On hybrid classification using model assisted posterior estimates

Anil. K. Ghosh[a,*], Fred Godtliebsen[b]

[a] *Theoretical Statistics and Mathematics Unit, Indian Statistical Institute, 203, Barrackpore Trunk Road, Kolkata 700108, India.*

[b] *Department of Mathematics and Statistics, University of Tromsø, N-9037 Tromsø, Norway.*

Abstract

Traditional parametric and nonparametric classifiers used for statistical pattern recognition have their own strengths and limitations. While parametric methods assume some specific parametric models for density functions or posterior probabilities of competing classes, nonparametric methods are free from such assumptions. So, when these model assumptions are correct, parametric methods outperform nonparametric classifiers, especially when the training sample is small. But, violations of these assumptions often lead to poor performance by parametric classifiers, where nonparametric methods work well. In this article, we make an attempt to overcome these limitations of parametric and nonparametric approaches and combine their strengths. The resulting classifiers, denoted the hybrid classifiers, perform like parametric classifiers when the model assumptions are valid, but unlike parametric classifiers, they also provide safeguards against possible deviations from parametric model assumptions. In this article, we propose some multiscale methods for hybrid classification, and their performance is evaluated using several simulated and benchmark data sets.

Keywords: Bayes rule, cross-validation, LDA, misclassification rates, multiscale analysis, nearest neighbor classification, QDA, stacking.

1. Introduction

Because of their simplicity and ease of calculation, linear discriminant analysis (LDA, see e.g., Fisher, 1936), quadratic discriminant analysis (QDA) and nearest neighbor (NN)

*Corresponding author

Email addresses: akghosh@isical.ac.in (Anil. K. Ghosh), fred.godtliebsen@uit.no (Fred Godtliebsen)

classification (see e.g., Cover and Hart 1967) are arguably the most popular classifiers in the field of statistical pattern recognition. However, each of these popular methods has its own strengths and limitations. Though Fisher (1936) introduced LDA from a different perspective, it is optimal when the underlying distributions are Gaussian. QDA also assumes parametric (Gaussian) models $f_j(\cdot) = f_j(\boldsymbol{\theta}_j, \cdot),\ j = 1, 2, \ldots, J$ for population densities and uses the training data to estimate the model parameters $\boldsymbol{\theta}_j$. Estimates of posterior probabilities, $p^1(\cdot \mid \mathbf{x})$, can be obtained from the formula $p^1(j \mid \mathbf{x}) = \pi_j \hat{f}_j(\mathbf{x}) / [\sum_{k=1}^{J} \pi_k \hat{f}_k(\mathbf{x})]$, where $\hat{f}_j(\cdot) = f_j(\hat{\boldsymbol{\theta}}_j, \cdot)$ is the parametric estimate of f_j, and π_j is the prior probability of the j-th class. When the π_js are not known, one can estimate them using sample proportions of different classes. Instead of modeling the f_js, some parametric classifiers like logistic discriminant analysis assume parametric models for posterior probabilities and directly estimate them. Clearly, the performance of these parametric classifiers depends on the validity of associated model assumptions. If they are valid, parametric methods perform substantially better than nonparametric methods, especially when the training sample is small. But violations of these assumptions often lead to poor performance by parametric classifiers.

NN-classifier, on the other hand, is nonparametric and free from all parametric model assumptions. So, when violations in model assumptions lead to poor performance by parametric methods, it can still work well. However, it is not above all limitations. Since the convergence of nonparametric estimates is rather slow, it suffers from statistical instability when we have a small training set. Moreover, nonparametric methods like the NN-classifier do not use any information about the parametric structure of population distributions. Therefore, even when one has some insights about the distribution structure of underlying populations, that important information is not utilized to modify the classification rule.

So, it would be ideal if one can develop a classifier that performs like a parametric classifier when the model assumptions hold and like a nonparametric classifier when they do not hold. In this article, we develop such classifiers by hybridizing LDA and QDA with the NN-classifier. The main goal of this hybridization is to overcome the limitations of parametric and nonparametric approaches and combine their strengths. Though similar hybrid methods have been proposed in the literature for density estimation (see e.g., Olkin and

Spiegelman, 1987; Hjort and Glad, 1995; Jones *et. al.*, 1995; Hjort and Jones, 1996; Hoti and Holmstrom, 2004) and regression (see e.g., Glad, 1998), they are somewhat missing for classification problems. One can develop a hybrid classifier using hybrid density estimates (see e.g., Chaudhuri *et. al.*, 2009). But, there are several parametric (e.g., logistic discriminant analysis) and nonparametric classifiers (e.g., NN-classifier, classification trees, neural networks) that give estimates of posterior probabilities but do not yield any density estimate. Even for discriminative methods like support vector machines, one can find estimates of posterior probabilities (see e.g., Ghosh and Chaudhuri, 2005), but no estimates of density functions. Keeping that in mind, in this article, we present a hybridization technique that does not involve any density estimation. Because of computational simplicity, here we have used LDA, QDA and the NN-classifier for hybridization. But, in principle, the proposed method can also be used for hybridizing other parametric and nonparametric classifiers. Our hybridization technique is much simpler than those proposed in Chaudhuri *et. al.* (2009), and unlike their methods, it can even be used for simultaneous hybridization of several parametric and nonparametric classifiers.

2. Hybridization of parametric and nonparametric classifiers

We begin with a simple method of hybridization, where we consider a class of hybrid posterior estimates $S = \left\{ p^{\lambda}(\cdot \mid \mathbf{x}) = \lambda\, p^{1}(\cdot \mid \mathbf{x}) \;+\; (1-\lambda)\, p^{0}(\cdot \mid \mathbf{x});\; 0 \leq \lambda \leq 1 \right\}$ and use the training data to choose a member from S (or equivalently a value of λ) to be used for classification. Here p^1 and p^0 stand for parametric (LDA or QDA) and nonparametric (NN) estimates, respectively. Clearly, this includes the possibility of selecting the parametric (in case of $\lambda = 1$) or the nonparametric classifier (in case of $\lambda = 0$). In that sense, it is model selection over a larger class. Note that when parametric model assumptions are correct, $p^1(\cdot \mid \mathbf{x})$ usually have $\sqrt{n}$ convergence (where n is the training sample size) to the true posterior $p(\cdot \mid \mathbf{x})$, but $p^0(\cdot \mid \mathbf{x})$ has much slower rate of convergence, which becomes even slower as the dimension increases. In such cases, if we can choose λ that converges to 1 at an appropriate rate, $p^{\lambda}(\cdot \mid \mathbf{x})$ also have $\sqrt{n}$ convergence to $p(\cdot \mid \mathbf{x})$. As a result, the hybrid classifier performs as good as the parametric classifier and much better than

