Pattern Analysis & Applications (2002)5:121–135

Bagging, Boosting and the Random Subspace Method for Linear Classifiers

Marina Skurichina and Robert P. W. Duin

Pattern Recognition Group, Department of Applied Physics, Faculty of Applied Sciences, Delft University of Technology, Delft, The Netherlands

Abstract: Recently bagging, boosting and the random subspace method have become popular combining techniques for improving weak classifiers. These techniques are designed for, and usually applied to, decision trees. In this paper, in contrast to a common opinion, we demonstrate that they may also be useful in linear discriminant analysis. Simulation studies, carried out for several artificial and real data sets, show that the performance of the combining techniques is strongly affected by the small sample size properties of the base classifier: boosting is useful for large training sample sizes, while bagging and the random subspace method are useful for critical training sample sizes. Finally, a table describing the possible usefulness of the combining techniques for linear classifiers is presented.

Keywords: Bagging; Boosting; Combining classifiers; Linear classifiers; Random subspaces; Training sample size

1. INTRODUCTION

When data are highly dimensional and the training sample size is small compared to the data dimensionality, it may be difficult to construct a good single classification rule. Usually, a classifier constructed on small training sets is biased and has a large variance as the classifier parameters (coefficients) are poorly estimated. Consequently, such a classifier may be weak, having a poor performance [1]. Moreover, often it will be unstable: small changes in the training set cause large changes in the classifier. In general, bad performance of a classifier can be caused by different factors: incorrect assumptions about the model when constructing a classifier; too low a complexity of the classification rule to solve the problem; incorrect settings for classifier parameters; instability of the classifier, etc. Consequently, in the literature the term 'weak classifier' can refer to different things: badly performing classifiers, unstable classifiers, classifiers of a low complexity, or classifiers depending upon certain assumed models that are not always true. However, in all cases when intending to improve a 'weak classifier', one actually aims to improve its performance. Therefore, describing a 'weak classifier' as one that has a poor performance seems to be the most general definition.

To improve a weak classifier (a classifier that has a poor performance), one may use different approaches. One way is to stabilise the decision of a weak classifier (as weak classifiers are often unstable) by regularisation [2] or noise injection [3]. Another approach is to construct many weak classifiers instead of a single one, and to combine them into a powerful decision rule. Recently, a number of combining techniques has been developed. The most popular are bagging [4], boosting [5] and the Random Subspace Method (RSM) [6]. In bagging, one samples the training set, generating random independent bootstrap replicates [7], constructs the classifier on each of these, and aggregates them by a simple majority vote in the final decision rule. In boosting, classifiers are constructed on weighted versions of the training set, which are dependent on previous classification results. Initially, all objects have equal weights, and the first classifier is constructed on this data set. Then, weights are changed according to the performance of the classifier. Erroneously classified objects get larger weights, and the next classifier is boosted on the reweighted training set. In this way, a sequence of training sets and classifiers is obtained, which is then combined by simple majority voting or by weighted majority voting in the final decision. In the random subspace method, classifiers are constructed in random subspaces of the data feature space. These classifiers are usually

Received: 3 November 2000
Received in revised form: 2 November 2001
Accepted: 13 December 2001

combined by simple majority voting in the final decision rule.

Bagging, boosting and the RSM are designed for, and usually applied to, Decision Trees (DT) [6,8–11], where they often produce an ensemble of classifiers, which is superior to a single classification rule. However, these techniques may also perform well for classification rules other than DTs. For instance, it was shown that bagging and boosting may be useful for perceptrons [12,13]. Bagging and the RSM may also be advantageous for k nearest neighbours classification rules [14]. Previously, we have demonstrated that bagging may be beneficial in Linear Discriminant Analysis (LDA) for critical training sample sizes (when the number of training objects is comparable with data dimensionality) [15]. Our earlier studies [16–19] have shown that boosting and the RSM may also be advantageous for linear classifiers.

In this paper, we study the usefulness of bagging, boosting and the RSM in LDA for two-class problems. In particular, we want to investigate their relations with the training sample size and the small sample size properties of the base classifier. All combining techniques discussed are designed for weak classifiers. The most popular linear classifiers are the Nearest Mean Classifier (NMC) [20] and the Fisher Linear Discriminant function (FLD) [20]. However, when the number of training objects is smaller than the data dimensionality, the sample estimate of the covariance matrix is singular. In these circumstances, the FLD cannot be constructed, as it requires the inverse of the covariance matrix. To avoid the direct inverse of an ill-conditioned covariance matrix, one may perform the Moore–Penrose Pseudo inverse, which is used in the Pseudo Fisher Linear Discriminant function (PFLD) [20]. Both the NMC and the PFLD are weak classifiers: the NMC is usually weak when data classes have another distribution than the Gaussian distribution with equal variances; the PFLD as such is weak for critical training sample sizes, and its use as a single classifier is not recommended. Combining techniques, however, may be useful for these classifiers. Therefore, we have chosen the NMC and the PFLD for our comparative simulation study. We discuss the performance and instability of these linear classifiers in Section 2.

Several artificial and real data sets representing two-class problems are used in our simulation study. They are described in Section 4, but first a description of bagging, boosting and the RSM, followed by a comparative discussion, is given in Section 3. Simulation results on the performance of bagging, boosting and the RSM in LDA are discussed in Section 5. Conclusions are summarised in Section 6.

2. PERFORMANCE AND THE INSTABILITY OF LINEAR CLASSIFIERS

To study the usefulness of combining techniques for linear classifiers in relation to their instability and training sample size, let us consider a few linear classifiers. The most popular and commonly used linear classifiers are the *Fisher Linear Discriminant* (FLD) [20] and the *Nearest Mean Classifier* (NMC), also called the *Euclidean Distance Classifier* [20]. The standard FLD is defined as

$$g_F(\boldsymbol{x}) = \boldsymbol{x}'\hat{\boldsymbol{w}}^F + \hat{w}_0^F = \left[\boldsymbol{x} - \frac{1}{2}(\bar{\boldsymbol{X}}^{(1)} + \bar{\boldsymbol{X}}^{(2)})\right]' \boldsymbol{S}^{-1}(\bar{\boldsymbol{X}}^{(1)} - \bar{\boldsymbol{X}}^{(2)}) \quad (1)$$

where $(\hat{\boldsymbol{w}}^F, \hat{w}_0^F)$ are the coefficients of the linear discriminant function $g_F(\boldsymbol{x})$, $\boldsymbol{S}$ is the standard maximum likelihood estimation of the $p \times p$ covariance matrix Σ, $\boldsymbol{x}$ is a p-variate vector to be classified, and $\bar{\boldsymbol{X}}^{(i)}$ is the sample mean vector of the class π_i, $i = 1, 2$.

Notice that Eq. (1) is the mean squared error solution for the linear coefficients $(\boldsymbol{w}, w_0)$ in

$$g_F(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + w_0 = L \quad (2)$$

with $\boldsymbol{x} \in \boldsymbol{X}$, and with L being the corresponding desired outcomes, 1 for class π_1 and -1 for class π_2. When the number of features p exceeds the number of training vectors n, the sample estimate $\boldsymbol{S}$ of the covariance matrix will be a singular matrix that cannot be inverted. For feature sizes p increasing to n, the expected probability of misclassification rises dramatically [1] (see Fig. 1(a)).

The Nearest Mean Classifier can be written as

$$g_{NMC}(\boldsymbol{x}) = \left[\boldsymbol{x} - \frac{1}{2}(\bar{\boldsymbol{X}}^{(1)} + \bar{\boldsymbol{X}}^{(2)})\right]' (\bar{\boldsymbol{X}}^{(1)} - \bar{\boldsymbol{X}}^{(2)}) \quad (3)$$

It minimises the distance between the vector $\boldsymbol{x}$ and the class mean $\bar{\boldsymbol{X}}^{(i)}$, $i = 1, 2$.

The NMC generates the perpendicular bisector of the class means, and thereby yields the optimal linear classifier for classes having the spherical Gaussian distribution of features with the same variance. The advantage of this classifier is that it is relatively insensitive to the number of training examples [21]. The NMC, however, does not take into account differences in the variances and covariances.

The modification of the FLD, which allows us to avoid the inverse of an ill-conditioned covariance matrix, is the so-called *Pseudo Fisher Linear Discriminant* (PFLD) [20]. In the PFLD, a direct solution of Eq. (2) is obtained by (using augmented vectors)

$$g_{PFLD}(\boldsymbol{x}) = (\boldsymbol{w}, w_0) \cdot (\boldsymbol{x}, 1) = (\boldsymbol{x}, 1)\,(\boldsymbol{X}, \boldsymbol{I})^{-1} L \quad (4)$$

where $(\boldsymbol{x}, 1)$ is the augmented vector to be classified and $(\boldsymbol{X}, \boldsymbol{I})$ is the augmented training set. The inverse $(\boldsymbol{X}, \boldsymbol{I})^{-1}$ is the Moore–Penrose Pseudo Inverse, which gives the minimum norm solution. Before the inversion the data are shifted such that they have zero mean. This method is closely related to singular value decomposition.

