbol{\Sigma}_o = ((\sigma_{ij}))$, and $q_2 = q_1 = 1/2$. For this data model, it is possible to formulate an elegant theory for the simple average and weighted average rules. In the Gaussian data case, standard statistical theory gives asymptotic classification error of the Fisher linear DF used as the WA rule:

$$P_\infty^{\mathrm{WA}} = \Phi\{-1/2\delta\}, \tag{3}$$

where $\Phi\{a\} = \int_{-\infty}^{a}(2\pi)^{-1/2}\sigma^{-1}\exp\{-t^2/(2\sigma^2)\}\mathrm{d}t$ is the cumulative distribution function of the Gaussian N(0, 1) random variable, $\delta^2 = (\delta^{(1)}, \ldots, \delta^{(D)})\boldsymbol{\Sigma}_o^{-1}(\delta^{(1)}, \ldots, \delta^{(D)})^{\mathrm{T}}$ is the squared Mahalanobis distance, and $\delta^{(j)}$ is the distance between expected values of outputs of the first class and the second class, corresponding to the $j$th base classifier.

The weight vector of SA and majority voting combining rules is $\boldsymbol{w} = [1\ 1, \ldots, 1]^{\mathrm{T}}$. The final allocation of the vector $\boldsymbol{o}$ is performed according to the sum $g^{\mathrm{SA}}(\boldsymbol{o}) = \sum_{j=1}^{L} o_j$. After calculating the mean and variance of $g^{\mathrm{SA}}(\boldsymbol{o})$, similarly to Eq. (3), we have

$$P_\infty^{\mathrm{SA}} = \Phi\left\{-1/2\sum_{j=1}^{L}\delta^{(j)} \Big/ \sqrt{\sum_{i=1}^{L}\sum_{j=1}^{L}\sigma_{ij}}\right\}. \tag{4}$$

If vector $(\delta^{(1)}\delta^{(2)}, \ldots, \delta^{(D)})$ and matrix $\boldsymbol{\Sigma}_o$ are known, one can calculate the classification errors of SA and WA (Fisher classifier) fusion rules and compare them numerically. Visibly, the optimal weighted sum should outperform the non-optimal simple sum rule if the parameters of the weighting rule are determined exactly.

**A toy example**. Consider MCS with two experts that give continuous, highly correlated ($\rho = 0.999$) Gaussian outputs with unit variances and mean values in the first and second classes, 0.25, −0.25 (Expert 1) and 0.125, −0.125 (Expert 2). Then $\delta^{(1)} = 0.5$, $\delta^{(2)} = 0.25$ and $P_\infty^{\text{Expert 1}} = \Phi\{-0.25\} = 0.4$, $P_\infty^{\text{Expert 2}} = 0.45$. Calculation according to Eqs. (3) and (4) gives: $P_\infty^{\mathrm{SA}} = 0.425$ and $P_\infty^{\mathrm{WA}} = 0.0025$. We see that the weighted average outperforms the fixed sum rule by 170 times in this example.

**Example with more realistic correlations.** In data model ($\mathbf{W}\boldsymbol{\Sigma}_5$), we have *five equally good experts*, i.e. each expert makes 8% error ($\delta^{(1)} = \delta^{(2)} = \cdots = \delta^{(5)} = 2.81$). The correlations between the experts' outputs and the components of the WA fusion rule are:

$$\mathbf{\Sigma} = \begin{bmatrix} 1.0000 & 0.7247 & 0.5889 & 0.3928 & 0.7103 \\ 0.7247 & 1.0000 & 0.3351 & 0.5703 & 0.4501 \\ 0.5889 & 0.3351 & 1.0000 & 0.6707 & 0.3954 \\ 0.3928 & 0.5703 & 0.6707 & 1.0000 & 0.5748 \\ 0.7103 & 0.4501 & 0.3954 & 0.5748 & 1.0000 \end{bmatrix} \quad (5)$$

$$\mathbf{w}^F = \begin{bmatrix} -36.60 & 26.54 & 25.21 & -28.10 & 23.05 \end{bmatrix}^{\mathrm{T}}. \quad (6)$$

This specific pattern of the correlation matrix makes the fusion weights remarkably different to the weights of the fixed sum rule. Two weights are negative. Eqs. (3) and (4) give: $P_\infty^{\mathrm{SA}} = 0.039$ and $P_\infty^{\mathrm{WA}} = 0.0039$ (ten times smaller!).

**Regularized discriminant analysis (RDA).** In small-sample-size and high-dimensionality cases, certain simplifications of covariance matrix estimates have to be applied. Most popular is an intermediate case between EDC and Fisher DF. Here, one adds a positive constant, $\lambda$ (regularization parameter), to all diagonal elements of sample covariance matrix $S_o$: $\mathbf{S}_{\mathrm{RDA}} = \mathbf{S}_o + \lambda\mathbf{I}$, where $\mathbf{I}$ is the $p \times p$ identity matrix. In the RDA, we reduce sample estimates of the correlations artificially. The RDA technique and the weight decay term traditionally utilized in perceptron training are of the same origin: both of them perform regularization of the sample covariance matrix. In each concrete application, the optimal $\lambda$ value depends on sample size and has to be selected by a trial-and-error method.

**More linear statistical classifiers** could be utilized as trainable fusion rules. We mention here only a few of them. While designing a ***minimum empirical error classifier***, one does not need to assume that the expert's output vector, $\boldsymbol{o}$, follows a multivariate Gaussian law. A criterion to be minimized is the number of training vectors that are misclassified (Raudys, 2001b; see also Ueda (2000)). An important group of classification rules is the maximal margin classifiers, currently called ***support vector machines*** (Vapnik, 1995).