the nonparametric classifier, especially when the training sample is small compared to the dimension of the data. On the contrary, when these model assumptions are not correct, $p^1(\cdot \mid \mathbf{x})$ does not converge to $p(\cdot \mid \mathbf{x})$, but $p^0(\cdot \mid \mathbf{x})$ converges to $p(\cdot \mid \mathbf{x})$ as before. In this case, if we can choose λ that shrinks to 0 at an appropriate rate, the hybrid classifier can match the performance of the nonparametric classifier. So, hybridization can improve the performance of parametric and nonparametric classifiers if λ is chosen appropriately. However, in addition to λ, here we need to choose the number of neighbors (k) involved in nonparametric estimation of posterior probabilities. Existing theoretical results (see e.g., Loftsgaarden and Quesenberry 1965; Cover and Hart 1967) suggest that k should tend to infinity and k/n should tend to zero as the training sample size n increases. But in practice, one needs to estimate it from the training data. So, instead of S, here we consider the class $S^* = \big\{\ p^{\lambda,k}(\cdot \mid \mathbf{x}) = \lambda\ p^1(\cdot \mid \mathbf{x}) + (1-\lambda)\ p^{0,k}(\cdot \mid \mathbf{x});\ 0 \le \lambda \le 1,\ k = 1, 2, \ldots, (n-1)\big\}$, where $p^{0,k}(\cdot \mid \mathbf{x})$ stands for the k-NN estimate of $p(\cdot \mid \mathbf{x})$. If k_j out of k neighbors of $\mathbf{x}$ come from the j-th population ($j = 1, 2, \ldots, J$), the ratio k_j/k is taken as $p^{0,k}(j \mid \mathbf{x})$. To choose the optimum values of k and λ, one can use resampling techniques like cross-validation (CV) (see e.g., Hastie *et. al.*, 2009) or likelihood cross-validation (LCV) (see e.g., Silverman 1986). In LCV, optimum k and λ are chosen by maximizing the loglikelihood function $L(\lambda, k) = \sum_{i=1}^{n} log\ p_{-i}^{\lambda,k}(c_i \mid \mathbf{x}_i)$, where c_i is the class label of $\mathbf{x}_i$, and $p_{-i}^{\lambda,k}$ denotes the hybrid posterior estimate computed from the training data using the leave-one-out method when $\mathbf{x}_i$ is not used as a data point. In CV, we choose λ and k that minimize the cross-validation estimate of error rate $\hat{\Delta}(\lambda, k) = \frac{1}{n} \sum I\{\arg\max_j\ p_{-i}^{\lambda,k}(j \mid \mathbf{x}_i) \neq c_i\}$. Note that LCV and CV can be used to choose the value of k in usual NN-classification as well. In future, we will refer to these two NN-classifiers as NN-LCV and NN-CV, respectively. Similarly, two hybrid classifiers will be denoted by Hybrid-LCV and Hybrid-CV, respectively.

Now, let us consider a simple example with two bivariate normal distributions having the same location parameter $(0, 0)$ but different dispersion matrices $\mathbf{I}_2$ (the 2×2 identity matrix) and $4\mathbf{I}_2$. In this classification problem, $\mathbf{x}'\mathbf{x} = \frac{16}{3} log 2$ is the optimal class boundary. We generated 50 observations from each class (see the scatter plot in Figure 1) to form the training sample and used different classification methods to estimate this class boundary.

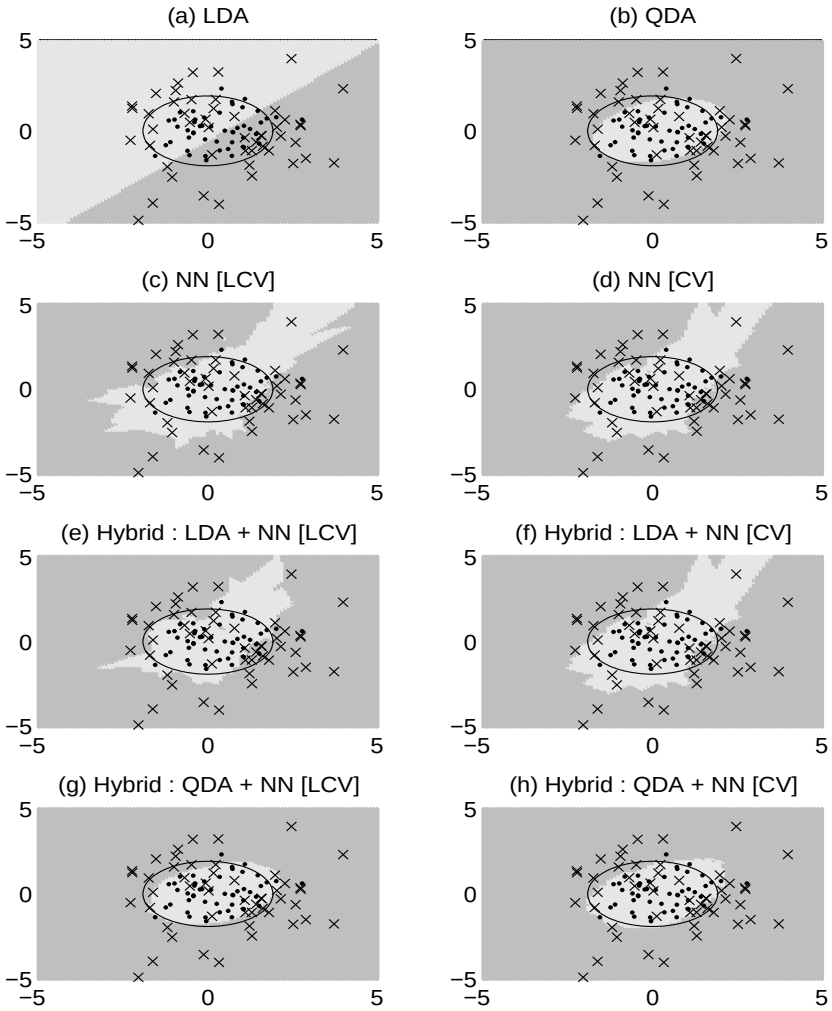

Figure 1: Class boundaries estimated by parametric, nonparametric and hybrid classifiers.

Note that LDA assumes the population distributions to be normal with different location parameters but the same dispersion matrix. On the contrary, here the competing population distributions have the same location but different scatter matrices. Due to this deviation from model assumption, LDA led to a poor estimate of the class boundary (see Figure 1a). However, for NN-classifiers, these estimates (see Figures 1c and 1d) were of much better shapes. When we hybridized LDA with NN-classifiers, very small values of λ were selected (0.28 and 0.10, respectively, for LCV and CV), which is desirable in such cases. So, in spite of model mis-specification, hybrid classifiers performed like NN-classifiers and much better than LDA (see Figures 1e and 1f). In this example, QDA makes the correct

model assumption, and it yielded an estimate of the class boundary close to the optimal one (indicated by the black curve). Hybrid classifiers also took the advantage of this correct model assumption when we hybridized QDA with NN-classifiers. They had similar estimates of the class boundary (see Figures 1g and 1h) as obtained in QDA. Note that in this case, the selected values of λ were much higher (0.91 and 0.72 for LCV and CV, respectively), which one should expect when the parametric model assumptions are correct.

2.1. Results on simulated examples

We used six simulated data sets to evaluate the performance of hybrid classifiers. In the first five examples, generating equal number of observations from competing classes, we formed 500 training and test sets of size 100 and 200, respectively. In Example-6, 75% of the observations were generated from class-1 and the rest from class-2. Except for Example-5, in all other cases, we considered classification problem between two competing classes. Average test set error rates of Hybrid-CV and Hybrid-LCV over these 500 trials are reported in Table 1 along with their corresponding standard errors. For Hybrid-CV, we had multiple minimizers of $\hat{\Delta}(\lambda, k)$ in some cases. Among them, we considered those having the smallest value of k, and then the one with the smallest value of λ was selected. However, one can choose any one of these optimizers, and usually that does not lead to any visible difference in the final result. Error rates of the Bayes classifier, LDA, QDA, NN-LCV and NN-CV are also reported to facilitate comparison. Throughout this section, training sample proportions of the competing classes are taken as their prior probabilities.

We begin with the example with two bivariate normal distributions discussed earlier (call it Example-1). We have observed that this is an ideal set up for QDA, but since there is no difference in locations of the two classes, LDA is expected to have poor performance. We observed the same phenomenon in our experiment. While LDA misclassified almost half of the test cases, QDA yielded an average error rate (27.53%) close to the Bayes risk (26.38%). NN-LCV and NN-CV had average error rates around 33%. When we hybridized LDA with NN-classifier, in spite of model mis-specification, hybrid methods could match the performance of NN-classifiers. However, when we chose the right parametric model, hybrid

classifiers had much better performance. Hybridization of QDA and NN-classifiers yielded average error rates close to that of QDA and the Bayes classifier.

In Example-2, two competing classes again differ only in their scales. In class-1, X_1 and X_2 are distributed as $X_1 = R\,cos\theta$ and $X_2 = R\,sin\theta$, for $R \sim U(0,1)$ and $\theta \sim U(0, 2\pi)$ being statistically independent. In class-2, R and θ have the same distributions, but X_1 and X_2 are defined as $X_1 = 3R\,cos\theta$ and $X_2 = 3R\,sin\theta$. Here also, NN-LCV and NN-CV performed much better than LDA, and hybridizing LDA with NN-classifiers, we obtained error rates similar to those of NN-classifiers. QDA once again outperformed the NN-classifiers, but hybridization of QDA with NN-classifiers yielded even lower error rates than QDA.