The behaviour of the PFLD as a function of the sample size is studied elsewhere [15,22]. For one sample per class this method is equivalent to the Nearest Mean and to the Nearest Neighbour methods (see Fig. 1(a)). For values $n > p$ the PFLD, maximising the total distance to all given samples, is equivalent to the FLD (1). For values $n \leq p$, however, the Pseudo Fisher rule finds a linear subspace, which covers all the data samples. The PFLD builds a linear discriminant perpendicular to this subspace in all other directions for which no samples are given. In between, the

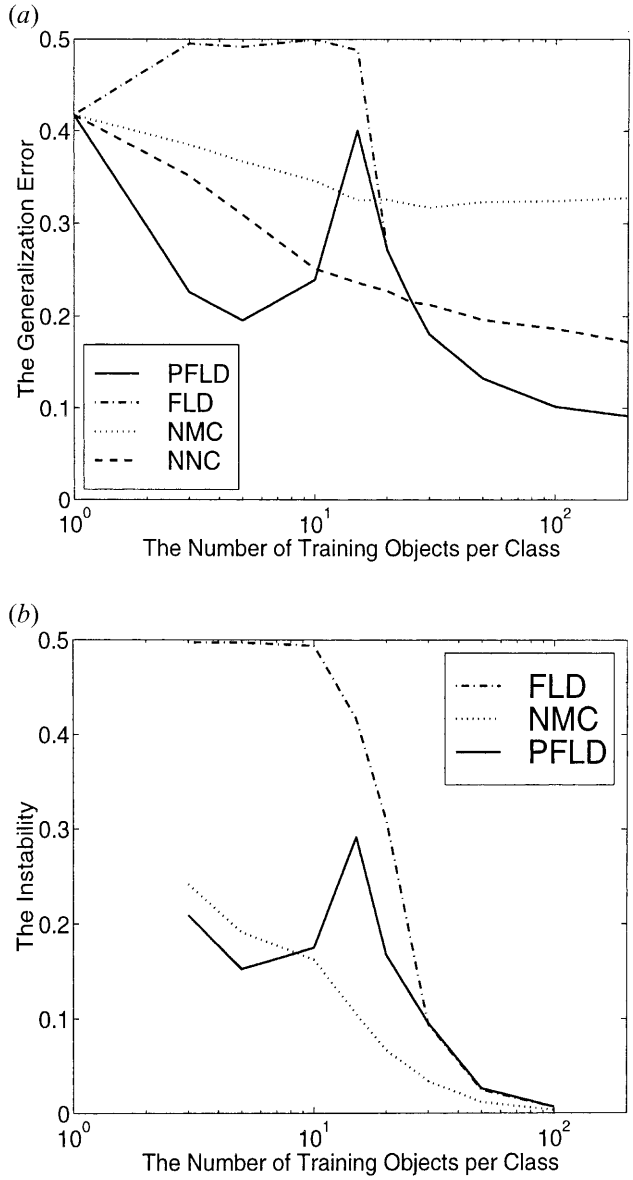

Fig. 1. (a) Generalisation error of the PFLD, FLD, NMC and Nearest Neighbour Classifier (NNC); (b) instability of the linear classifiers versus the training sample size for the 30-dimensional Gaussian correlated data.

generalisation error of the PFLD shows a minimum, and then a maximum, at the point $n = p$. This can be understood from the observation that the PFLD succeeds in finding hyperplanes with equal distances to all training samples until $n = p$. One obtains the exact solution, but all noise presented in the data is covered. In Raudys and Duin [23], an asymptotic expression for the generalisation error of the PFLD is derived, which theoretically explains the behaviour of the PFLD. The advantage of the PFLD is that it is a straightforward, simple classification rule, which takes into account differences in variance. However, it is a weak and a very noise sensitive classifier when it is constructed on critical training sample sizes. In such a situation, it is not recommended to use it as a single classifier. Nevertheless, due to the peaking phenomenon shown by the generalisation error of the PFLD, this classifier is very suitable for studying dimensionality (or training sample size) effects. Here it will be used as a base classifier for the combining techniques.

To understand better when combining techniques can be beneficial, it is useful to consider the instability of a base classifier [15]. The classifier instability is measured by calculating the changes in classification of a test set caused by the bootstrap replicate of the original training data set. Repeating this procedure several times on the training set (we did it 25 times) and averaging the results, one obtains an estimate of the classifier instability on one training set. The mean instability of linear classifiers (on 50 independent training sets) defined in this way is presented in Fig. 1(b). One can see that the instability of the classifier decreases when the training sample size increases. The instability and performance of a classifier are correlated: more stable classifiers perform better than less stable ones. In this example, however, the performance of the NMC does not depend upon the training sample size. In contrast to other classifiers, it remains a poor performing classifier for large training sample sizes, while its stability increases.

3. BAGGING, BOOSTING AND THE RANDOM SUBSPACE METHOD

To improve the performance of weak regression and classification rules, a number of combining techniques can be used. During the last few years, the most popular methods have become bagging, boosting and the random subspace method. They all modify the training data set, build classifiers on these modified training sets, and then combine them into a final decision rule by simple or weighted majority voting. However, they perform in a different way.

3.1. Bagging

Bagging was proposed by Breiman [4], and is based on bootstrapping [7] and aggregating concepts, so it incorporates the benefits of both approaches. Bootstrapping is based on random sampling with replacement. Therefore, taking a bootstrap replicate $\boldsymbol{X}^b = (\boldsymbol{X}_1^b, \boldsymbol{X}_2^b, \ldots, \boldsymbol{X}_n^b)$ (random selection with replacement) of the training set $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)$, one can sometimes avoid or get less misleading training objects in the bootstrap training set. Consequently, a classifier constructed on such a training set may have a better performance. Aggregating actually means combining classifiers. Often a combined classifier gives better results than individual classifiers, because of combining the advantages of the individual classifiers in the final solution. Therefore, bagging might be helpful to build a better classifier on training sample sets with misleaders. In bagging, bootstrapping and aggregating techniques are implemented in the following way:

1. Repeat for $b = 1, 2, \ldots, B$:
   (a) Take a bootstrap replicate $\boldsymbol{X}^b$ of the training data set $\boldsymbol{X}$.

(b) Construct a classifier $C^b(\boldsymbol{x})$ (with a decision boundary $C^b(\boldsymbol{x}) = 0$) on $\boldsymbol{X}^b$.

2. Combine classifiers $C^b(\boldsymbol{x})$, $b = 1, 2, \ldots, B$, by simple majority voting (the most often predicted label) to a final decision rule

$$\beta(\boldsymbol{x}) = \underset{y \in \{-1, 1\}}{argmax} \sum_b \delta_{\mathrm{sgn}(C^b(\boldsymbol{x})),y}$$

where

$$\delta_{i,j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

is the Kronecker symbol, $y \in \{-1, 1\}$ is a decision (class label) of the classifier.

Let us note that linear classifiers may be also aggregated by other combining rules: the mean (when the decision is made according to the mean of *a posteriori* probabilities given by the base classifiers), the product (when the decision is made by the product of *a posteriori* probabilities presented by the base classifiers) or the average combing rule (when the final classifier is obtained by averaging the coefficients of the base classifiers $C^B(\boldsymbol{x}) = \sum_{b=1}^{B} C^b(\boldsymbol{x})$). When using the average combining rule to aggregate linear classifiers, the final classifier is also a linear classifier with the same complexity as the base classifiers.

As mentioned before, bagging is based on bootstrapping the training set and aggregating (combining) the bootstrap versions of the original classifier. On average, when taking a bootstrap sample of the training set, approximately $\frac{1}{e} \approx 37\%$ of the objects are not presented in the bootstrap sample, meaning that possible 'outliers' in the training set sometimes do not show up in the bootstrap sample. Thus, better classifiers (with a smaller apparent error – classification error on the training data set) may be obtained by the bootstrap sample than by the original training set. These classifiers will be presented 'sharper' in the apparent error than those obtained on the training sets with outliers. Therefore, they will be more decisive than other bootstrap versions in the final judgement. Thus, aggregating classifiers in bagging can sometimes give a better performance than individual classifiers. To illustrate this, let us consider the FLD applied to an imaginary two-class problem of one-dimensional data with one outlier. In Fig. 2, one can see that the FLD constructed on the complete data set is not able to separate the data classes without error. Constructing classifiers on bootstrap samples of the training sets sometimes gives a better classifier, sometimes a worse one. The better classifier is sharper in the domain of the *a posteriori* probabilities, so it dominates in the final decision. Therefore, aggregating (combining) bootstrap versions of the original classifier allows us to get a better classifier than the original one.

Another example, presented in Fig. 3, shows the case when, by aggregating bootstrap versions of the classifier, one obtains a solution (a discriminant function) that is impossible to reach by separate bootstrap versions of the classifier. In this example, two-dimensional Gaussian correlated data are used (see Fig. 7). The NMC is bagged on 100 bootstrap replicates of a training data set consisting of 10 objects, and the average combining rule is used to aggregate base classifiers. A scatter plot of the NMC, its bootstrap versions and its bagged version in the space of normalised coefficients w_1/w_0 and w_2/w_0 are shown in Fig. 3. One can see that the bagged classifier gives a solution inside an empty space between different bootstrap versions of the classifier. Separate bootstrap versions of the classifier cannot reach the solution obtained by aggregating bootstrap versions of the classifier.