In the two-category case, the ***single-layer perceptron*** can also be used to build the fusion rule. When certain conditions are satisfied, after the first training iteration one obtains the EDC. With an increase in the number of iterations, one moves towards more complex decision rules: RDA, standard Fisher rule, robust linear classifier, minimum empirical error, and support vector classifiers (Raudys, 1998, 2001b). The number of possible classifiers could be increased and the convergence time could be reduced if, prior to training the perceptron, one performs data a whitening linear transformation (Raudys, 2001b, chap. 5). Linear classifiers can also be used to combine crisp binary outputs of base experts.

### *3.2. Non-linear rules*

In continuous expert output space, nonlinear decision boundaries could be realized by linear classifiers (EDC, Fisher DF, SPL and their modifications) if the decision were performed in new artificially generated non-linear space. Polynomial features, radial basic function (RBF) features, etc. are popular. To obtain a non-linear decision boundary, a quadratic discriminant function (Duda et al., 2000; Friedman, 1989; Fukunaga, 1990) could also be applied. Provided that the expert outputs are transformed into binary (0, 1) feature space, non-linear decision boundaries could be designed for crisp outputs too. Popular techniques for designing non-linear trainable fusion rules are multilayer perceptron and radial basic function networks (Haykin, 1999).

**The *k*-nearest neighbors (*k*-NN) rule**. In Fig. 2(b) we were obliged to use a complex-shaped non-linear decision boundary in order to obtain good separation of pattern classes. Popular methods for designing non-linear local statistical classifiers include the $k$-nearest neighbors rule and Kernel Discriminant Analysis, often called Parzen window classifiers. In both approaches, we have to store all training vectors. In the $k$-NN rule, for each vector $\boldsymbol{o}$ to be classified, one calculates the distances (similarity measures) to all training pattern vectors and seeks $k$ nearest neighbors. Vector $\boldsymbol{o}$ is classified according to the majority of class labels in a subset of $k$ distances. Modifications of this rule differ in the definitions of the distance. Two examples: Euclidean distance between $\boldsymbol{o}_r$ and $\boldsymbol{o}_l^{(s)}$, $D_{\mathrm{E}}(\boldsymbol{o}_r, \boldsymbol{o}_l^{(s)}) = (\boldsymbol{o}_r - \boldsymbol{o}_l^{(s)})^{\mathrm{T}}(\boldsymbol{o}_r - \boldsymbol{o}_l^{(s)})$; Mahalanobis distance $D_{\mathrm{M}}(\boldsymbol{o}_r, \boldsymbol{o}_l^{(s)}) = (\boldsymbol{o}_r - \boldsymbol{o}_l^{(s)})^{\mathrm{T}}\mathbf{A}(\boldsymbol{o}_r - \boldsymbol{o}_l^{(s)})$, where matrix $\mathbf{A}$ is usually an inverse of sample covariance matrix $\mathbf{S}_o$. Sometimes, instead of $\mathbf{A}$, one uses the inverse of the diagonal matrix composed only of variances. Utilization of the Mahalanobis distance is recommended if components of the vector $\boldsymbol{o}$ have different variances, are highly correlated, and the sample size used to obtain the estimate $\mathbf{S}_o$ is sufficiently large. When designing non-parametric fusion rules in discrete space, one can use such distances as the number of disagreements between $\boldsymbol{o}_r$ and $\boldsymbol{o}_l^{(s)}$ (Hamming distance). While applying the $k$-NN classifier for the fusion rule design, one ought to keep in mind that, for categorical valued vectors, it is frequently impossible to obtain exactly $k$ nearest neighbors.

**Kernel discriminant analysis (KDA).** To get the decision rule, one uses non-parametric estimates of the conditional densities of each pattern class, and relies on the theory of statistical decision functions. Let $q_s$ be the prior probability of class $\Pi_s$ and $N_s$ be the number of training vectors from $\Pi_s$. Then the conditional probability density estimate is

$$\hat{p}_s(\boldsymbol{o}) = \frac{1}{N_s}\sum_{l=1}^{N_s} \boldsymbol{\kappa}\{D(\boldsymbol{o}, \boldsymbol{o}_l^{(s)})/\lambda^2\}, \quad (s = 1, 2, \ldots, K), \quad (7)$$

where $\lambda$ is a smoothing constant that should be selected in accordance with the training set size and data configuration, and $\boldsymbol{\kappa}(D)$ is a decreasing function of $D$. We used $\boldsymbol{\kappa}(D) = \text{const} \times \exp(-D)$.

A theory of statistical decision functions gives the following classification rule: vector $\boldsymbol{o}$ is classified according to the maximum of the products $q_1\hat{p}_1(\boldsymbol{o}), q_2\hat{p}_2(\boldsymbol{o}), \ldots, q_K\hat{p}_K(\boldsymbol{o})$. This rule can be used both for continuous and categorical outputs of the expert classifiers. In applications, the values of smoothing parameters of the $k$-NN and KDA classifiers, $k$ and $\lambda$, are very important. If $k = 1$ or $\lambda \to 0$, we have no