The importance of hybrid classification becomes more transparent if we add five $N(0,1)$ variables as noise to the data generated in Example 2 (call it Example-3). In this high dimensional setting, statistical instability of nonparametric methods becomes more clear. While QDA was less affected (average error rate = 24.89%) due to this noise, its effect on NN-classifiers was substantially higher. NN-LCV and NN-CV had average error rates of 40.67% and 37.41%, respectively. However, hybridization of QDA and NN-classifiers yielded average error rates even lower than that of QDA. LDA had almost 50% error rate in this example, but hybridizing LDA with NN-classifiers, we could achieve much lower error rates. Note that in Example-1, when the population distributions were normal, it was not possible to beat QDA, but in Examples 2 and 3, Hybrid-CV led to significant improvement (at 5% level) over QDA. One should also notice that in all these three examples, QDA had standard errors much smaller than NN classifiers, which shows its better statistical stability. Table 1 clearly shows that hybrid classifiers were as stable as QDA in all these examples.

In Example-4, each class is an equal mixture of two bivariate normal distributions each having the same dispersion matrix with diagonal entries (1,1) and the off-diagonal entry -0.75. For class-1, location parameters of these two distributions are (10,10) and (12,12), whereas those for class-2 are (11,11) and (13,13). In this example, both LDA and QDA had average error rates close to 44%, but NN-classifiers yeilded error rates around 17%. However, unlike parametric methods, hybrid classifiers did not get much affected by model mis-specification, and they yielded error rates comparable to that of NN-classifiers.

Table 1 : Error rates (in %) of different classifiers on simulated examples and their standard errors.

	Example-1	Example-2	Example-3	Example-4	Example-5	Example-6
Bayes risk	26.38	16.67	16.67	11.79	14.23	16.81
LDA	48.99 (0.17)	46.54 (0.17)	49.81 (0.16)	44.32 (0.12)	44.01 (0.17)	24.09 (0.05)
QDA	27.53 (0.14)	20.76 (0.13)	24.89 (0.15)	43.58 (0.12)	31.79 (0.16)	17.83 (0.10)
NN(LCV)	33.38 (0.20)	26.86 (0.18)	40.57 (0.24)	17.11 (0.13)	28.91 (0.23)	24.68 (0.03)
NN(CV)	33.01 (0.19)	25.78 (0.17)	37.41 (0.18)	17.02 (0.14)	25.57 (0.18)	22.65 (0.12)
Hybridization of LDA and Nearest neighbor						
Hybrid(LCV)	34.20 (0.21)	26.26 (0.17)	40.65 (0.22)	17.65 (0.15)	26.45 (0.19)	23.21 (0.10)
Hybrid(CV)	34.30 (0.21)	26.02 (0.18)	40.99 (0.21)	17.01 (0.14)	25.46 (0.18)	22.74 (0.11)
Hybridization of QDA and Nearest neighbor						
Hybrid(LCV)	27.73 (0.14)	20.66 (0.13)	24.87 (0.15)	17.64 (0.15)	24.78 (0.18)	17.89 (0.10)
Hybrid(CV)	28.69 (0.15)	19.94 (0.13)	23.91 (0.15)	17.44 (0.14)	23.00 (0.16)	18.47 (0.11)

So far, we have considered some examples, where the parametric classifier (QDA) is either better (Examples 1-3) or worse (Example-4) than the nonparametric classifier over the entire measurement space. In most of these cases, our hybrid classifiers performed as good as the better of the parametric and the nonparametric classifiers. Only in Examples 2 and 3, Hybrid-CV outperformed both of them. Now, we consider an example (call it Example-5), where QDA is superior to the NN classifiers in one part of the measurement space, whereas the NN classifiers are better in the other part. A combination of Example-2 and Example-4 is used to construct such an example with four competing classes. In this example, NN-classifiers had lower error rates than LDA and QDA. But hybridizing QDA with NN classifiers, we achieved even lower errors rates. While NN-LCV and NN-CV had average error rates of 28.91% and 25.57%, those for Hybrid-LCV and Hybrid-CV were 24.78% and 23.00%, respectively. We had some improvements over the error rates of NN-classifiers even when LDA was used for hybridization.

Example 6 deals with the two classes as in Example 1, but here the training and the test sets contain 75% observations from class-1. However, this unbalancedness did not make any change in our findings. Hybridization of QDA and NN classifier led to the error rates close to that of QDA, which was the best among the parametric and nonparametric classifiers.

2.2. Results on benchmark data sets

We analyze 18 benchmark data sets for further illustration of proposed methods. Among them, there are two vowel recognition data sets, one containing 10 dimensional observations

from 11 classes, and the other containing bivariate observations from 10 classes. We will refer to them as the vowel data and the 2D-vowel data, respectively. The latter one was generated by Petersen and Barney (1952). Salmon data can be found in Johnson and Wichern (1992). The rest of the data sets and their descriptions are available either at UCI Machine Learning Repository (http://www.ics.uci.edu/∼mlearn) or at CMU Data Archive (http://lib.stat.cmu.edu). In the case of biomedical data, we ignored the observations with missing values and carried out our analysis with the remaining 194 cases. A brief summary of these data sets is given in Table 2. Unless mentioned otherwise, throughout this section, we use Gaussian distributions as parametric models, and hybrid classifiers are developed by hybridizing LDA or QDA with NN-classifiers. In some of these data sets, the measurement variables are not of comparable units and scales. So, in all these cases, we use the Mahalanobis distance (with the moment based estimate of the pooled dispersion matrix) for finding nearest neighbors. For data sets having specific training and test sets (see Table 2), test set error rates of different classifiers are reported in Table 3. When a classifier led to an error rate Δ, its standard error was computed as $\sqrt{\Delta(1-\Delta)/n_t}$, for n_t being the size of the test set. In case of other data sets, we formed 500 training and test sets by randomly partitioning the data. Sizes of these training and test samples for different data sets are reported in Table 2, and average test set error rates of different methods over these 500 partitions are reported in Table 3 along with their corresponding standard errors. In all these cases, training sample proportions of different classes are used as their prior probabilities.

In some of these data sets, parametric methods (either LDA or QDA or both) had better performance than NN-classifiers. In such cases, the performance of hybrid classifiers was comparable to the corresponding parametric classifier and better than NN-classifiers. For instance, in cases of biomed and diabetes data, NN classifiers had significantly higher error rates than QDA, but the error rates of hybrid classifiers were close to that of QDA. On the other hand, in some other cases (e.g., letter recognition data and vowel data), both LDA and QDA had error rates much higher than NN-classifiers. Clearly, in these cases, the underlying distributions were far from being normal. But, in spite of invalid model assumptions, hybrid classifiers could match the performance of NN classifiers. In the case of Sonar data, they

Table 2 : Brief description of benchmark data sets.

Data sets	d	J	Train	Test	Data sets	d	J	Train	Test	Data sets	d	J	Train	Test
Salmon	2	2	50	50	Crab	5	4	100	100	Vehicle	18	4	400	446
Synthetic•	2	2	250	1000	Pima Indian	8	2	400	368	Waveform	21	3	3000	2000
2D-vowel•	2	10	338	333	Vowel•	10	11	528	462	WDBC	30	2	300	269
Biomed	4	2	100	94	Wine	13	3	100	78	Satimage•	36	6	4435	2000
Iris	4	3	75	75	Letter•	16	26	16000	4000	Sonar	60	2	150	58
Diabetes	5	3	100	45	Kangaroo	18	2	80	21	Control chart	60	6	450	150

• Data sets have specific training and test sets.

Table 3 : Error rates (in %) of different classifiers on benchmark data sets and their standard errors.