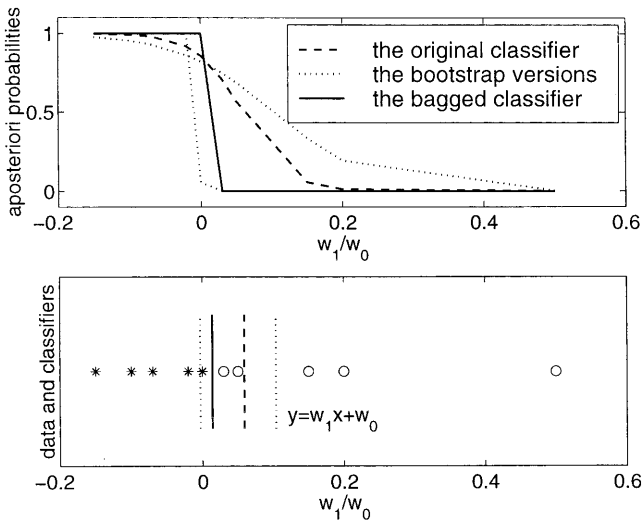

Fig. 2. Scatter plot of the one-dimensional data set with one outlier, the discriminant functions and the *a posteriori* probabilities of the original training set, obtained for the FLD built on the original training set, for two bootstrap versions of the FLD and for the bagged FLD (with the average combining rule) constructed from these two bootstrap versions.

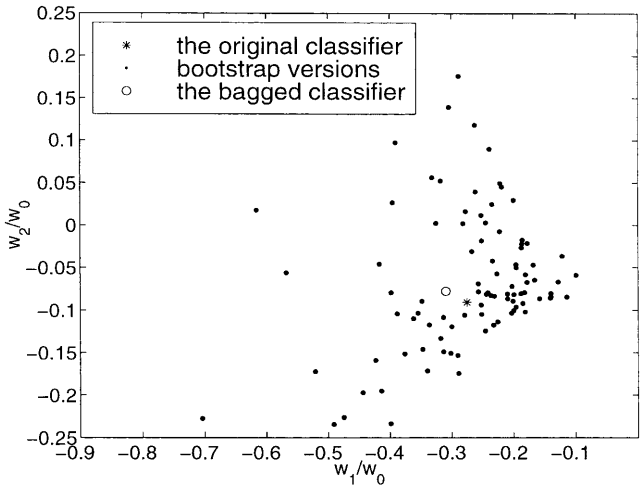

Fig. 3. Scatter plot of the two-dimensional projection of the NMC in the space of its normalised coefficients w_1/w_0 and w_2/w_0 for the two-dimensional Gaussian correlated data set.

3.2. Boosting

Boosting, proposed by Freund and Schapire [5], is another technique to combine weak classifiers having a poor performance in order to get a classification rule with a better performance. In boosting [5] (not in arcing [9]), classifiers and training sets are obtained in a strictly *deterministic* way. Both training data sets and classifiers are obtained *sequentially* in the algorithm, in contrast to bagging, where training sets and classifiers are obtained *randomly* and *independently* (parallelly) from the previous step of the algorithm. At each step of boosting, training data are reweighed in such a way that incorrectly classified objects get larger weights in a new, modified training set. Thus, one actually maximises the margins between training objects. It suggests the connection between boosting and Vapnik's Support Vector Classifier (SVC) [8,24], as the objects obtaining large weights may be the same as the support objects. Although boosting and the SVC have the same aim, computationally they perform in a different way. In the SVC one performs global optimisation in order to maximise the minimal margin, while in boosting one maximises the margin locally for each training object. To illustrate the similarities and differences between boosting and the SVC, let us consider the example of the 30-dimensional Gaussian correlated data set described in Section 4. In Fig. 4(a) one can see that, on average, support vectors found by the SVC get larger weights in boosting than non-support vectors. However, objects getting large weights (larger than 1) in boosting are not always identical to the support vectors (see Fig. 4(b). When the training sample size is small, almost all training objects are found to be the support vectors, while only part of the training objects get large weights in boosting. When the training sample size increases, the set of objects having large weights in boosting becomes larger as well. It also contains more of the support vectors (but not all of them) found by the SVC. On the other hand, the number of support vectors found is smaller than the total number of training objects obtaining large weights in boosting. This means that objects with large weights in boosting are not identical to the support vectors found by the SVC.

Boosting is organised by us in the following way. It is based on the 'arc-fs' algorithm described by Breiman [9], where we reweight the training set instead of resample it. The 'arc-fs' algorithm is the improved version of the standard AdaBoost algorithm [5]. Additionally, we set initial weight values w_i^1, i = 1, …, n to 1 instead of $1/n$, in order to be independent on data normalisation. Therefore, boosting is implemented by us as follows:

1. Repeat for $b = 1, 2, \ldots, B$:
   (a) Construct the classifier $C^b(\boldsymbol{x})$ on the weighted version $\boldsymbol{X}^\star = (w_1^b\boldsymbol{X}_1, w_2^b\boldsymbol{X}_2, \ldots, w_n^b\boldsymbol{X}_n)$ of training data set $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)$, using weights w_i^b, $i = 1, \ldots, n$ (all $w_i^b = 1$ for $b = 1$).
   (b) Compute probability estimates of the error

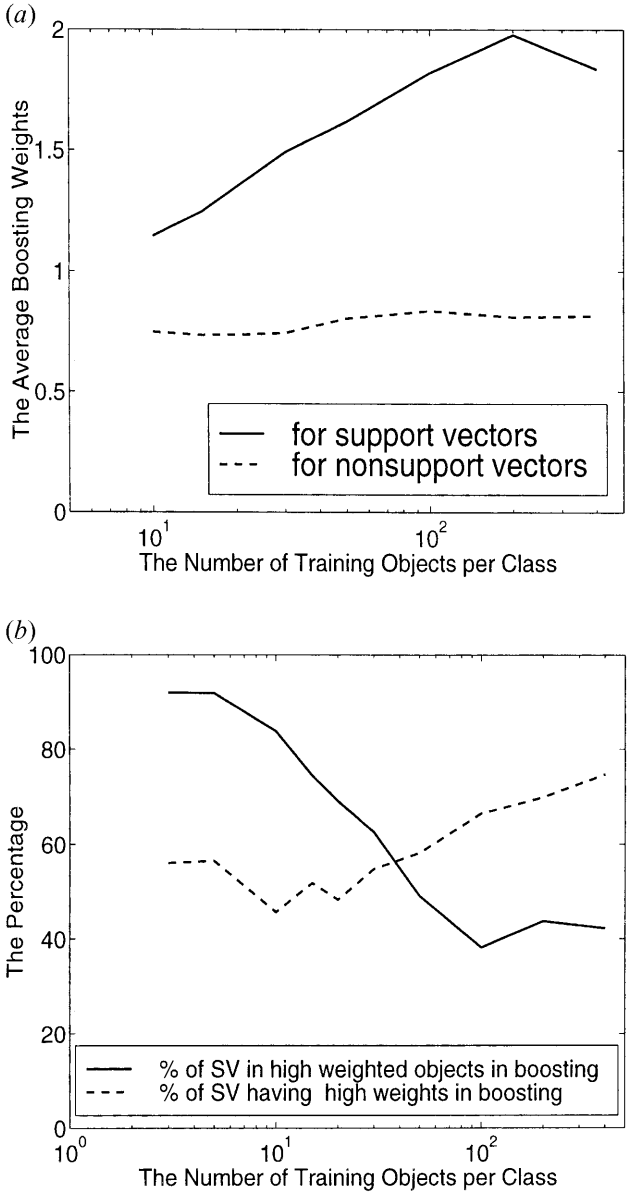

Fig. 4. (a) Average weights obtained in boosting the NMC after 250 steps, for non-support and support vectors, found by the linear SVC, versus the training sample size; (b) percentage of Support Vectors (SV) in training objects that obtain high weights (larger than 1) in boosting the NMC, and the percentage of support vectors having high weights (larger than 1) in boosting versus the training sample size.

$$err_b = \frac{1}{n}\sum_{i=1}^{n} w_i^b \xi_i^b$$

where

$$\xi_i^b = \begin{cases} 0, & \text{if } \boldsymbol{X}_i \text{ is classified correctly} \\ 1, & \text{otherwise} \end{cases}$$

and combining weights

$$c_b = \frac{1}{2}\log\left(\frac{1-err_b}{err_b}\right)$$

(c) If $0 < err_b < 0.5$, set $w_i^{b+1} = w_i^b \exp(c_b \xi_i^b)$, $i = 1, \ldots, n$, and renormalise so that $\sum_{i=1}^{n} w_i^{b+1} = n$. Otherwise, set all weights $w_i^b = 1$, $i = 1, \ldots, n$ and restart the algorithm.

2. Combine classifiers $C^b(\boldsymbol{x})$, $b = 1, 2, \ldots, B$, by weighted majority voting with weights c_b to a final decision rule

$$\beta(\boldsymbol{x}) = \underset{y \in \{-1, 1\}}{argmax} \sum_b c_b \delta_{\mathrm{sgn}(C^b(\boldsymbol{x})), y}$$

where $\delta_{i,j}$ is the Kronecker symbol, and $y \in (-1, 1\}$ is a decision (class label) of the classifier.