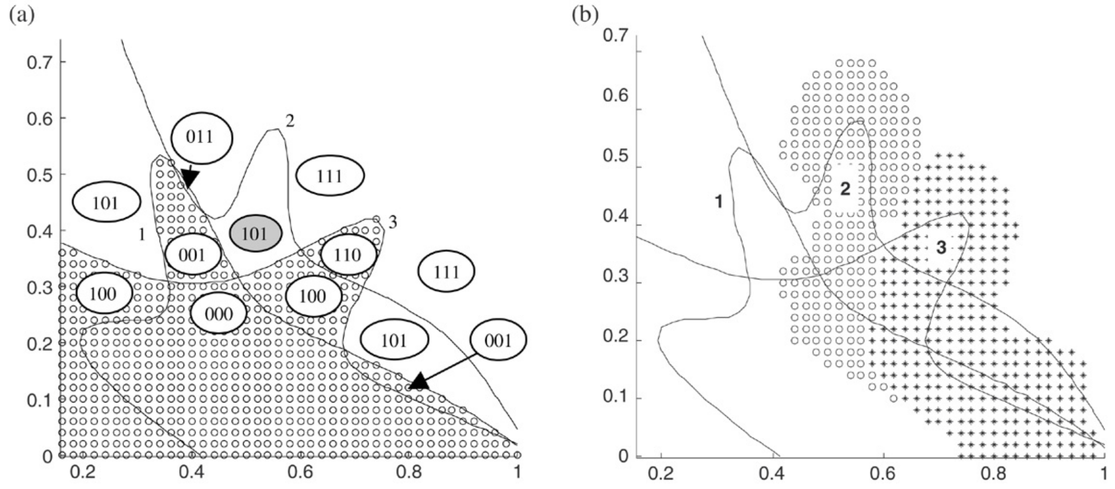

Fig. 3. (a) Decision boundaries (1, 2, 3) of three MLP classifiers (we present binary codes of the base experts' outputs inside ellipses). (b) Competence areas for three expert classifiers (1—non-marked, 2—circle marked and 3—rhombus marked).

smoothing. With an increase in $k$ or $\lambda$, smoothing increases. Optimal values of $k$ or $\lambda$ depend on the training set size, the complexity of the data, and the complexity of the decision boundary required to obtain good classification. Usually, we use sizeable $k$ and smaller $\lambda$ in larger sample situations.

In order to find the optimal value, $\lambda_{\text{opt}}$, for a particular problem, we recommend evaluating the classifiers' performance for several values of $\lambda$ and choosing the value that provides the best performance. When the variances of all $p$ features differ significantly and one uses the same $\lambda$ value for all features, then it is often necessary to normalize the features prior to using the classifier. In such a case, $\lambda_{\text{opt}}$ often falls in the interval (0.01, 10). In most practical problems, the valley of the dependence of generalization error on parameter $\lambda$ is rather flat. Thus, a set of ten values, 0.001, 0.01, 0.03, 0.1, 0.3 1, 3, 10, 100, 1000, is often sufficient to determine $\lambda_{\text{opt}}$ empirically.

Evaluation of the classification errors for ten different values of $\lambda$ can be computationally demanding. In high-dimensional problems, most of the computation time is spent in calculating distances $\boldsymbol{D}(\boldsymbol{o}, \boldsymbol{o}_l^{(s)})/\lambda^2$ between vector $\boldsymbol{o}$ and the $l$th training vector of the $s$th class. To conserve computer time, we recommend that all ten error rates be estimated simultaneously. After finding distance $\boldsymbol{D}(\boldsymbol{o}, \boldsymbol{o}_l^{(s)})$, we calculate the ten terms $\boldsymbol{D}(\boldsymbol{o}, \boldsymbol{o}_l^{(s)})/\lambda_r^2\ r = 1, \ldots, 10$ corresponding to each value of $\lambda_r$ and classify the vector $\boldsymbol{o}$ for values of $\lambda_r$. The computer time required to obtain ten simulation estimates of error rates is considerably smaller that the time for ten independently obtained estimates (Raudys, 1991; Raudys & Jain, 1991).

**Behaviour-knowledge space (BKS) method.** This is a popular non-linear classifier combination method (Huang & Suen, 1995) designed for work with crisp outputs. In statistical pattern recognition literature, it is known as a ***multinomial classifier*** (Cochran & Hopkins, 1961; Lachenbruch & Goldstein, 1979; Linhart, 1959). Assume that we have $L$ expert classifiers, where the $j$th expert classifies unknown input vector $\boldsymbol{x}$ into one of the $K_j$ classes (states). For the sake of generality, we supposed that the various expert classifiers can be good specialists in classifying into a *different* number of classes. In standard situations, $Kj = K$. Denote the decision made by the $j$th expert by $o_j$. The total number of possible combinations (states, cells) of $L$ crisp outputs $o_1, o_2, \ldots, o_L$ is $m = \prod_{j=1}^{L} K_j$. Each vector, $\boldsymbol{o}$, can assume only one state, $s_r$, out of the $m$ possible states, $s_1, s_2, \ldots, s_m$. To design a fusion rule, we take into account that $\boldsymbol{o} = (o_1, o_2, \ldots, o_L)^{\text{T}}$ is a discrete valued random vector. Then the conditional distribution of the $s$th class vector $\boldsymbol{o}$ is characterised by $m$ probabilities

$$P_1^{(s)}, P_2^{(s)}, \ldots, P_{m-1}^{(s)}, P_m^{(s)},$$
$$\text{with} \sum_{r=1}^{m} P_r^{(s)} = 1, (s = 1, 2, \ldots, K). \tag{8}$$

The *Bayes rule* makes the final classification according to the maximal value of the products

$$q_1 P_r^{(1)}, q_2 P_r^{(2)}, \ldots, q_K P_r^{(K)}. \tag{9}$$

To design a BKS classifier, we have to know *all* probabilities $P_1^{(1)}, \ldots, P_m^{(K)}$. Let the fusion rule provide its solution only on the basis of the class labels and probabilities $P_1^{(1)}, \ldots, P_m^{(K)}$ be known. Then the BKS rule is the ***optimal (Bayes) fusion rule***.