Data sets	LDA	QDA	Likelihood CV			Cross-validation		
			NN	LDA+NN	QDA+NN	NN	LDA+NN	QDA+NN
Salmon	8.21 (0.14)	7.69 (0.13)	8.46 (0.15)	8.56 (0.14)	8.09 (0.14)	8.97 (0.15)	8.83 (0.15)	8.62 (0.15)
Synthetic	10.80 (0.98)	10.20 (0.96)	10.00 (0.95)	10.40 (0.97)	10.30 (0.96)	11.70 (1.02)	11.70 (1.02)	10.20 (0.96)
2D-Vowel	25.26 (2.38)	19.83 (2.19)	22.82 (2.30)	19.24 (2.16)	19.83 (2.19)	18.06 (2.11)	19.25 (2.16)	19.25 (2.16)
Biomed	15.72 (0.13)	12.66 (0.12)	18.70 (0.16)	15.75 (0.13)	12.69 (0.12)	17.72 (0.15)	16.43 (0.15)	13.38 (0.13)
Iris	2.51 (0.07)	2.78 (0.07)	2.75 (0.08)	2.69 (0.07)	2.87 (0.07)	2.98 (0.08)	2.96 (0.08)	2.93 (0.08)
Diabetes	10.48 (0.18)	9.41 (0.18)	14.65 (0.31)	10.61 (0.18)	9.33 (0.18)	10.04 (0.18)	9.69 (0.19)	9.60 (0.18)
Crab	5.75 (0.09)	6.36 (0.09)	6.66 (0.10)	5.99 (0.09)	6.42 (0.09)	6.80 (0.10)	6.61 (0.10)	6.12 (0.10)
Pima	23.37 (0.07)	25.98 (0.08)	25.66 (0.07)	23.41 (0.07)	25.68 (0.09)	25.76 (0.07)	23.98 (0.07)	25.61 (0.08)
Vowel	55.62 (2.31)	52.81 (2.32)	48.27 (2.32)	46.54 (2.32)	45.02 (2.31)	46.75 (2.32)	46.75 (2.32)	46.75 (2.32)
Wine	2.00 (0.06)	2.47 (0.09)	2.40 (0.07)	2.23 (0.07)	2.27 (0.08)	2.34 (0.07)	2.32 (0.07)	2.30 (0.07)
Letter	30.94 (0.73)	12.42 (0.52)	4.22 (0.32)	4.77 (0.34)	4.84 (0.34)	4.22 (0.32)	4.57 (0.33)	4.59 (0.33)
Kangaroo	29.70 (0.44)	43.43 (0.42)	36.22 (0.35)	31.30 (0.41)	36.48 (0.38)	36.52 (0.36)	31.66 (0.40)	36.37 (0.37)
Vehicle	22.19 (0.06)	16.42 (0.07)	22.04 (0.07)	20.46 (0.07)	16.77 (0.09)	21.93 (0.08)	20.84 (0.07)	16.49 (0.07)
Waveform	14.18 (0.03)	15.18 (0.03)	22.62 (0.04)	14.18 (0.03)	15.18 (0.03)	15.75 (0.03)	14.39 (0.03)	15.20 (0.03)
WDBC	4.71 (0.05)	4.65 (0.05)	13.68 (0.14)	8.21 (0.11)	8.37 (0.11)	9.63 (0.09)	5.44 (0.05)	4.52 (0.05)
Satimage	16.03 (0.82)	14.11 (0.78)	21.65 (0.92)	16.42 (0.83)	15.75 (0.81)	16.49 (0.83)	14.18 (0.78)	14.23 (0.78)
Sonar	26.84 (0.24)	31.91 (0.24)	25.64 (0.25)	25.04 (0.23)	24.19 (0.25)	27.47 (0.26)	25.68 (0.25)	27.33 (0.26)
Control chart	2.78 (0.05)	25.72 (0.17)	7.00 (0.09)	2.77 (0.05)	6.76 (0.09)	7.20 (0.09)	3.46 (0.06)	5.28 (0.08)

had lower error rates than that of both parametric and nonparametric classifiers. Mainly motivated by the normality of underlying distributions, in all these examples, we used LDA and QDA as parametric classifiers. But this normality assumption may not be valid in some cases, where we can further improve the performance of hybrid classifiers by choosing more appropriate parametric models. For instance, in the case of synthetic data, if we assume each of the two populations to be a mixture of two bivariate normal distributions and use the EM algorithm (see e.g., MacLachlan and Krishnan 1997) to estimate the model parameters, Hybrid-LCV and Hybrid-CV can achieve error rates of 9.1% and 9.2%, respectively.

3. Multiscale approach in hybrid classification

In the previous section, we considered the class of models S^* and chose one of them for classification of all observations. However, this way of selecting only one classifier ignores the uncertainty involved in model selection. Further, in addition to depending on the training data, a good choice of k and λ may depend on the observation to be classified. Therefore, in practice, instead of using a fixed (λ, k) over the entire measurement space, it may be more useful to consider the results for different choices of k and λ and then aggregate them judiciously. Here k is the smoothing parameter that controls the scale of smoothing involved in nearest neighbor estimation of posterior probabilities. If k gets larger, $p^{0,k}$ and hence $p^{\lambda,k}$ tend to be smoother in some sense. The parameter λ also controls the smoothness of the hybrid estimate. If λ is small, $p^{\lambda,k}$ behaves like a nonparametric estimate, which is supposed to be a local estimate, whereas for bigger λ, it behaves like a parametric estimate, which is global in some sense. So, the local variation or the roughness of $p^{\lambda,k}$ is expected to be smaller as λ increases. Therefore, different values of λ and k can be viewed as different scales of smoothing, and the aggregated classifier can be referred to as the multiscale classifier.

The usefulness of multiscale analysis has been discussed in the literature in the context of function estimation (see e.g., Chaudhuri and Marron 1999, 2000; Godtliebsen *et. al.*, 2002) and classification (see e.g., Holmes and Adams 2002; Ghosh *et. al.*, 2005; Ghosh *et. al.*, 2006). One popular way to aggregate the results of different classifiers is to use the weighted average of posterior probabilities. The aggregated classifier is given by $\mathsf{d}_A(\mathbf{x}) = \arg\max_j \sum_{\lambda,k} w(\lambda,k)\, p^{\lambda,k}(j \mid \mathbf{x})$, where $w(\lambda,k)$ is the weight ($\sum_{\lambda,k} w(\lambda,k) = 1$) assigned to the classifier indexed by (λ,k). Following Ghosh *et. al.* (2005), one can adopt the multi-scale version of cross-validation, and use the Gaussian-type weight function

$$w(\lambda,k) = C \exp\left\{ -\frac{1}{2} \frac{\hat{\Delta}(\lambda,k) - \hat{\Delta}(\lambda_0,k_0)}{\sqrt{\hat{\Delta}(\lambda_0,k_0)(1-\hat{\Delta}(\lambda_0,k_0))/n}} \right\}$$

that decreases with the cross-validation error rate $\hat{\Delta}(\lambda,k)$ of a classifier at an exponential rate. Here $\hat{\Delta}(\lambda_0,k_0) = \min_{\lambda,k} \hat{\Delta}(\lambda,k)$, and C is a normalizing constant. Note that $\hat{\Delta}(\lambda_0,k_0)$ and $\{\hat{\Delta}(\lambda_0,k_0)\ (1-\hat{\Delta}(\lambda_0,k_0))\}/n$ can be viewed as estimates for the mean and the variance

of the empirical misclassification rate of the hybrid classifier with the best choice of λ and k, when it is used to classify n independent observations. Also note that this estimated variance converges to zero as n tends to infinity. So, when the training sample size is very large, it puts almost all weights on the classifiers having the lowest cross-validation error. If this classifier is unique, multiscale method performs almost like a single scale method. But when the sample size is not so large, due to high stochastic variation of the cross-validation estimate of error rate, the single scale method often fails to select the best model even when such a model (which is uniformly better than other models over the entire measurement space) exists. Multi-scale method takes care of this model uncertainty and aggregate the results obtained by several good classifiers by putting higher weights on them. The use of the Gaussian type weight function also helps us to appropriately weigh down the poor classifiers (classifiers with poor choices of λ and k) by putting almost zero weights on them. The multiscale method based on this Gaussian weight function led to fairly good results in our examples. Our empirical experience also suggests that the final result is not very sensitive to the choice of the weight function as long as it decreases at an exponential rate.