3.3. The Random Subspace Method

The Random Subspace Method (RSM) is the combining technique proposed by Ho [6]. In the RSM, one also modifies the training data. However, this modification is performed in the feature space. Let each training object $\boldsymbol{X}_i$ ($i = 1, \ldots, n$) in the training sample set $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)$ be a p-dimensional vector $\boldsymbol{X}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$, described by p features (components). In the RSM, one randomly selects $r < p$ features from the p-dimensional data set $\boldsymbol{X}$. One thus obtains the r-dimensional random subspace of the original p-dimensional feature space. Therefore, the modified training set $\tilde{\boldsymbol{X}}^b = (\tilde{\boldsymbol{X}}_1^b, \tilde{\boldsymbol{X}}_2^b, \ldots, \boldsymbol{X}_n^b)$ consists of r-dimensional training objects $\tilde{\boldsymbol{X}}_i^b = (x_{i1}^b, x_{i2}^b, \ldots, x_{ir}^b)$ ($i = 1, \ldots, n$), where r components x_{ij}^b ($j = 1, \ldots, r$) are randomly selected from p components x_{ij} ($j = 1, \ldots, p$) of the training vector $\boldsymbol{X}_i$ (the selection is the same for each training vector). Then one constructs classifiers in the random subspaces $\tilde{\boldsymbol{X}}^b$ and combines them by simple majority voting in the final decision rule. So, the RSM is organised in the following way:

1. Repeat for $b = 1, 2, \ldots, B$:
   (a) Select an r-dimensional random subspace $\tilde{\boldsymbol{X}}^b$ from the original p-dimensional feature space $\boldsymbol{X}$.
   (b) Construct a classifier $C^b(\boldsymbol{x})$ (with a decision boundary $C^b(\boldsymbol{x}) = 0$) in $\tilde{\boldsymbol{X}}^b$.
2. Combine classifiers $C^b(\boldsymbol{x})$, $b = 1, 2, \ldots, B$, by simple majority voting to a final decision rule

$$\beta(\boldsymbol{x}) = \underset{y \in \{-1, 1\}}{argmax} \sum_b \delta_{\mathrm{sgn}(C^b(\boldsymbol{x})), y}$$

where $\delta_{i,j}$ is the Kronecker symbol, and $y \in \{-1, 1\}$ is a decision (class label) of the classifier.

The RSM may benefit from using random subspaces for both constructing and aggregating the classifiers. When the number of training objects is relatively small compared with the data dimensionality, by constructing classifiers in random subspaces one may solve the small sample size problem. The subspace dimensionality is smaller than in the original feature space, while the number of training objects remains the same. Therefore, the relative training sample size increases. When data have many redundant features, one may obtain better classifiers in random subspaces than in the original feature space. The combined decision of such classifiers may be superior to a single classifier constructed on the original training set in the complete feature space.

3.4. Discussion

Bagging, boosting and the RSM have been theoretically and experimentally investigated and compared to each other and to other combining techniques by many researchers. It was shown that they are beneficial for regression and classification trees [4–10,11,22]. They may be useful for perceptrons [12,13], and for k nearest neighbours classifiers [14,25]. It was demonstrated that boosting often gives a better performance than bagging, and the RSM may outperform them both. In Breiman [9] it was stated that bagging and boosting are useful for unstable classifiers. It was demonstrated [9,22] on several examples that both bagging and boosting reduce the variance of the classifier. However, it was shown [8] that a large variance of the classifier is not a requirement for boosting to be effective.

The usefulness of bagging and boosting in LDA was also studied by simulation experiments [9]. Just one linear classifier (probably the FLD) was considered, and the relation between the training sample size and the data dimensionality was not taken into account. From this study, the conclusion was made that neither bagging nor boosting are beneficial for linear classifiers, because in general, linear classifiers are stable. Section 2 of this paper contradicts that conclusion, illustrating that the instability of linear classifiers depends upon the training sample size and on their complexity. In Breiman [9] large training sample sizes are used. Indeed, the linear classifiers are stable when trained on large training data sets, and bagging is not beneficial in that case. The failure of boosting was probably caused by the improper choice of the linear classifier. We should mention that, as a rule, no systematic study (for decision trees and other classifiers) was performed on the performance of boosting and bagging and on their comparison with respect to the training sample size. Usually, large training sample sizes were considered (e.g. taking 90% of available objects for training and 10% for testing). In our study, the training sample size plays an important role. In Section 5, we investigate the usefulness of bagging, boosting and the RSM for linear classifiers in relation to the training sample size.

We studied bagging for linear classifiers [15]. Previously, it was noticed [4] that the efficiency of bagging depends upon the stability of regression or classification. We have also established that bagging may be useful for linear classifiers when they are unstable. This happens when the training sample size is critical, that is when the number of training objects is comparable with the data dimensionality (and with the number of parameters in linear classifiers). For very small and also for very large training sample sizes, bagging is usually useless, as bootstrapping such training sets is not effective. Additionally, in bagging the actual training sample size is reduced, because on average only $1-1/e = 63\%$ of the training objects is used in each bootstrap replicate of the training data set. Ignoring the object repetition in a

bootstrap sample, the training set becomes smaller. In this way, the bootstrap sample is comparable with a smaller training set, therefore classifiers should be similar. So the bagged classifier will have similar characteristics as the classifier built on the smaller training set.

This phenomenon is the most evident for the PFLD due to its characteristic maximum of the generalisation error for critical sizes of the training set. Indeed, considering the effect of bagging on the PFLD (see Fig. 5), it can be clearly seen that *the generalisation error of the bagged classifier is shifted* with reference to the generalisation error of the original classifier in the direction of the generalisation error of the classifier built on a smaller training sample set (see Fig. 6). Thus, the usefulness of bagging also depends upon the small sample size properties of the base classifier [15]. This means that, due to the shifting effect on the generalisation error in the direction of the generalisation error obtained on the smaller training sets, bagging will not be useful for classifiers whose generalisation error decreases with an increase in the training sample size (we call such classifiers those with a decreasing learning curve). Therefore, one may expect that bagging may only be useful for two linear classifiers: for the NMC and especially for the PFLD. Usually, the behaviour of the generalisation error of the NMC is not affected by the training sample size. The generalisation error of the PFLD has a peaking behaviour (see Fig. 1(a)). Bagging may thus be beneficial for these two classifiers when they are constructed on critical training sample sizes.

In boosting one is focused on the difficult objects in the training set, which are usually objects on the border between data classes. Therefore, the number of actually used training objects decreases, and the performance of boosting is also affected by the training sample size [17]. It is not advantageous for classifiers having a decreasing learning curve. In addition, it was shown [26] that in boosting, the generalisation error $\hat{P}_N$ of the final rule $\beta(\boldsymbol{x})$ is bounded,

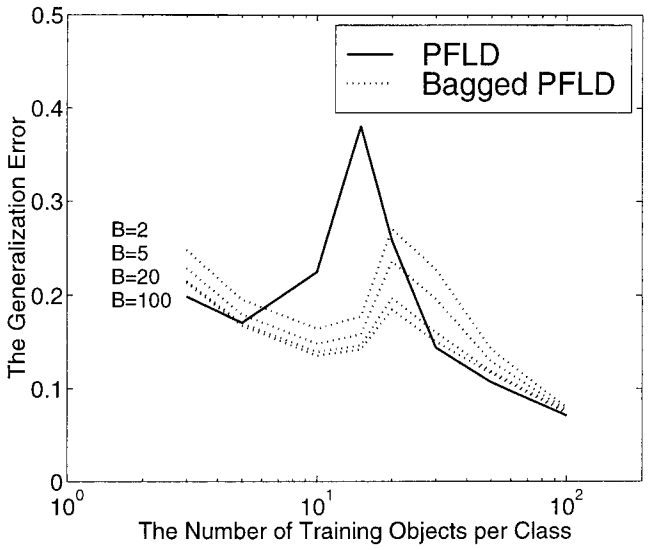

Fig. 5. Generalisation error of the PFLD and the 'bagged' PFLD for different numbers of bootstraps B versus the training sample size for the 30-dimensional Gaussian correlated data.

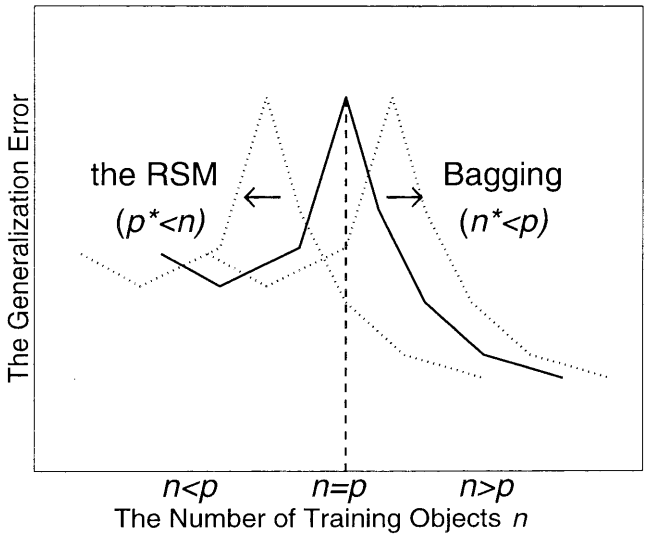

Fig. 6. Shifting effect of the generalisation error of the PFLD for bagging and the RSM. In bagging, the learning curve shifts with respect to the learning curve of the original classifier in the direction of larger training sample sizes (the bagged classifier performs similar to the original classifier constructed on a smaller training set). In the RSM, the learning curve of the final classifier shifts with respect to the learning curve of the original classifier in the direction of smaller training sample sizes (the final classifier obtained in the RSM performs similar to the original classifier constructed on a larger training set). Here, n is the number of training objects and p is the data dimensionality.

$$\hat{P}_N \leq \hat{P}_A + \tilde{O}\left(\sqrt{\frac{Bd}{n}}\right) \tag{5}$$

where $\hat{P}_A$ is the apparent error, n is the training sample size, d is the Vapnik–Chervonenkis (VC) dimensionality [27] of the parameter space of the weak learner, and B is the number of iterations (the number of combined base classifiers) used in boosting. This formula shows that boosting is the most effective for classifiers having a simple complexity (with the low VC-dimensionality d) and for large training sample sizes n. Therefore, in LDA, boosting will only be useful for the NMC constructed on large training sample sizes, as the NMC is the most simple linear classifier and remains weak on large training sample sizes.