In general, the multinomial classifier-based fusion rule will fail against ideal ***oracle***, the fusion rule that utilizes *certain additional information* (vector $\boldsymbol{x}$). We have such an example in Fig. 3(a). Here we see decision boundaries (1, 2, 3) of three MLP ($h = 7$) base classifiers that perform very well in different areas of the input feature space. Three MLPs were trained by vectors of different compact areas of feature space divided by lines $x_1$ = constant. Each base classifier is a good expert in his own "area of competence". None of them, however, is a good classifier in the entire input feature space: they produce 22.5%, 22.5% and 24.3% classification errors, respectively. In the example with the palm data and three base experts, the ideal oracle can classify both training and test data vectors without error.

The multinomial classifier-based fusion rule is useful: it outperforms the best individual expert. Unfortunately, *complex,* *non-linear expert classifiers can form cells where vectors that fall into one single cell can be distant in the original feature space*. In Fig. 3(a), we have three such cells, the 001th, 100th and 101th. The vectors in these cells, belong to distinct classes. For that reason, the multinomial rule with three experts makes a large number of classification errors (16.75%). In the next section, we will have an example where four additional expert MLP classifiers are added. Then the BKS fusion rule generalizes without error.

In practice, $K \times (m-1)$ probabilities $P_1^{(1)}, P_2^{(1)}, \ldots, P_m^{(K)}$ are unknown and have to be estimated from training data. One can use the maximum likelihood estimates of prior probabilities, $q_s$, and the probabilities of the cells, $P_r^{(s)}$, $\hat{q}_s = N_s / \sum_{s=1}^{K} N_s$, $\hat{P}_r^{(s)} = n_r^{(s)}/N_s$, where $n_r^{(s)}$ is a number of training vectors from $\Pi_s$ in the $r$th cell. In such a case, the allocation of vector $\boldsymbol{o}$ is performed according to the majority of training vectors in the $r$th cell. Consequently, we have the ***sample-based multinomial classifier***. If the number of training vectors in some of the cells is equal to zero, a useful alternative way to estimate the unknown probabilities, $P_r^{(s)}$, is the Bayes predictive approach (see e.g. Fukunaga (1990) and Raudys (2001b)). For uninformative *uniform* prior distribution of the cell's probabilities $P_1^{(s)}, P_2^{(s)}, \ldots, P_{m-1}^{(s)}, P_m^{(s)}$, the predictive Bayes methodology gives $\hat{P}_{r\,\text{Bayes}}^{(s)} = (\hat{P}_r^{(s)} + 1)/(N_s + m)$.

**Three main drawbacks of the sample-based multinomial classifier**. ***First***, the vectors of one single cell can be distant in the original feature space and belong to diverse categories. An example we have had in Fig. 3(a): vectors in cells 001, 100 and 101 belong to two classes.

***Second***, difficulty arises when we have many experts and many pattern classes. Then the number of cells, $m$, becomes extremely large. For example, in the character classification problem, with $2 \times 26 = 52$ pattern classes and 5 experts, $m = 52^5 = 380{,}204{,}032$. It is difficult to realize such a classification rule and estimate 760,408,062 probabilities.

***Third***, difficulties arise if we take into account that the number of training vectors used to estimate the cells' probabilities $P_1^{(s)}, P_2^{(s)}, \ldots, P_{m-1}^{(s)}, P_m^{(s)}$ ($s = 1, 2, \ldots, K$), is finite. Most of the states will be empty! Empty cells can be assigned to the class with the highest prior probability. If the prior probabilities are equal, the assignment can be arbitrary. In this situation, one can use a number of existing Boolean algebra or combinatory techniques in order to simplify the "decision-making path". To estimate cells' probabilities, Kang (2003) utilized the second- and third-order dependence tree models. Güler, Sankur, Kahya, Skurichina, and Raudys (1996), Janeliunas and Raudys (2002), Raudys (2003) and Raudys and Roli (2003) added a noise to learning data. They trained the BKS rule on an artificially created training set. In such a way, small sample properties of the fusion rule were improved.

**Decision tree classifier** (Breiman, Friedman, Olshen, & Stone, 1984; Lbov, 1981; Quinlan, 1993) is one more possibility for building a simpler allocation rule. Here, the decision-making process is represented as a tree with a relatively small number of final leaves $m_{\text{final}}$. To simplify the decision-making path, we merge final leaves and "branches" with identical classifications. Fig. 3(a) shows that cell 011 is empty and its class membership could be arbitrary. In cell 101, we have training vectors of both classes. Vectors in this cell will be allocated to class $\Pi_1$. So, in the decision tree classifier, the third expert will perform classification at first and will assign class number $\Pi_1$ if $o_3 = 1$. If $o_3 = 0$, the first expert will perform classification after that. The decision tree will assign class number $\Pi_0$ if $o_1 = 0$, and $\Pi_2$ if $o_1 = 1$. In such a decision-making rule, we have three final leaves instead of eight ones in the multinomial classifier. From the point of view of learning data sets, conversion of a multinomial classifier into a simple decision tree does not change the decision-making rule. Inaccuracies while estimating the cells' probabilities do not vanish. So, both algorithms have the same sample size/complexity properties. In the finite training set case, undemanding advice to simplify the allocation rule is: use a decision tree classifier with a smaller number of final leaves and allow a higher apparent error rate.