For multiscale version of LCV, one can assign weights to different values of (λ, k) depending on the corresponding likelihood $L(\lambda, k)$. Given the training data $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, their class labels $C = (c_1, c_2, \ldots, c_n)$ and the value of (λ, k), $p^{\lambda,k}(. \mid \mathbf{x})$ can be viewed as the conditional probability $p(. \mid \mathbf{x}, \mathbf{X}, C, \lambda, k)$. Clearly, these posterior probability estimates and the resulting classifier depend on the selected model (i.e. the values of λ and k). To remove this model uncertainty, we compute $p(\cdot \mid \mathbf{x}, \mathbf{X}, C)$, which is free from k and λ, and given by

$$\begin{aligned} p(\cdot \mid \mathbf{x}, \mathbf{X}, C) &= \textstyle\sum_k \int p(. \mid \mathbf{x}, \mathbf{X}, C, \lambda, k)\ \pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C)\ d\lambda \\ &= \textstyle\sum_k \int p^{\lambda,k}(. \mid \mathbf{x})\ \pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C)\ d\lambda, \end{aligned}$$

where $\pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C)$ denotes the posterior distribution of (λ, k) given $\mathbf{x}, \mathbf{X}$ and C. If $\xi(\lambda, k)$ denotes the prior distribution of (λ, k), we have $\pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C) = C_0\ \xi(\lambda, k)\ L(\lambda, k)$, where C_0 is a constant that does not depend on k and λ. So, one can forget about C_0 and use the usual quadrature formula to compute $p(\cdot \mid \mathbf{x}, \mathbf{X}, C)$. However, sometimes it is computationally advantageous to use the Markov Chain Monte Carlo (MCMC) algorithm

(see e.g., Gilks *et. al.*, 1996). Using MCMC, one can generate sufficiently many observations $(\lambda^{(1)}, k^{(1)})$, $(\lambda^{(2)}, k^{(2)})$, $\ldots$, $(\lambda^{(M)}, k^{(M)})$ from $\pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C)$ and then approximate the integral by $\frac{1}{M}\sum_{m=1}^{M} p^{\lambda^{(m)},k^{(m)}}(\cdot \mid \mathbf{x})$. In this article, we have always generated a Markov chain consisting of 11000 observations, and leaving the first 1000 for burning, the rest have been used to compute posterior probability estimates of different classes. Since $\pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C)$ does not depend on $\mathbf{x}$, one does not need to generate MCMC samples repeatedly for different observations, and this leads to substantial saving in computing time. However, $\pi(\lambda, k \mid \mathbf{x}, \mathbf{X}, C)$ depends on the prior $\xi(\lambda, k)$. In this article, for all data analytic purpose, we have used uniform prior both for λ and k assuming their independence. This prior is non-informative and gives no preference to any value of (λ, k). For generating the MCMC sequence, we consider the proposal distribution $N(\lambda, \sigma) \times U(k, k \pm 1, k \pm 2, k \pm 3)$ with appropriate boundary corrections, where σ is chosen during the simulation to have 30% acceptance. Here also, if the training sample size is vary large compared to the dimension of the data, one can show that this multiscale version of LCV puts all almost all weights on the model having the highest likelihood, and in that case, it behaves almost like the single scale version. However, when the sample size is not so large, it usually outperforms its single scale analog by considering the results of several good classifiers. This multiscale method performed quite well in our data sets. In all these cases, we used both quadrature and MCMC methods, and the error rates of these two methods were almost the same.

3.1. Comparison among single scale and multiscale hybrid classifiers

Here we analyze the benchmark data sets used in Section 2 to compare the performance of multiscale hybrid classifiers with their single scale analogs (Hybrid-LCV and Hybrid-CV). We ran the multiscale classifiers on the same training and test sets used before, and their test error rates are reported in Table 4 along with their corresponding standard errors. Error rates of Hybrid-LCV and Hybrid-CV are also reported to facilitate comparison. Note that we have two multiscale hybrid classifiers, one based on LCV and the other one based on CV. In future, we will refer to them as MSLCV and MSCV, respectively. In Section 2, we observed that though in some cases, hybrid classifiers outperformed both parametric and

Table 4 : Error rates (in %) of different classifiers and their standard errors.

Hybridization of LDA and nearest neighbor classifiers									
	Salmon	Synthetic	2D-Vowel	Biomed	Iris	Diabetes	Crab	Pima Indian	Vowel
Select-LCV	8.65 (0.15)	10.00 (0.95)	25.26 (2.38)	15.81 (0.13)	2.81 (0.07)	11.16 (0.20)	6.00 (0.10)	23.74 (0.08)	48.27 (2.32)
Hybrid-LCV	8.56 (0.14)	10.40 (0.97)	19.24 (2.16)	15.75 (0.13)	2.69 (0.07)	10.61 (0.18)	5.99 (0.09)	23.41 (0.07)	46.54 (2.32)
MSLCV	8.10 (0.14)	9.20 (0.91)	18.64 (2.13)	15.89 (0.13)	2.44 (0.06)	10.60 (0.18)	5.77 (0.09)	23.20 (0.07)	46.75 (2.32)
Select-CV	9.01 (0.15)	11.70 (1.02)	18.06 (2.11)	17.38 (0.15)	2.99 (0.08)	10.04 (0.18)	6.80 (0.10)	24.26 (0.09)	46.75 (2.32)
Hybrid-CV	8.83 (0.15)	11.70 (1.02)	19.25 (2.16)	16.43 (0.15)	2.96 (0.08)	9.69 (0.19)	6.61 (0.10)	23.98 (0.07)	46.75 (2.32)
MSCV	8.06 (0.13)	10.70 (0.98)	21.37 (2.25)	16.17 (0.14)	2.50 (0.07)	10.48 (0.18)	5.73 (0.09)	23.17 (0.07)	46.75 (2.32)
	Wine	Letter	Kangaroo	Vehicle	Waveform	WDBC	Satimage	Sonar	Control chart
Select-LCV	2.33 (0.07)	30.94 (0.73)	36.20 (0.36)	21.99 (0.07)	14.18 (0.03)	8.26 (0.11)	16.03 (0.82)	25.66 (0.23)	2.76 (0.05)
Hybrid-LCV	2.23 (0.07)	4.77 (0.34)	31.30 (0.41)	20.46 (0.07)	14.18 (0.03)	8.21 (0.11)	16.42 (0.83)	25.04 (0.23)	2.77 (0.05)
MSLCV	1.82 (0.06)	4.77 (0.34)	29.79 (0.43)	20.35 (0.07)	14.51 (0.03)	5.69 (0.06)	17.25 (0.84)	24.40 (0.23)	2.77 (0.05)
Select-CV	2.34 (0.07)	4.22 (0.32)	34.42 (0.39)	21.93 (0.07)	14.18 (0.03)	5.05 (0.05)	16.03 (0.82)	25.68 (0.25)	2.83 (0.06)
Hybrid-CV	2.32 (0.07)	4.57 (0.33)	31.66 (0.40)	20.84 (0.07)	14.39 (0.03)	5.44 (0.05)	15.75 (0.81)	25.63 (0.25)	3.46 (0.06)
MSCV	1.81 (0.06)	4.04 (0.31)	29.42 (0.43)	21.40 (0.07)	14.27 (0.03)	5.25 (0.06)	16.49 (0.83)	23.91 (0.24)	2.82 (0.05)
Hybridization of QDA and nearest neighbor classifiers									
	Salmon	Synthetic	2D-Vowel	Biomed	Iris	Diabetes	Crab	Pima Indian	Vowel
Select-LCV	7.93 (0.14)	10.20 (0.96)	19.83 (2.19)	15.33 (0.17)	2.88 (0.08)	10.50 (0.22)	6.48 (0.09)	25.82 (0.08)	52.81 (2.32)
Hybrid-LCV	8.09 (0.14)	10.30 (0.96)	19.83 (2.19)	12.69 (0.12)	2.87 (0.07)	9.33 (0.18)	6.42 (0.09)	25.68 (0.09)	45.02 (2.31)
MSLCV	7.76 (0.13)	10.10 (0.95)	19.83 (2.19)	12.67 (0.12)	2.67 (0.07)	9.30 (0.18)	6.29 (0.09)	25.66 (0.08)	45.45 (2.32)
Select-CV	8.65 (0.15)	10.20 (0.96)	19.83 (2.19)	13.42 (0.13)	2.97 (0.08)	10.04 (0.18)	6.70 (0.09)	25.95 (0.08)	46.75 (2.32)
Hybrid-CV	8.62 (0.15)	10.20 (0.96)	19.25 (2.16)	13.38 (0.13)	2.93 (0.08)	9.60 (0.18)	6.12 (0.10)	25.61 (0.08)	46.75 (2.32)
MSCV	7.72 (0.13)	10.10 (0.95)	19.83 (2.19)	12.66 (0.12)	2.69 (0.07)	9.24 (0.18)	6.14 (0.09)	25.61 (0.08)	46.75 (2.32)
	Wine	Letter	Kangaroo	Vehicle	Waveform	WDBC	Satimage	Sonar	Control chart
Select-LCV	2.46 (0.08)	12.42 (0.52)	36.22 (0.36)	20.98 (0.12)	15.18 (0.03)	10.98 (0.13)	14.11 (0.78)	24.26 (0.25)	7.03 (0.09)
Hybrid-LCV	2.27 (0.08)	4.84 (0.34)	36.48 (0.38)	16.77 (0.09)	15.18 (0.03)	8.37 (0.11)	14.18 (0.78)	24.17 (0.25)	6.76 (0.09)
MSLCV	1.57 (0.07)	4.84 (0.34)	35.70 (0.38)	16.37 (0.07)	15.18 (0.03)	4.70 (0.05)	14.52 (0.79)	27.33 (0.29)	6.72 (0.10)
Select-CV	2.38 (0.07)	4.22 (0.32)	36.64 (0.37)	16.50 (0.07)	15.18 (0.03)	5.07 (0.05)	14.11 (0.78)	27.63 (0.27)	7.20 (0.09)
Hybrid-CV	2.30 (0.07)	4.59 (0.33)	36.37 (0.37)	16.49 (0.07)	15.20 (0.03)	4.52 (0.05)	14.23 (0.78)	27.35 (0.26)	5.28 (0.08)
MSCV	1.21 (0.06)	4.29 (0.32)	39.13 (0.45)	16.36 (0.07)	15.11 (0.03)	4.71 (0.05)	14.08 (0.78)	25.17 (0.26)	4.87 (0.08)