In the RSM, by using random subspaces, one actually decreases the data dimensionality. This implies that the performance of the RSM in LDA will be affected by the small sample size properties of the base classifier. We expect that, similar to bagging, in the RSM the final classifier will have a shifting effect of the generalisation error with respect to the generalisation error of the original classifier (see Fig. 6). However, this shift will be in the opposite direction: in the direction of the generalisation error obtained on larger training sample sizes. Thus, the RSM will be useful for classifiers having a decreasing learning curve (e.g. the FLD), and will be useless for the NMC, as its performance does not depend upon the training sample size. We also expect the RSM to be effective for the PFLD constructed on critical training sample sizes.

In relation to the above discussion, one can see that to study the efficiency of bagging, boosting and the RSM for linear classifiers, with respect to the training sample size and the small sample size properties of the base classifier, is of great interest. We perform our study for two linear classifiers: the NMC and the PFLD. This choice is made for two reasons: first, they have quite different small sample size properties that allow us to study the effect of these on the performance of the combining techniques considered; secondly, it seems that only these two classifiers in LDA may benefit from the combining techniques.

4. DATA

To illustrate and compare the performance of combining techniques in different situations, a number of artificial and real data sets are used in our experimental investigations. Mostly, highly dimensional data sets are considered, because we are interested in unstable situations when the data dimensionality is smaller or comparable to the training sample size. In our study, we use artificial data sets which have many redundant features, as some of the combined techniques (e.g. the RSM) are most effective for data with a small intrinsic dimensionality. Real data sets are needed to show that these techniques may be effective when solving real world problems. Some of the real data sets considered are used by other researchers [6,9] in the comparative study of bagging, boosting and the RSM for DTs and in the LDA. Therefore, we decided to involve them in our study too.

Two artificial data sets and five real data sets are used for our experimental study. The first set is a *200-dimensional correlated Gaussian data* set made of two classes with equal covariance matrices. Each class consists of 500 vectors. The mean of the first class is zero for all features. The mean of the second class is equal to 3 for the first two features and 0 for all other features. The common covariance matrix is a diagonal matrix with a variance of 40 for the second feature and a unit variance for all other features. The intrinsic class overlap (Bayes error) is 0.0717. This data set is rotated for the first two features using a rotation matrix $\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$. These features are presented in Fig. 7. The first 30 features of this data set constitute the *30-dimensional correlated Gaussian data set.*

The second artificial data set consists of two *200-dimensional spherical Gaussian data* classes with equal covariance matrices. Each data class contains 500 vectors. The first data class is distributed spherically with a unit covariance matrix and with zero mean. The covariance matrix of the second data set is also unit. The mean of the second class is equal to 0.25 for all features.

The real data sets are taken from the UCI Repository [28]. They are the 34-dimensional *ionosphere* data set, the 8-dimensional *pima-diabetes* data set, the 60-dimensional *sonar* data set, the 30-dimensional *wdbc* data set and the 24-dimensional *german* data set. These data sets have been used [6,9], when studying bagging, boosting and the RSM for decision trees. The diabetes data set was also used when bagging and boosting were applied in LDA [9].

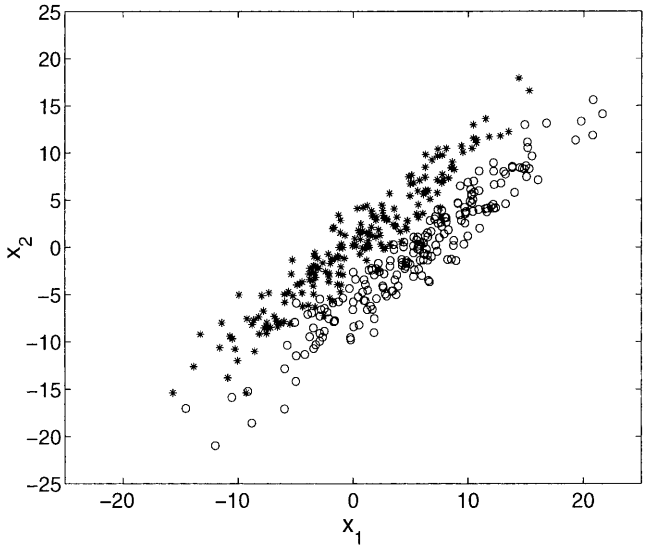

Fig. 7. Scatter plot of a two-dimensional projection of the 200-dimensional Gaussian correlated data.

Training sets are chosen randomly from a total data set. The remaining data are used for testing. All experiments are repeated 50 times on independent training sets. In all figures (besides Figs 2 and 3), the averaged results over 50 repetitions are presented. The standard deviations of the mean generalisation errors for single and combined linear classifiers are of similar orders for each data set. When increasing the training sample size, they are decreasing approximately from 0.018 to 0.005.

Bagging, boosting and the RSM are techniques which combine multiple classifiers. They fundamentally differ from each other in the way in which different versions of the classifier are obtained. By using a different combining rule in each of the combining techniques, we make the difference between them even larger. Our previous study [16,19] has revealed that the choice of combining rule may be very important when combining classifiers in bagging, boosting and the RSM (see Fig. 8). Therefore, to perform a fair comparison of bagging, boosting and the RSM, we decided to use the weighted majority voting combining rule (as it is defined in boosting) in each of them.

5. PERFORMANCE OF BAGGING, BOOSTING AND THE RSM IN LDA

Let us now consider the performance of bagging, boosting and the RSM in LDA on the example of the NMC and the PFLD. These combining techniques perform differently for different base classifiers. In addition, their performance is strongly affected by the training sample size.

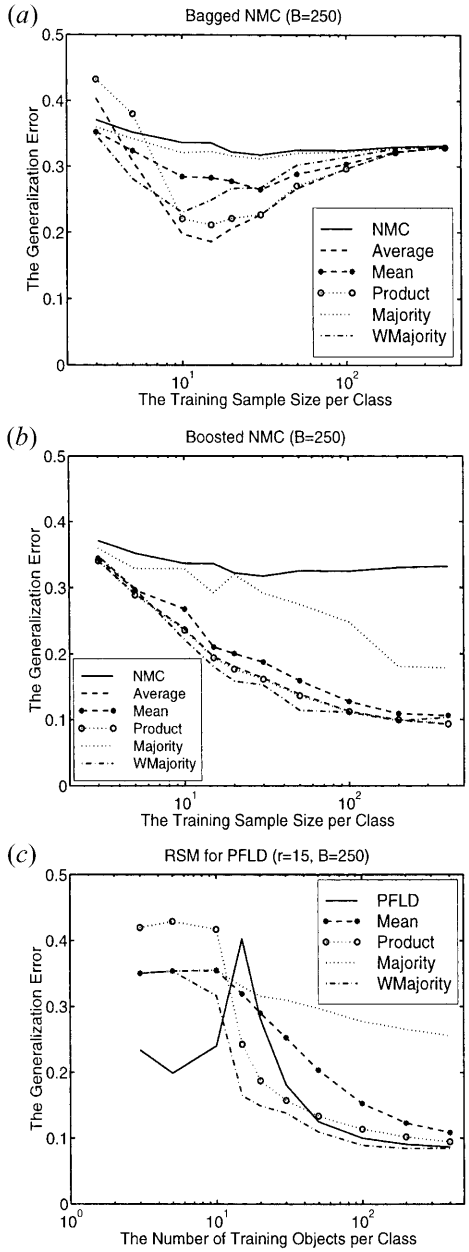

Fig. 8. Generalisation error of (a) the bagged NMC, (b) the boosted NMC and (c) the RSM ($R = 15$) applied to the PFLD using different combining rules (the average, mean, product, simple majority and weighted majority voting) for the 30-dimensional Gaussian correlated data.

5.1. The NMC

Simulation results (see Fig. 9) show that the combining techniques may be useful for the NMC. Bagging improves the generalisation error of the NMC for critical training sample sizes (see Figs 9(a),(c),(d)), when the classifier is unstable and bootstrapping the training data set is the most effective. When training sets are very small, they often represent the distribution of the entire data set incorrectly. By bootstrapping such small training sets, they are expected to become even worse. Classifiers obtained on them may have a poor performance, and combining such classifiers will probably not be beneficial. Therefore, bagging is useless for very small training sample sizes. When the training sample size is large, the classifier is stable. Large training sets represent the real data distribution more accurately, therefore perturbations in the composition of the training set (bootstrapping) do not change the training set very much. Classifiers obtained on bootstrap replicates will also be similar to each other. Combining similar classifiers is not very effective, so bagging is also useless for large training sample sizes.