## 3.3. Sample-based oracles

The *oracle* assigns decision making to one of the experts. It is a hypothetical fusion rule that utilizes crisp outputs of experts and additional information, the input vector $\boldsymbol{x}$. If at least one of the classifiers produces the correct class label, then the oracle produces the correct class label too (Kuncheva, 2004). In some sense, the oracle evaluates an excellence of a set of several base classifiers. Therefore, it is usually used in comparative experiments.

It is worth noting that classification error of the oracle classifier, although sometimes useful for comparisons, may also contain no or little information on the combined classifier. A simple example is MCS of two base classifiers, one always outputting class 1, and the other one outputting class 2. Then the oracle classifier has a 100% recognition rate.

The example presented in Fig. 3(a) was a good illustration of the non-optimality of the multinomial fusion rule. Vectors of the 101th cell belong to both categories and the multinomial rule, in principle, cannot resolve this dilemma. In such a situation, the oracle, however, can perform ideal classification. Several versions of sample-based oracles have been suggested.

**Use of the input vector to design the oracle.** In the Rastrigin and Erenstein (1973, 1981) approach, the sample-based oracle (the boss, governor) uses additional information, the input vector $\boldsymbol{x}$, in order to decide which expert is the most competent to classify this particular vector, $\boldsymbol{x}$. Thus, ***the boss's fusion rule*** allocates vector $\boldsymbol{x}$ to one of $L$ ***virtual pattern classes*** (recall that $L$ is the number of expert classifiers). The $j$th expert's competence is estimated as a "potential"

$$\hat{p}_j(\boldsymbol{x}) = \sum_{s=1}^{K} \sum_{l=1}^{N_s} q_{jl}^{s} \boldsymbol{\kappa}\{D(\boldsymbol{x}, \boldsymbol{x}_l^{(s)})/\lambda^2\}, \quad (j = 1, 2, \ldots, L), \tag{10}$$

where $q_{jl}^{s} = 1$ if training vector $\boldsymbol{x}_l^{(s)}$ was classified by the $j$th expert correctly, and $\boldsymbol{x}_l^{(s)} = -1$ if the vector $\boldsymbol{x}_l^{(s)}$ was

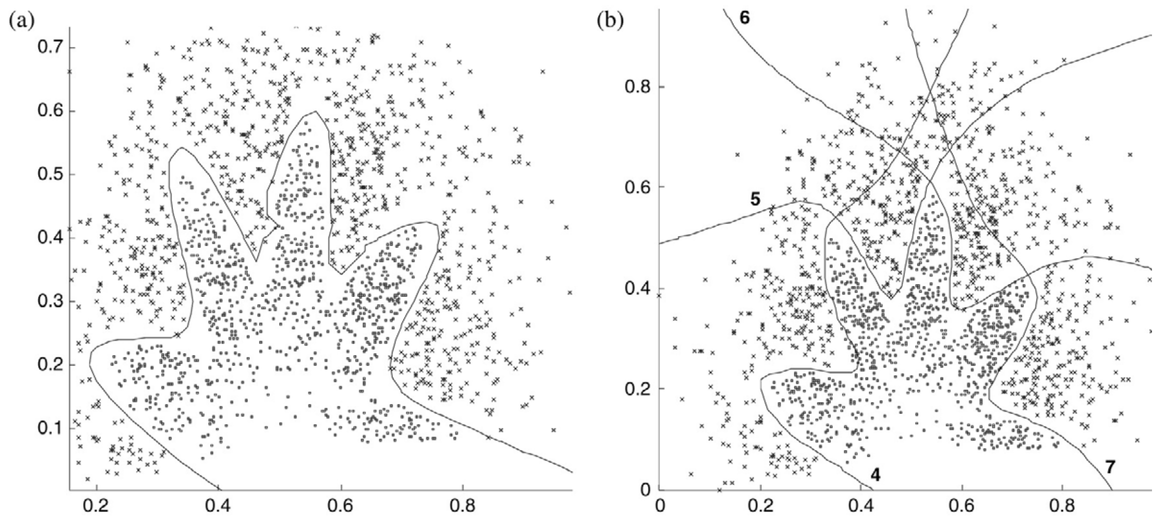

Fig. 4. (a) Decision boundary of sample-based oracle; (b) decision boundaries (4, 5, 6, 7) of four MLP ($h = 7$) classifiers that are perfect experts in distinct compact areas of the input feature space.

classified incorrectly, $D(\mathbf{x}, \boldsymbol{x}_l^{(s)})$ is the distance between vectors $\boldsymbol{x}$ and $\boldsymbol{x}_l^{(s)}$, $\boldsymbol{\kappa}(D)$ is a kernel or a smoothing function, and $\lambda$ is a smoothing constant.

In fusion, the expert with maximal sum, $\hat{p}_j(\boldsymbol{x})$, performs the classification of vector $\boldsymbol{x}$. To find the most competent expert, Rastrigin and Erenstein (1981) utilized a potential function classifier. The potential function classifiers, in effect, are very similar to the well-known and popular non-parametric Kernel Discriminant Analysis discussed in Section 3.2. In Fig. 3(b), we have depicted 'competence areas' for the three expert classifiers shown in Fig. 3(a). As the kernel, the Gaussian window (Eq. (7)) with $\mathbf{A} = \mathbf{I}$ and $\lambda = 0.01$ was used.

The non-linear decision boundary of the ensemble of three MLP-based base classifiers with the sample-based non-parametric oracle is depicted in Fig. 4(a). We see that, with 1000 training samples from one category, the fusion rule based on the KDA approach allows obtaining a perfect sample-based oracle (almost faultless classification of the test set). Recall that, in classifying test set vectors, the multinomial rule performed much worse, with 16.75% error.