nonparametric classifiers, in most of the cases, they performed as good as the better of these two methods. So, instead of going for hybridization, one may be tempted to use either LCV or CV to select either the parametric or a nonparametric classifier and use it for classification of all test cases. Table 4 also reports the error rates of these two methods, which are referred to as Select-LCV and Select-CV, respectively.

Table 4 clearly shows the effectiveness of the multiscale approach in hybrid classification. In 53 out of 72 (18×4) cases, multiscale classifiers had the lowest error rates, and in 25 out of these 53 cases (written using grey colors in Table 4), differences between their error

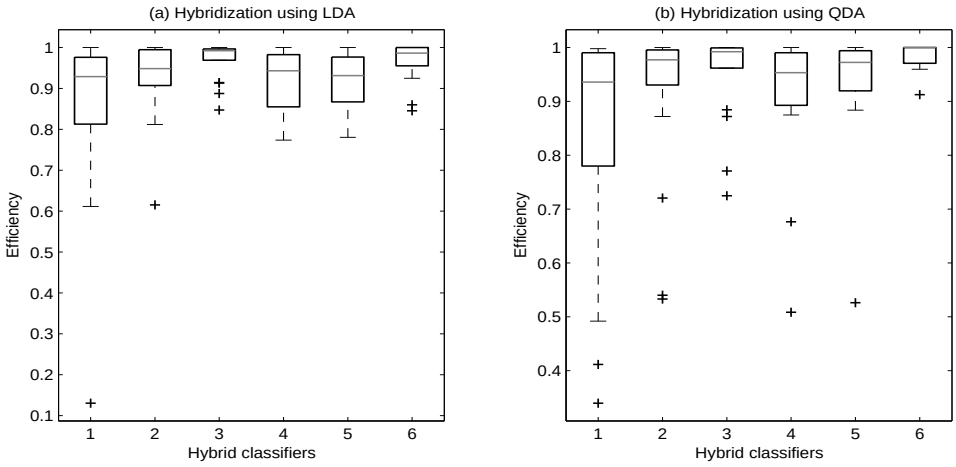

Figure 2: Efficiencies of different classifiers : [1] Select-LCV, [2] Hybrid-LCV, [3] MSLCV, [4] Select-CV, [5] Hybrid-CV, [6] MSCV.

rates and those of their competitors were found to be statistically significant at 5% level when the usual t-test was used for comparison. To compare the overall performance of these classifiers, following Chaudhuri *et. al.* (2009), we computed efficiencies of different methods across different data sets, and they are presented in Figure 2 using box plots. On a particular data set, the efficiency of the t-th classifier is defined as $e_t = \Delta_0/\Delta_t$, where Δ_t is the test set misclassification rate of t-th classifier and $\Delta_0 = \min \Delta_t$. Clearly, e_t takes the value 1 for the best classifier and $e_t < 1$ for other classifiers. Also, a small value of e_t indicates the lack of efficiency of the classifier t. Box plots in Figure 2 clearly shows the superiority of multiscale hybrid classifiers. It also strongly indicates that using hybrid classifiers is better than selecting one of the parametric and nonparametric classifiers.

3.2. Computationally efficient multiscale hybrid classifiers and their comparison with other methods of aggregation

Since $p^{\lambda,k}$ is a convex combination of p^1 and $p^{0,k}$, the multi-scale classifiers discussed above can also be expressed as $d_A(\mathbf{x}) = arg\max_j \left\{w_0\ p^1(j \mid \mathbf{x}) + \sum_k w_k\ p^{0,k}(j \mid \mathbf{x})\right\}$, where $w_0, w_1, \ldots$ are positive constants and $\sum_{k\geq 0} w_k = 1$. Therefore, instead of assigning weights to hybrid classifiers, one can assign weights directly to parametric and nonparametric classifiers to come up with a new method of aggregation or classifier combination. Note that the second

method requires less computation, and one can use that idea to develop alternative versions of our multiscale methods discussed earlier. In future, we will refer to these alternative versions of MSLCV and MSCV as MSLCV-2 and MSCV-2, respectively. There is a vast literature on the area of classifier combination (see Kittler *et. al.*, 1998 for discussion), but almost all aggregation methods can be broadly divided into two main categories. One class of methods use the same classifier on different subsamples to develop different classification rules and then aggregate them to arrive at the final classifier. Popular ensemble methods like bagging (see e.g., Breiman 1996b) and boosting (see e.g., Schapire *et. al.* 1998; Friedman, Hastie and Tibshirani 2000) belong to this class. Another class of methods use different classifiers on the same data set and then aggregate them. These classifier combination methods are known as stacking (see e.g., Wolpert, 1992; Breiman, 1996a). Ofcourse, there are other aggregation methods like cascading (see e.g., Alpaydin and Kayank, 1998; Kayank and Alpaydin, 2000) and gating (see e.g., Jacobs *et. al.*, 1991). In cascading, one uses a simple and computationally efficient classifier (like LDA or QDA) to all observations and then a relatively complex classifier (like k-NN) is used to classify those cases, which were not confidently (in terms of estimated posterior or otherwise) classified by the simple classifier. However, instead of a single NN classifier, here we work with a class of NN classifiers, and it is difficult to judge which one is more simple and to be used before. If one uses the 1-NN classifier before others, it gives posterior estimates either 0 or 1 and hence leaves no room for other classifiers to be used. Moreover, in some examples (e.g., Example-4 in Section 2), parametric classifiers misclassify many observations with high confidence (in terms of estimated posterior). Cascading will fail to correctly classify those observations. Gating on the other hand, adopts a probabilistic approach to decide which of the classifiers is to be used for classification of a specific observation. These probabilities (also known as weights) are dynamically adjusted during classification. This can be viewed as a locally adaptive version of Select-CV and Hybrid-CV (or Select-LCV and Hybrid-LCV) where different models may get selected for classification of different observations. However, this differs from our multiscale approach, where we do not choose any particular model (classifier) but consider the results of all classifiers to arrive at the final decision. Locally

adaptive versions of our multiscale methods will be discussed later in Section 3.4, and that is why here we do not discuss gating separately.