In boosting, incorrectly classified objects get larger weights. These are mainly objects on the border between data classes which are difficult to classify. Therefore, boosting performs the best for large training sample sizes, when the border between data classes becomes more informative and gives a good representation of the real distribution of the data classes. In this case (for large training sample sizes), boosting the NMC performs similar to the linear SVC (see Figs 9(a),(c),(d),(e),(f)). However, when the training sample size is large, the NMC is stable. This made us think that, in contrast to bagging, the usefulness of boosting does not depend directly upon the stability of the classifier. It depends upon the 'quality' (distribution) of the 'difficult' objects, and on the potential ability of the classifier (its complexity) to distinguish among them correctly.

One may see that for small and critical training sample sizes, the boosted NMC may perform worse or even much worse (having a high peak of the generalisation error) than the original NMC. This is for two reasons: first, the objects on the border between data classes (which are getting larger weights in the boosting algorithm) often have a distribution other than the original training set; secondly, in boosting, only a portion of training objects (on the border) get large weights. These objects are more decisive than other objects when constructing the classifier. Therefore, the number of effectively used training objects in boosting is actually reduced, compared with the original training set. Smaller training sets give more biased sample estimates of class means. Using very small training sets, that consist of objects located on the border between data classes, may be very misleading when constructing the classifier for the entire data set. Therefore, the NMC constructed on such training sets may perform very poorly. A combination of the bad quality classifiers constructed on the smaller training sets may perform worse than the single classifier constructed on the larger training set.

The RSM is usually useless for the NMC (see Figs 9(a),(b),(c),(e),(f)). However, if some data features are redundant (and the discrimination power is evenly spread over the features [6,19]) and the learning curve of the NMC decreases with an increase of the training sample size, sometimes the RSM may improve the performance of the NMC (see Figs 9(d,g)). It happens because the relative number of training objects increases when constructing classifiers in random subspaces. If classifiers obtained in random subspaces are diverse and perform better than the original

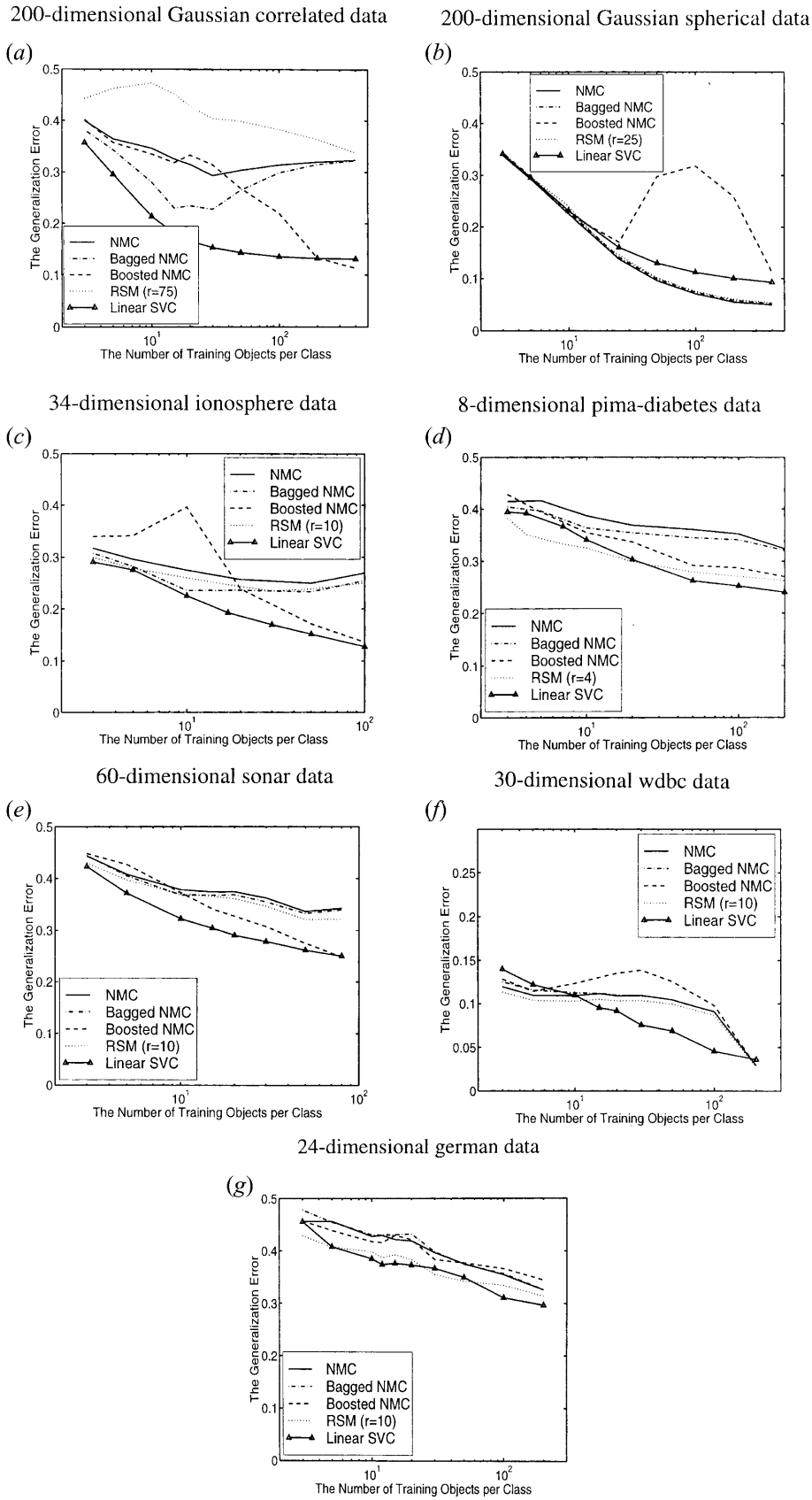

Fig. 9. Performance of bagging, boosting and the RSM (B = 250) for the NMC.

classifier constructed in the complete feature space, combining such classifiers is useful.

Let us note that bagging, boosting and the RSM are useless for the NMC constructed on the Gaussian spherical data set and on the wdbc data set (see Figs 9(b),(f)). The NMC is not a weak classifier for these data sets. Additionally, the NMC is the optimal classifier for the Gaussian spherical data set. By modifying the training set, it is difficult to get better classifiers than the single classifier obtained on the original training set. Combining such classifiers is not beneficial.

5.2. The PFLD

Simulation results (see Fig. 10) show that bagging may be useful for the PFLD constructed on critical training sample sizes. The generalisation error of the PFLD has a peaking behaviour for increasing the training sample size: first, the generalisation error may decrease, then it increases achieving a maximum at the point when the training sample size is equal to the data dimensionality $n = p$; afterwards it again decreases. Bagging the PFLD is useless for very small and very large training sets, as bootstrapping these data sets is not effective. However, due to the shifting effect of bagging on the generalisation error of the classifier in the direction of the generalisation errors obtained on smaller training sample sizes (see Fig. 6), one may significantly improve the performance of the PFLD in the region of critical training sample sizes.

Boosting the PFLD is useless. For training sample sizes larger than the data dimensionality, the PFLD, maximising the distance to all given samples, is equivalent to the FLD. For training sample sizes smaller than the data dimensionality, however, the PFLD finds a linear subspace, which covers all the data samples. On this plane the PFLD estimates the data means and the covariance matrix, and builds a linear discriminant perpendicular to this subspace in all other directions for which no samples are given. Therefore, for these training sample sizes, the apparent error (the classification error on the training set) of the PFLD is always zero. The boosting algorithm does not start as it requires a non-zero apparent error. For training sample sizes larger than the data dimensionality, $n > p$, when the PFLD is equivalent to the FLD, the PFLD is a stable and strong classifier. In this case, boosting may perform similar to a single classifier (if the number of objects on the border is sufficiently large to construct a good FLD), or may worsen the situation (if the number of actually used training objects at each step of boosting is not sufficiently large to define a good FLD). Thus, boosting is useless for the PFLD.

The RSM may be very useful for the PFLD. Due to the shifting effect on the generalisation error of the classifier in the direction of the generalisation error obtained on larger training sets (see Fig. 6), the RSM may significantly improve the performance of the PFLD constructed on critical training sample sizes, even when constructing the classifiers in very small subspaces. Obviously, the efficient dimensionality of random subspaces depends upon the level of redundancy in the data feature space. When the training sample size is large, the PFLD is no longer a weak classifier, and the RSM becomes less useful.

6. CONCLUSIONS

Summarising the simulation results presented in the previous section, we can conclude that bagging, boosting and the random subspace method may be useful in LDA. However, their efficiency is strongly affected by the training sample size, and by the choice of the base classifier (see Fig. 11).

- **Bagging**. Bagging is useful in LDA for weak and unstable classifiers. Usually this happens when classifiers are constructed on critical training sample sizes (when the number of training objects is comparable with the data dimensionality and, therefore, with the number of parameters in a linear classifier).

  Bagging has a shifting effect on the generalisation error of the base classifier in the direction of the generalisation error obtained on smaller training sample sizes (see Fig. 6). Bagging is therefore useless for classifiers having a decreasing learning curve (that is, when the generalisation error of the base classifier decreases with an increase in the training sample size). Therefore, bagging may be beneficial only for the NMC (where performance is not affected much by the training sample size, if data classes have another distribution than the Gaussian with equal variances) and for the PFLD (which has a peaking behaviour of the generalisation error), when they are constructed on critical training sample sizes.
- **Boosting**. Boosting may be useful in LDA only for classifiers that perform poor on large training sample sizes. Such a classifier is the NMC.