An alternative way to KDA approach in experts' fusion is to use a $k$-nearest neighbour classifier (Sabourin, Mitiche, Thomas, & Nagy, 1993; Woods, Kegelmeyer, & Bowyer, 1997). In the high-dimensional space and finite design set situations, however, both the KDA and $k$-nearest neighbour estimates become inaccurate (Raudys (1991, 2001b); see also Part II of this paper).

**Use of the experts' outputs to design the sample-based oracle.** Giacinto, Roli, and Fumera (2000) proposed the sample-based oracle called *dynamic expert selection*. While classifying vector $\boldsymbol{o}$, they used the classifiers' outputs to evaluate the experts' accuracy and selected the most competent one to classify vector $\boldsymbol{o}$. If the $L$ experts give crisp outputs $o_1, o_2, \ldots, o_L$, then the classifier accuracy is estimated as the fraction of training (or validation) patterns with the same combination of outputs, $o_1, o_2, \ldots, o_L$. In such a case, the sample-based oracle is implemented as the BKS (multinomial) classifier. We recall that the total number of combinations is $m = \prod_{j=1}^{L} mj$. If, for input vector $\boldsymbol{x}$, the $L$ experts provide continuous outputs, $o_1, o_2, \ldots, o_L$, then a special calculation schema was suggested in order to evaluate the classifiers' accuracy and select the most competent of them. This schema is similar to the non-parametric KDA approach.

Differently from the Rastrigin and Erenstein methodology, in order to design the sample-based oracle, rather than the input vector $\boldsymbol{x}$, the experts' outputs, $\boldsymbol{o} = (o_1, o_2, \ldots, o_L)^{\mathrm{T}}$, were used. *To reduce dimensionality*, *only experts with highest $o_j$ were considered*. The fusion rule became faster and acquired enhanced small sample properties. Giacinto and Roli (2001) also proposed a modification of a sample-based oracle that uses both the input vector and the experts' outputs in order to perform dynamic classifier selection. First, in the training data, one finds the $k$-nearest neighbours to the input pattern vector $\boldsymbol{x}$. Then the experts' output vectors, $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_k$, of the $k$-nearest neighbours were calculated and used to compute the experts' local accuracies and to select the most competent expert.

## 4. Illustration

Artificial palm data was used as a testing ground example to illustrate specific characteristics of the fusion rule design methodologies. MLPs with seven hidden units were used as the base classifiers (see Figs 3(a) and 4(b)). 1000 artificially generated vectors from each category were used for training. 1000 vectors from each category composed the test set. Since the training set was large, it was used as a validation set in order to determine the optimal number of iterations in local expert and fusion rule training. In order to have dissimilar experts, the MLPs were trained by standard back propagation with diverse sets selected from regions of the feature space divided by lines $x_1 =$ constant. Training and test errors are presented in Table 1. Generalization error of the best expert is shown in bold.

**Fusion rules compared.** Note that two experts' combinations, composed of either the first three experts or the last four experts,

Table 1
Training and test errors for 2D artificial palm data

| Expert | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Training data error | 0.2180 | 0.2335 | 0.2190 | 0.3265 | 0.3025 | 0.3475 | 0.3145 |
| Test data error | **0.2245** | **0.2245** | 0.2425 | 0.3100 | 0.3175 | 0.3360 | 0.3305 |

Table 2
Performance of different fusion rules

| Aggregates | Oracle | Multinomial | Voting | SLP | Fisher | Quadr. | SLP_Q |
|---|---|---|---|---|---|---|---|
| 1, 2, 3 | 0.001 | 0.1675/**0.1675** | 0.2030 | **0.1675** | 0.2030 | 0.2030 | **0.1675** |
| 1, 2, 3, 4 | 0.001 | 0.0580/**0.0580** | 0.1375 | **0.0580** | 0.0935 | **0.0580** | **0.0580** |
| 1, 2, 3, 4, 5 | 0.000 | 0.0110/**0.0115** | 0.0825 | **0.0115** | 0.0130 | 0.0120 | **0.0115** |
| 1, 2, 3, 4, 5, 6, 7 | 0.000 | 0.0035/**0.0045** | 0.1750 | 0.0055 | 0.0130 | 0.2915 | **0.0045** |
| 4, 5, 6, 7 | 0.002 | 0.0055/**0.0055** | 0.3270 | **0.0055** | 0.0500 | 0.2920 | **0.0055** |

resulted in zero classification error if ideal oracles were used for fusion. The *majority voting rule* was used as the benchmark method. Two other rules included for comparison were the *weighted voting* and the *BKS rule.*

In the experiments, we consider four weighted average fusion rules: (a) single-layer perceptron trained in the linearly transformed (whitened) experts' output space (details in Raudys (2001b, chap. 5)), (b) Fisher linear discriminant function, (c) the standard quadratic discriminant function (Quadr), and (d) the SLP classifier in the polynomial feature space ($o_1$, $o_2$, $o_1^2$, $o_1o_2$, $o_2^2, \ldots$). The latter classifier is named SLP_Q.

**Results.** Table 2 contains estimates of the generalization error, evaluated for five aggregates of the experts. Results for the best fusion rules (for crisp outputs of the experts) are shown in bold. For the multinomial classifiers, we present two generalization error values: left—of the "ideal rule" (estimates of probabilities $P_1^{(s)}$, $P_2^{(s)}, \ldots, P$, $(s = 1, 2)$ were obtained from the *test* set) and right—error rate of the *training* set-based rule. The difference between these two classification error estimates serves as *an indicator of the training set size sufficiency*. Table 2 shows that, when the number of experts is small (3–4), the re-substitution and the test error rate estimates (left and right values in the column for the multinomial classifier) coincide. This means that we do not have small sample effects. For a larger number of experts ($L \geq 5$), we become aware of small sample effects: classification errors of ideal and learning set-based multinomial rules differ.