Note that unlike bagging and boosting, here we do not deal with different subsamples. From the above discussion, it is also quite clear that stacking is more relevant in our context. In stacking, the major issues are the choice of level-0 and level-1 classifiers and also the choice of the input variables for level-1 classification. In our case, these level-0 classifiers are well defined (i.e., parametric and nonparametric classifiers), and we want to choose a good level-1 classifier. Ting and Witten (1999) proposed to use posterior probability estimates of different classes obtained by level-0 classifiers as input features for level-1 generalization. For level-1 classification, they used the multiple linear regression (MLR) method under the non-negativity constraint on the regression coefficients as suggested by Breiman (1996a). Dzeroski and Zenko (2004) compared the performance of different stacking algorithms on several benchmark data sets and showed that MLR with posterior probability estimates usually performs better than most of other stacking algorithms, and its performance is usually better than the method based on cross-validation, especially when there are diversities among the level-0 classifiers. Since the parametric classifier and the nonparametric classifiers with different choices of k are expected to have reasonable diversities among themselves, here one can expect to have better performance using the stacking algorithm. In this article, we have used the MLR method of Ting and Witten (1999) for stacking, where the non-negativity constraint is imposed using the algorithm given in Lawson and Hansen (1995).

Since LDA, QDA and NN-classifiers are all stable classifiers, one cannot expect to have significant diversity in the decision rules if one of these classifiers is used on different subsamples (see e.g., Breiman, 1996b; Zhou and Yu, 2005). So, bagging or boosting is not a good option for our classifier combination. Moreover, bagging and boosting needs repetitive use of NN-classifiers on different subsamples, which increases the computing cost and also requires substantial memory space to keep track of these subsamples. However, one can still adopt the weight function used in bagging or boosting for aggregation. Note that bagging assigns equal weight to all classifiers, but like MSCV, boosting assigns different weights to different classifiers based on their misclassification rates. However, instead of an exponential

Table 5 : Average computing times (in seconds) for classification of all test cases.

Data sets	LDA	NN-LCV	NN-CV	Sel.-LCV	Sel.-CV	Hyb.-LCV	Hyb.-CV	MSLCV	MSCV	MSLCV-2	MSCV-2	Stacking
Salmon	0.0003	0.0014	0.0013	0.0016	0.0016	0.0103	0.0115	0.0425	0.0163	0.0020	0.0019	0.0022
Synthetic	0.0050	0.1098	0.1061	0.1139	0.1106	0.8373	0.9720	0.4846	1.5162	0.1133	0.1104	0.1512
2D-Vowel	0.0025	0.1005	0.1073	0.1026	0.1092	1.2400	2.9781	0.6879	3.9143	0.1045	0.1098	1.9893
Biomed	0.0013	0.0095	0.0089	0.0106	0.0098	0.0532	0.0487	0.1057	0.0704	0.0101	0.0097	0.0111
Iris	0.0012	0.0043	0.0039	0.0045	0.0042	0.0280	0.0317	0.0933	0.0492	0.0051	0.0045	0.0070
Diabetes	0.0012	0.0078	0.0061	0.0086	0.0070	0.0498	0.0623	0.1148	0.0869	0.0177	0.0168	0.0343
Crab	0.0016	0.0108	0.0107	0.0121	0.0119	0.0538	0.0792	0.1327	0.1250	0.0120	0.0123	0.0189
Pima	0.0085	0.1418	0.1394	0.1492	0.1473	5.0403	5.2136	1.4405	5.4991	0.1477	0.1467	0.2097
Vowel	0.0230	0.2503	0.2671	0.2694	0.2838	2.1470	5.2662	1.1587	7.7196	0.3552	0.3056	4.3016
Wine	0.0044	0.0141	0.0139	0.0181	0.0180	0.0558	0.0632	0.1148	0.0869	0.0177	0.0167	0.0343
Letter	0.7056	161.88	164.00	168.52	172.20	275.01	573.97	223.31	615.94	172.52	173.39	998.92
Kangaroo	0.0028	0.0069	0.0073	0.0088	0.0089	0.0347	0.0352	0.0792	0.0395	0.0096	0.0099	0.0108
Vehicle	0.0267	0.1573	0.1587	0.1781	0.1812	0.5316	0.9147	0.5998	1.0111	0.1992	0.2019	0.2608
Waveform	0.2031	7.2063	7.1989	7.2761	7.2744	11.348	12.200	10.446	12.907	7.3414	7.3296	7.7936
WDBC	0.0361	0.1212	0.1206	0.1420	0.1413	0.3512	0.3342	0.4046	0.3459	0.1573	0.1562	0.1617
Satimage	0.5172	17.621	17.625	18.469	18.547	25.703	31.437	25.750	32.125	19.031	18.991	23.985
Sonar	0.0379	0.0669	0.0665	0.0996	0.0986	0.1657	0.1626	0.2191	0.1781	0.1205	0.1197	0.1409
Control chart	0.1500	0.3386	0.3398	0.5349	0.5445	1.0428	1.6359	0.9303	1.7394	0.6911	0.6995	0.7944

weight function, it uses the log of the odd ratio (see e.g., Zhu *et. al.*, 2005 for details on multi-class adaboost algorithm). We can assign these logarithmic weights either to hybrid classifiers (as it was done in Section 3.1) or to parametric and nonparametric classifiers (as it was done in stacking, MSLCV-2 and MSCV-2). Based on that, we get two different versions of aggregation, and we will refer to them as LogWeight and LogWeight-2, respectively.

We used all these aggregation methods on 18 benchmark data sets, but instead of reporting their error rates in another table, to save space and to have better visualization, these results are summarized using box-plots in Figure 3. Average computing times of these methods are also reported in Table 5. Since the computing times of LogWeight and Logweight-2 were similar to that of MSCV and MSCV-2, we do not report them here. For MSLCV, we have reported the computing time for the method based on the MCMC technique, which was found to be computationally efficient than the quadrature method in most of the cases. From Figure 3, it is quite clear that in terms of error rate, MSCV-2 and MSCV had comparable performance. This result is quite encouraging because MSCV-2 is computationally efficient than MSCV (see Table 5), and it can be used for combining several parametric and

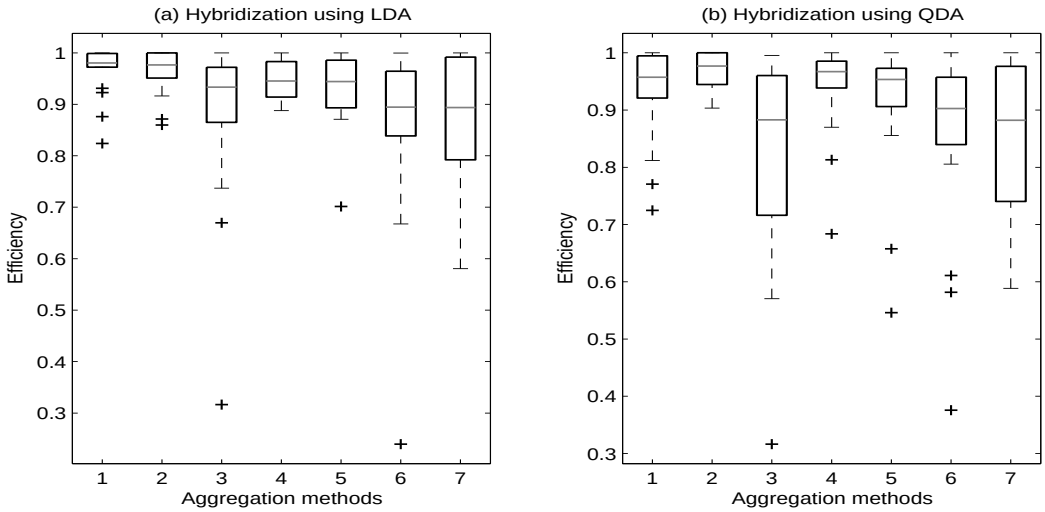

Figure 3: Efficiencies of different aggregation methods : [1] MSLCV, [2] MSCV, [3] MSLCV-2, [4] MSCV-2, [5] Stacking, [6] LogWeight, [7] LogWeight-2.