  Boosting is useful only for large training sample sizes, if the objects on the border give a better representation of the distribution of the data classes than the original distribution of the data classes and the classifier is potentially able (by its complexity) to distinguish among them well.

  It was shown theoretically and experimentally for DTs [8] that boosting increases the margins of the training objects. Boosting is thus similar to the maximum margin classifiers [24], based on the number of support vectors. In this paper, we have experimentally shown that boosted linear classifiers may achieve the performance of the linear support vector classifier when training sample sizes are large compared with the data dimensionality.

  As boosting is useful only for large training sample sizes, when classifiers are usually stable, the performance of boosting does not depend upon the instability of the classifier.
- **The Random Subspace Method**. The RSM may be useful for weak linear classifiers obtained on small and critical training sample sizes, because when constructing classifiers in random subspaces, one relatively increases the number of training objects. One may thus improve the performance of linear classifiers which often suffer from the curse of dimensionality.

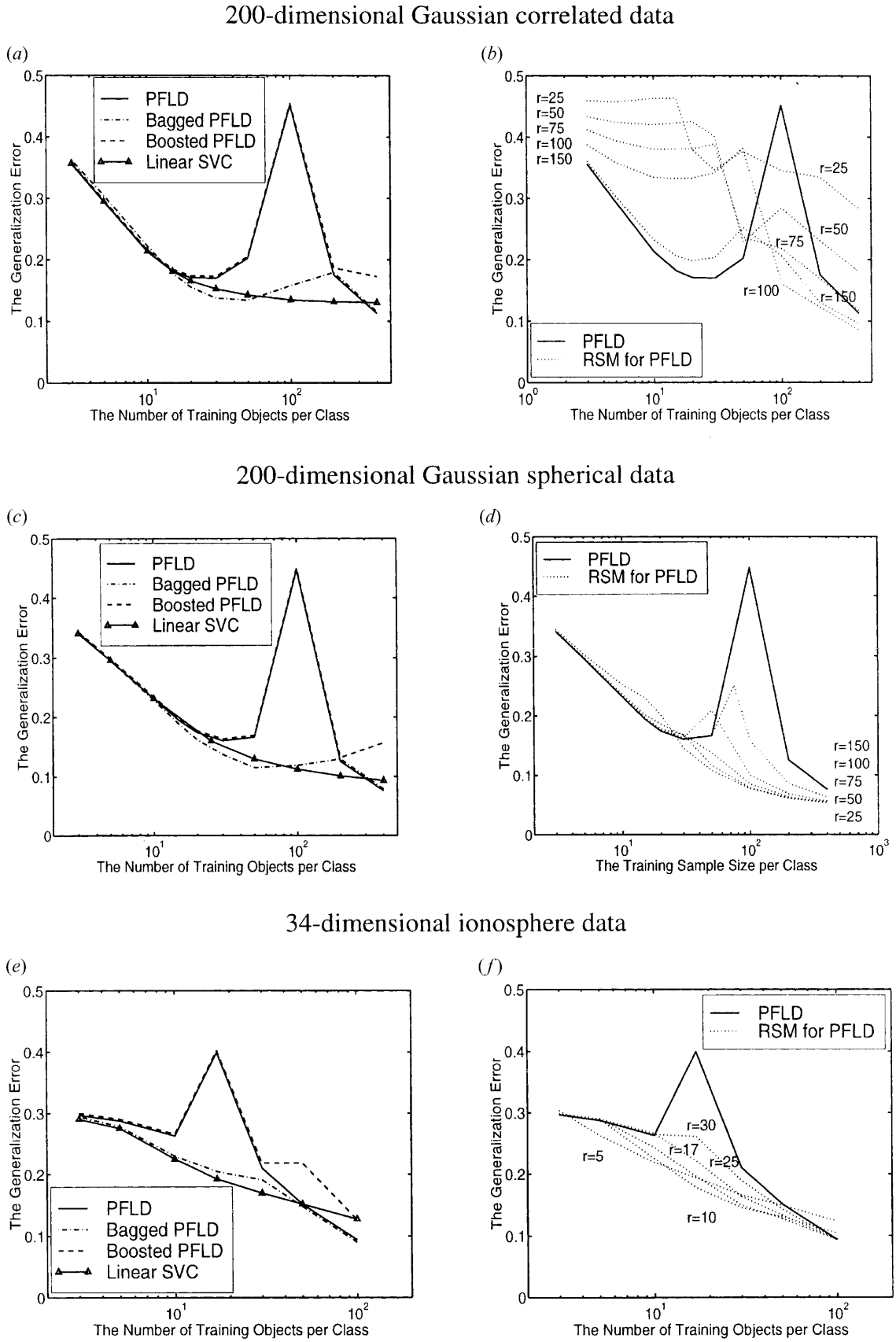

Fig. 10. Performance of bagging, boosting and the RSM ($B = 250$) for the PFLD.

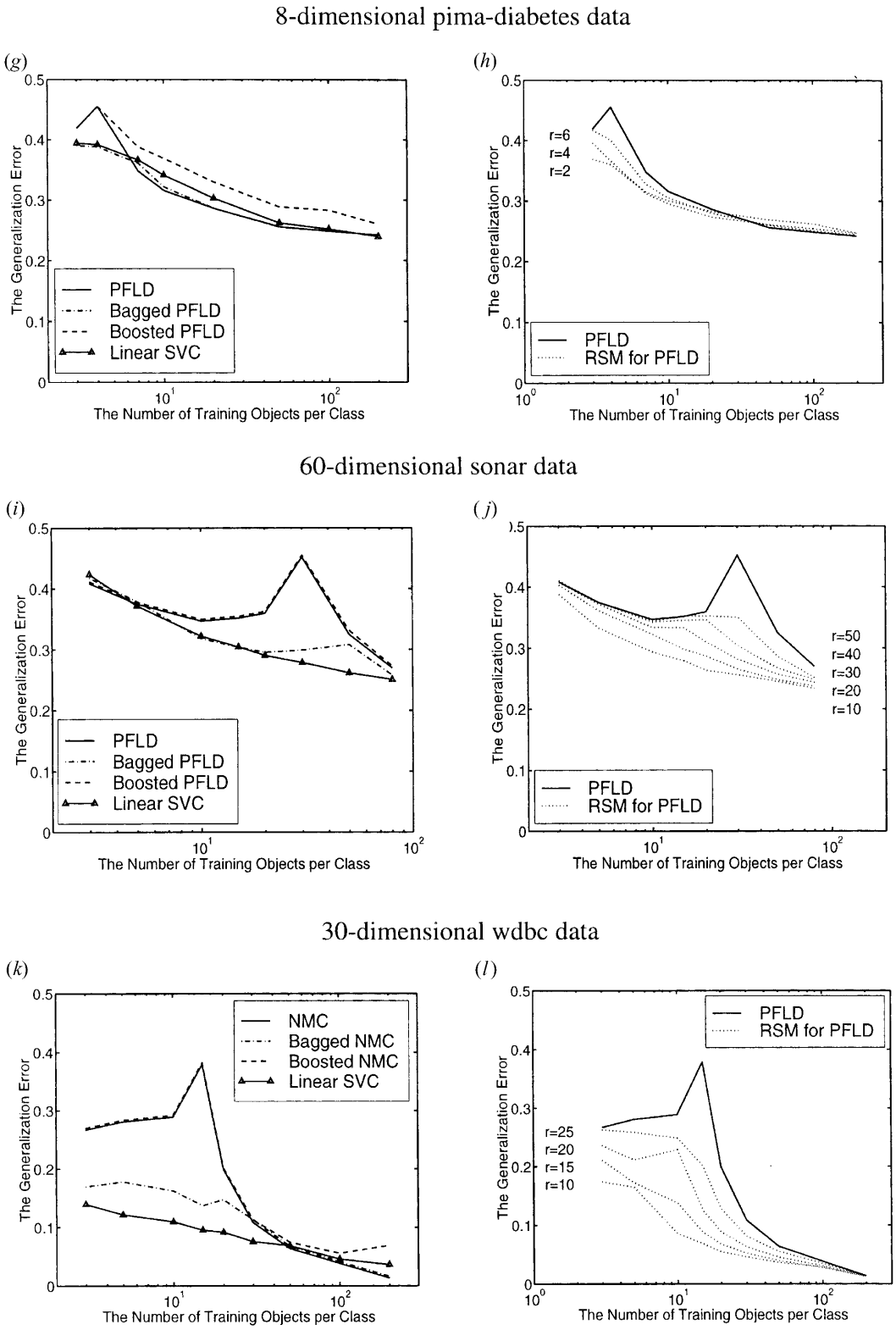

Fig. 10. Continued.