Table 2 shows that the SLP-based linear and quadratic fusion rules are close in accuracy to the non-parametric local fusion rule—the multinomial classifier. In the large learning set size case, majority voting, SLP and standard Fischer classifiers were notably worse in comparison to the multinomial rule and to the SLP.

For a *small number of experts* the multinomial rule is easily outperformed by the ideal fusion rule, the oracle. Earlier, we already explained the reason: vectors that fall into one single cell (101) belong to opposite categories and are distant in the original feature space (Fig. 3(a)). For that reason, for the aggregate composed of the first three experts, we performed additional experiment. We designed and compared linear SLP and non-linear KDA classifiers *in the continuous space* of the local experts' outputs. The best linear discrimination performed by means of the SLP resulted in 0.1675 test error. It is actually the same performance as the SLP-based fusion rule's in the experts' crisp output space. Using the non-linear fusion rule, the KDA classifier, helped to reduce the generalization error up to 0.012! The experiments of Section 3.1also demonstrated that, in the three first experts' space, the Rastrigin and Erenstein oracle performed very well too. This example makes evident the non-optimality of the classical version of the BKS fusion rule when using a small number of complex, non-linear experts. For a large number of experts, however, employing crisp outputs was advantageous: the generalization errors of both the BKS- and SLP-based fusion rules approach zero. In these experiments, the majority voting rule was always the worst.

The results obtained with artificial data do not generalize. They illustrate the fusion algorithms and explain reasons why a certain fusion rule does not work agreeably, even in large sample size situations. Experiments with real-world data sets (Fumera & Roli, 2005; Raudys, 2001a; Roli, Raudys, & Marcialis, 2002) supported the above conclusions: (a) if the number of experts is low and they are based on different subsets of data, BKS- and SLP-based fusion rules are good in the large sample case; (b) if the number of experts is high and they do not differ in accuracy, the simple average rule or majority voting are preferable in the small sample case.

## 5. Discussion

Multiple classifier systems provide a way to introduce informal user's information into the classifier's design. In this approach, the original pattern recognition task (the data, features, etc.) is divided into several parts. Then, a cooperative decision is made. If the fusion rule is fixed a priori, the designer assumes implicitly that both the separation of the pattern recognition problem into parts and/or splitting the design data set have been performed correctly. Use of trainable fusion rules in an implicit way illustrates the designer's disbelief that all information contained in the base classifiers design was already extracted from design data.

The first conclusion following from our analysis concerns the choice between crisp and continuous outputs of expert classifiers: if one succeeds in having a small number of expert classifiers working perfectly in different parts of the input feature space, then crisp outputs may be preferable over continuous outputs. Exceptions arise if the collective of experts form the cells with distant training vectors of diverse categories inside particular cells. For that reason, new modifications of the BKS method should be developed.

If the experts are incompetent in any part of the input feature space and the numbers of categories and experts are small, then using the continuous experts' outputs could be more useful. In such a case, the difficulties of the pattern recognition task could be transmitted onto the boss's shoulders: to obtain reliable MCS, the boss needs to design a complex non-linear fusion rule. In general, in order to utilize a multiple classifier system effectively, a trainable fusion rule should be designed in low-dimensional feature space. To fulfill this requirement, the number of experts and the number of pattern classes must not to be large.

However, the theoretical considerations of Section 3 indicate that, in optimal weighted average fusion, we obtain *negative weights*. This suggests new research directions for seeking methods to improve the coverage of the expert classifiers in MCS design. A successful attempt to do this for a simple average rule was made in Islam, Yao, and Murase (2003) recently.

A variety of linear and non-linear classification algorithms developed in statistical pattern recognition can serve as trainable fusion rules. The relative value of different fusion rules depends on the associations between the experts, the fusion rule's complexity and the sample size. Sample size issues and possible ways to resolve the problems will be discussed in Part II of this paper (Raudys, 2006).

## Acknowledgments

The author thanks Professors Ray Somorjai, Fabio Roli and Aini Hussain for useful remarks, challenging discussions and aid in preparing the final version of the paper. The author appreciates the notable contribution of anonymous reviewers.

## References

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. New York: Chapman & Hall.

Clemen, R. T. (1989). Combining forecasts: a review and annotating bibliography. *International Journal of Forecasting*, *5*, 559–583.

Cochran, W. D., & Hopkins, C. (1961). Some classification problems with multivariate qualitative data. *Biometrics*, *17*, 11–31.

Diettrich, T. G. (2000). Ensemble methods in machine learning. In *Lecture notes in computer science*: *Vol. 1857* (pp. 1–15).

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification and scene analysis* (2nd ed.). New York: John Wiley.

Friedman, J. M. (1989). Regularized discriminant analysis. *Journal of American Statistical Association*, *84*, 165–175.

Fukunaga, K. (1990). *Introduction to statistical pattern recognition* (2nd ed.). New York: Academic Press.

Fumera, F., & Roli, F. (2005). A theoretical and experimental analysis of linear combiners for multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 942–956.

Giacinto, G., Roli, F., & Fumera, G. (2000). Selection of classifiers based on multiple classifier behaviour. In *Lecture notes in computer science*: *Vol. 1876* (pp. 87–93).

Giacinto, G., & Roli, F. (2001). Dynamic classifier selection based on multiple classifier behaviour. *Pattern Recognition*, *34*(9), 179–181.

Gosh, J. (2002). Multi-classifier systems: back to the future. In *Lecture notes in computer science*: *Vol. 2364* (pp. 1–15).