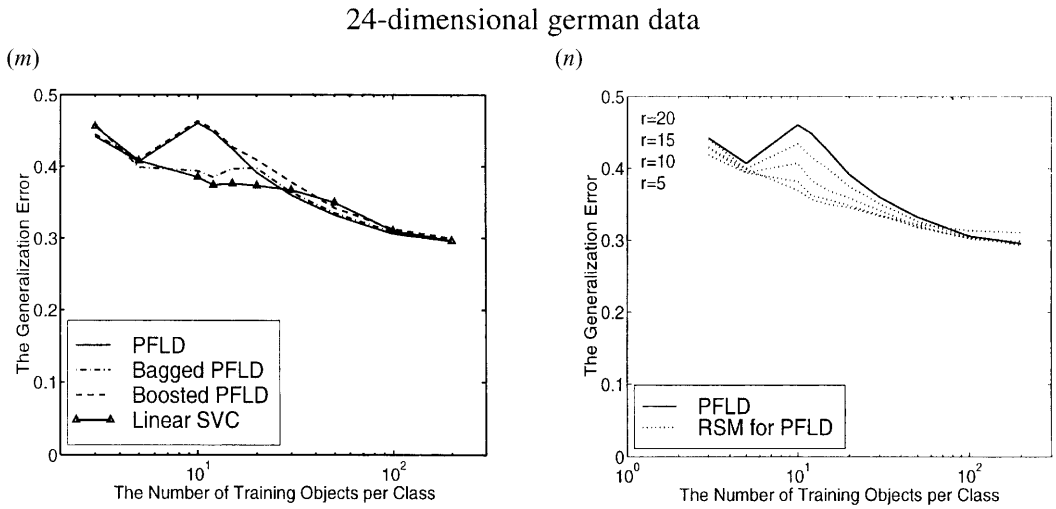

Fig. 10. Continued.

THE USER GUIDE			
the linear classifier	**the training sample size**		
	small ($n<p$)	**critical ($n\sim p$)**	**large ($n>3p$)**
FLD (a decreasing learning curve)	RSM	RSM	Combining is useless
PFLD (a learning curve with the peaking behaviour)	Combining is useless	bagging, RSM	Combining is useless
NMC (a non-decreasing learning curve)	Combining is useless	bagging	boosting
NMC (a decreasing learning curve)	RSM	RSM	Combining is useless

Fig. 11. Possible uses of combining techniques for linear classifiers.

The RSM has a shifting effect on the generalisation error of the base classifier in the direction of the generalisation error obtained on larger training sample sizes (see Fig. 6). Therefore, the RSM may improve the performance of classifiers having decreasing learning curves. Examples of such classifiers are the FLD and the PFLD (which is equivalent to the FLD for training sample sizes larger than the data dimensionality).

The usefulness of the RSM, as well as the efficient dimensionality of random subspaces, also depends upon the level of redundancy in the data feature space [6,19].

A summary of the possible usefulness of combining techniques for linear classifiers is given in Fig. 11. The abbreviations and notifications used in the table have the following meanings: FLD is the Fisher linear discriminant; NMC is the nearest mean classifier; PFLD is the Pseudo Fisher linear discriminant; RSM is the Random Subspace Method; n is the training sample size; p is the data dimensionality.

The success of bagging, boosting and the RSM depends upon many factors, including the training sample size, the choice of a base classifier (DT, PFLD, NMC, or others), the exact way in which the training set is modified, the choice of the combining rule [16] and, finally, on the data distribution and the potential ability of the chosen base classifier to solve the problem. Thus, it becomes quite difficult to establish universal criteria to predict the usefulness of combining techniques. Obviously, this question needs more investigation in future.

Acknowledgements

This work is supported by the Foundation for Applied Sciences (STW) and the Dutch Organization for Scientific Research (NWO).

References

1. Jain AK, Chandrasekaran B. Dimensionality and sample size considerations in pattern recognition practice. In: Krishnaiah PR, Kanal LN (eds) Handbook of Statistics, vol 2. North-Holland, Amsterdam, 1987; 835–855

2. Friedman JH. Regularized discriminant analysis. J Am Statistical Assoc 1989; 84: 165–175
3. An G. The effects of adding noise during backpropagation training on a generalization performance. Neural Computation 1996; 8: 643–674
4. Breiman L. Bagging predictors. Machine Learning J 1996; 24(2): 123–140
5. Freund Y, Schapire RE. Experiments with a new boosting algorithm. Proceedings 13th International Conference on Machine Learning 1996; 148–156
6. Ho TK. The Random subspace method for constructing decision forests. IEEE Trans Pattern Analysis and Machine Intelligence 1998; 20(8): 832–844
7. Efron B, Tibshirani R. An Introduction to the Bootstrap. Chapman & Hall, New York, 1993
8. Schapire RE, Freund Y, Bartlett P, Lee W. Boosting the margin: a new explanation for the effectiveness of voting methods. Ann Statistics 1998; 26(5): 1651–1686
9. Breiman L. Arcing classifiers. Ann Statistics 1998; 26(3): 801–849
10. Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning 2000; 40(2): 139–157
11. Breiman L. Random forests – random features. Technical Report 567, University of California, Berkley, 1999
12. Avnimelech R, Intrator N. Boosted mixture of experts: an ensemble learning scheme. Neural Computation 1999; 11: 483–497
13. Schwenk H, Bengio Y. Boosting neural networks. Neural Computation 2000; 12: 1869–1887
14. Alkoot FM, Kittler J. Population bias control for bagging k-NN experts. Proceedings Sensor Fusion: Architectures, algorithms and applications V. Orlando, FL, April 2001
15. Skurichina M, Duin RPW. Bagging for linear classifiers. Pattern Recognition 1998; 31(7): 909–930
16. Skurichina M, Duin RPW. The role of combining rules in bagging and boosting. In: Ferri FJ, Inesta JM, Amin A, Pudil P (eds) Advances in Pattern Recognition (Proceedings Joint International Workshops SSPR'2000 and SPR'2000, Alicante, Spain, August/September 2000). Lecture Notes in Computer Science, vol 1876. Springer-Verlag, Berlin, 2000; 631–640
17. Skurichina M, Duin RPW. Boosting in linear discriminant analysis. In: Kittler J, Roli F (eds) Multiple Classifier Systems (Proceedings First International Workshop MCS 2000, Cagliari, Italy). Lecture Notes in Computer Science, vol 1857. Springer-Verlag, Berlin, 2000: 190–199
18. Pekalska E, Skurichina M, Duin RPW. Combining Fisher linear discriminants for dissimilarity representations. In: Kittler J, Roli F (eds) Multiple Classifier Systems (Proceedings First International Workshop MCS 2000, Cagliari, Italy). Lecture Notes in Computer Science, vol 1857. Springer-Verlag, Berlin, 2000; 117–126
19. Skurichina M, Duin RPW. Bagging and the random subspace method for redundant feature spaces. In: Kittler J, Roli F (eds) Multiple Classifier Systems (Proceedings Second International Workshop MCS 2001, Cambridge, UK). Lecture Notes in Computer Science, vol 2096. Springer-Verlag, Berlin, 2001; 1–10
20. Fukunaga K. Introduction to Statistical Pattern Recognition. Academic Press, 1990
21. Raudys Š, Pikelis V. On dimensionality, sample size, classification error and complexity of classification algorithm in pattern recognition. IEEE Trans Pattern Analysis and Machine Intelligence 1980; 2(3): 242–252
22. Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: bagging, boosting and variants. Machine Learning 1999; 36(1/2): 105–142
23. Raudys Š, Duin RPW. On expected classification error of the Fisher linear classifier with pseudo-inverse of the covariance matrix. Pattern Recognition Letters 1998; 19(5–6): 385–392
24. Cortes C, Vapnik V. Support-vector networks. Machine Learning 1995; 20: 273–297
25. Ho TK. Nearest neighbours in random subspaces. Proceedings 2nd International Workshop on Statistical Techniques in Pattern Recognition. Sydney, Australia, 1998; 640–648
26. Freund Y, Schapire RE. A decision-theoretic generalization of online learning and an application to boosting. J Computer and System Sciences 1997; 55(1): 119–139
27. Vapnik VN. The Nature of Statistical Learning Theory. Springer-Verlag, Berlin, 1995
28. Blake CL, Merz CJ. UCI Repository of machine learning databases. http:// www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, Department of Information and Computer Science, 1998

Robert P.W. Duin studied applied physics at Delft University of Technology in the Netherlands. In 1978 he received a PhD for a thesis on the accuracy of statistical pattern recognisers. In his research he included various aspects of the automatic interpretation of measurements, learning systems and classifiers. Between 1980 and 1990 he studied and developed hardware architectures and software configurations for interactive image analysis. At present he is an associate professor of the Faculty of Applied Sciences of Delft University of Technology. His research interest is in the design and evaluation of learning algorithms for pattern recognition applications. This includes in particular neural network classifiers, support vector classifiers and classifier combining strategies. Recently, he started to study alternative object representations for classification, and thus became interested in the use of relational methods for pattern recognition.

Marina Skurichina studied applied mathematics at Vilnius State University in Lithuania. From 1989 to 1996 she worked as a research fellow and later as a PhD student in the department of Data Analysis at the Institute of Mathematics and Informatics in Vilnius, Lithuania. In 1996 she joined the Pattern Recognition Group of the Faculty of Applied Sciences of Delft University of Technology in the Netherlands. She received her PhD for a thesis on the stabilising weak classifiers in 2001. She is the author of about 20 scientific papers. Her scientific interests include regularisation and combining techniques in discriminant analysis.

Correspondence and offprint requests to: M. Skurichina, Pattern Recognition Group, Department of Applied Physics, Delft University of Technology, P.O. Box 5046, Delft 2600 GA, The Netherlands. E-mail: marina@ph.tn.tudelft.nl