Güler, C., Sankur, B., Kahya, Y., Skurichina, M., & Raudys, S. (1996). Classification of respiratory sound patterns by means of cooperative neural networks. In G. Ramponi, G. L. Sicuranza, S. Carrato, S. Marsi (Eds.), *Proceedings of 8th European signal processing conference* (pp. 859–862).

Hashem, S. (1997). Optimal linear combination of neural networks. *Neural Networks*, *19*, 599–614.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Ho, T. K. (2001). Data complexity analysis for classifier combination. In *Lecture notes in computer science*: *Vol. 2096* (pp. 53–67).

Huang, Y. S., & Suen, C. Y. (1995). A method of combining multiple experts for recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*, 90–94.

Islam, M. M., Yao, X., & Murase, K. (2003). A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, *14*, 820–834.

Janeliunas, A., & Raudys, S. (2002). Reduction of boasting bias' of linear expert. In *Lecture notes in computer science*: *Vol. 2364* (pp. 242–251).

Kang, H. J. (2003). Combining multiple classifiers based on third-order dependency for handwritten numeral recognition. *Pattern Recognition Letters*, *24*, 3027–3036.

Kittler, J. (1998). Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, *1*, 18–27.

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

Kittler, J., & Roli, F. (Eds.) (2000). Multiple classifier systems. *Lecture notes in computer science*: *Vol. 1857*.

Kittler, J., & Roli, F. (Eds.) (2001). Multiple classifier systems. *Lecture notes in computer science*: *Vol. 2096*.

Kittler, J., & Roli, F. (Eds.) (2002). Multiple classifier systems. *Lecture notes in computer science*: *Vol. 2364*.

Kittler, J., & Roli, F. (Eds.) (2003). Multiple classifier systems. *Lecture notes in computer science*: *Vol. 2709*.

Kittler, J., & Roli, F. (Eds.) (2004). Multiple classifier systems. *Lecture notes in computer science*: *Vol. 3077*.

Kuncheva, L. I., Bezdek, J. C., & Duin, R. P. W. (2001). Decision templates for multiple classifier fusion: And experimental comparison. *Pattern Recognition*, *34*, 299–314.

Kuncheva, L. I. (2004). *Combining pattern classifiers: Methods and Algorithms*. Hoboken, NJ: Wiley.

Lachenbruch, P. A., & Goldstein, M. (1979). Discriminant analysis. *Biometrics*, *5*(3), 9–85.

Lbov, G. S. (1981). *Methods of processing experimental data with mixed variables*. Novosibirsk: Nauka (in Russian).

Linhart, G. (1959). Techniques of discriminant analysis with discrete variables. *Metrika*, *2*(116), 138–140.

Nilsson, N. (1965). *Learning machines*. New York: McGraw-Hill Book Company.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Rahman, A. F. R., & Fairhurst, M. C. (2003). Multiple classifier decision combination strategies for character recognition: A review. *International Journal of Document Analysis and Recognition*, *5*(4), 166–194.

Ramachandran, R. P., Farrell, K. R., & Mammone, R. J. (2002). Speaker recognition—general classifier approaches and data fusion methods. *Pattern Recognition*, *35*, 2801–2821.

Rastrigin, L. A., & Erenstein, R. Ch. (1973). On decision making in a collective of decision rules. *Priborostroenie*, *16*(11), 31–35 (in Russian).

Rastrigin, L. A., & Erenstein, R. Ch. (1981). *Method of collective recognition*. Moscow: Energoizdat (in Russian).

Raudys, S. (1991). On the effectiveness of Parzen window classifier. *Informatica*, *2*, 434–454.

Raudys, S. (1998). Evolution and generalization of a single neurone. I. SLP as seven statistical classifiers. *Neural Networks*, *11*, 283–296.

Raudys, S. (2001a). Combining the expert networks: a review. In R. Sadykhov (Ed.), *Proc. of internat. conf. on neural networks and artificial intelligence* (pp. 81–91). Minsk.

Raudys, S. (2001b). *Statistical and neural classifiers: An integrated approach to design*. London: Springer-Verlag.

Raudys, S. (2002). Multiple classification systems in the context of feature extraction and selection. In *Lecture notes in computer science*: *Vol. 2364* (pp. 27–41).

Raudys, S. (2003). Experts' boasting in trainable fusion rules. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1178–1182.

Raudys, S. (2006). Trainable fusion rules. II. Small sample-size effects. *Neural Networks*. doi:10.1016/j.neunet.2006.01.019.

Raudys, S., & Jain, A. K. (1991). Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*, 252–264.

Raudys, S., & Roli, F. (2003). The behavior knowledge space fusion method: analysis of generalization error and strategies for performance improvement. In *Lecture notes in computer science*: *Vol. 2709* (pp. 55–64).

Roli, F., Raudys, S., & Marcialis, G. L. (2002). An experimental comparison of fixed and trained rules for crisp classifiers outputs. In *Lecture notes in computer science*: *Vol. 2364* (pp. 232–241).

Sabourin, M., Mitiche, A., Thomas, D., & Nagy, G. (1993). Classifier combination for handprinted digit recognition. In *Proc. second int. conf. document analysis and recognition* (pp. 163–166).

Telksnys, L. (1968). Two-stage optimal recognition systems. *Cybernetics*, *4*(2), 89–92. Kiev.

Ueda, N. L. (2000). Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, *22*, 207–215.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Berlin: Springer-Verlag.

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

Xu, L., Krzyzak, A., & Suen, C. Y. (1992). Methods of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions Systems Man and Cybernetics*, *22*, 418–435.