nonparametric classifiers, which may not be computationally feasible for MSCV. However, that was not the case for MSLCV-2. It had comparatively higher error rates than MSLCV. But that doesn't harm much since the MSLCV method based on the MCMC technique is not computationally very expensive (see Table 5). The stacking classifier based on MLR had higher computing time than MSCV-2 and MSLCV-2, especially when there are several competing classes. Though stacking performed quite well in these examples, MSCV-2 had an edge over stacking both in terms of computing time and average misclassification rate. The logarithmic weight function could not weigh down the poor classifiers properly, and as a result, LogWeight and LogWeight-2 often had significantly higher error rates. This matches with the findings of Ghosh *et. al.* (2006), and we will not further investigate these logarithmic weighting schemes in this article. The performance of the equal weightage schemes was even worse, and we do not report them here.

3.3. Aggregation of several parametric and nonparametric classifiers

Our aggregation methods can be used for hybridizing several parametric and nonparametric classifiers as well. Let us consider a set of T classifiers, and let $p_t^{\beta_t}(\cdot \mid \mathbf{x})$ be the posterior probability estimate obtained from the t-th classifier ($t = 1, 2, \ldots, T$), when β_t is

Table 6: Error rates (in %) of different aggregation methods and their corresponding standard errors.

	Salmon	Synthetic	2D-vowel	Biomed	Iris	Diabetes	Crab	Pima	Vowel
MSLCV	7.88 (0.13)	10.10 (0.95)	19.83 (2.18)	12.71 (0.11)	2.35 (0.06)	9.00 (0.18)	5.66 (0.08)	25.22 (0.09)	46.75 (2.32)
MSLCV-2	8.01 (0.14)	10.10 (0.95)	19.83 (2.18)	15.13 (0.13)	2.49 (0.07)	11.19 (0.19)	6.06 (0.09)	23.79 (0.08)	52.81 (2.32)
MSCV-2	8.01 (0.13)	9.80 (0.94)	18.94 (2.15)	14.79 (0.14)	2.43 (0.07)	9.52 (0.18)	6.16 (0.09)	24.69 (0.08)	46.75 (2.32)
Stacking	8.49 (0.14)	10.20 (0.96)	20.73 (2.22)	12.12 (0.12)	2.82 (0.08)	9.49 (0.18)	5.99 (0.09)	23.82 (0.08)	46.75 (2.32)
	Wine	Letter	Kangaroo	Vehicle	Waveform	WDBC	Satimage	Sonar	Control chart
MSLCV	1.07 (0.05)	4.41 (0.32)	36.48 (0.34)	16.62 (0.08)	14.98 (0.04)	4.77 (0.05)	14.24 (0.78)	26.86 (0.25)	3.15 (0.05)
MSLCV-2	1.99 (0.07)	12.42 (0.52)	36.01 (0.31)	20.67 (0.07)	15.18 (0.03)	7.84 (0.11)	16.03 (0.82)	27.07 (0.25)	2.77 (0.05)
MSCV-2	1.96 (0.06)	3.93 (0.31)	34.32 (0.33)	17.43 (0.09)	14.71 (0.04)	4.70 (0.05)	14.10 (0.78)	25.93 (0.24)	3.12 (0.06)
Stacking	2.36 (0.07)	3.83 (0.30)	33.11 (0.38)	16.93 (0.08)	14.52 (0.03)	4.83 (0.05)	13.30 (0.76)	26.57 (0.25)	8.14 (0.08)

used as the associated smoothing parameter (ignore β_t for parametric methods like LDA and QDA, which do not involve any smoothing parameter). Now, for aggregation of these T classifiers, we can consider the class of models $S^{**} = \{p^{\boldsymbol{\lambda},\boldsymbol{\beta}}(\cdot \mid \mathbf{x}) = \sum_{t=1}^{T} \lambda_t \; p_t^{\beta_t}(\cdot \mid \mathbf{x}); \; \lambda_t, \beta_t \geq 0 \; \forall \; t, \; \sum \lambda_t = 1\}$. Clearly, this class may contain a large number of models, and it could be computationally difficult to use LCV or cross-validation to find the best candidate in this class. For the same reason, it could be difficult to use MSCV. But we do not have this problem for MSLCV that explores the model space using the MCMC technique. Computationally efficient aggregation methods like stacking, MSCV-2, MSLCV-2 can also be used. Unlike MSLCV, instead of putting weights on all hybrid models in S^{**}, they put weights only on parametric and nonparametric models (as discussed in Section 3.2) to compute the aggregated posteriors for different classes. We have seen that in some of the benchmark data sets, hybridization of LDA yielded better performance than hybridization of QDA, whereas in some other cases, the latter one was better. One can test the validity of the homoscedastic assumption (same variance-covariance structure in different populations) made by LDA to decide which parametric method is to be used for hybridization. However, if classification is our prime concern, we can bypass this testing problem, and as a safeguard, use LDA, QDA and NN-classifiers simultaneously for hybridization.

Table 6 presents the error rates of these aggregation methods, and these results are summarized using box plots in Figure 4(a). They clearly show that MSCV-2 is a good option as a computationally efficient alternative, but one should avoid using MSLCV-2. This is consistent with what we observed before. Among different aggregation methods,

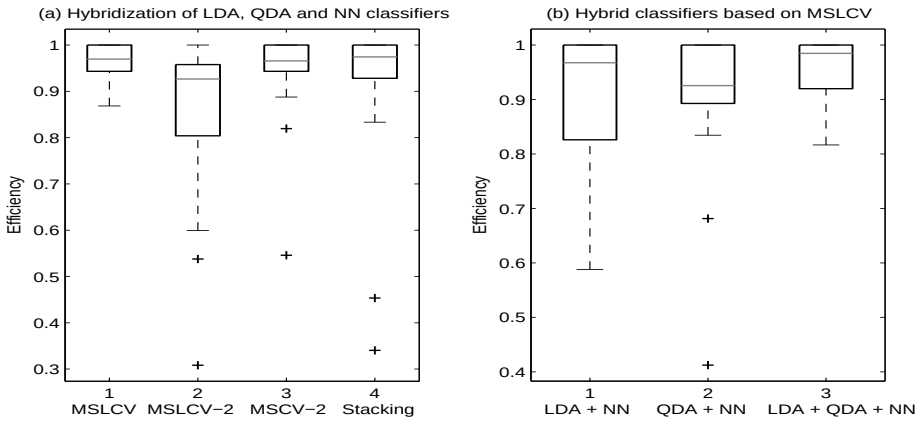

Figure 4: Hybridization of LDA, QDA and NN classifiers : efficiencies of different aggregation methods.

overall performance of MSLCV and MSCV-2 was better than the rest. Though MSLCV had a slight edge over MSCV-2 in terms of error rate, MSCV-2 had substantial advantage in terms of computing time. In terms of average (median) efficiency, stacking and MSCV-2 were comparable, but the latter one had a clear advantage in terms stability (see Figure 4(a)). It showed more stable performance than stacking across these benchmark data sets and required less computing time as well. Box plots in Figure 4(b) also show that for MSLCV, we could achieve much better performance when both LDA and QDA are simultaneously used for hybridization. We observed the same phenomenon also for the other three aggregation methods considered here.

3.4. Locally adaptive aggregation

Instead of using same weights over the entire region, sometimes it is more reasonable to use different weights for aggregation in different parts of the measurement space. This could be helpful if the parametric classifier is better in one part of the measurement space and the nonparametric method is better in another part. For example, let us consider the 'easy' and the 'difficult' examples considered in Hastie *et. al.* (2009, p. 468). We combined these two examples as described below to generate observations from two competing classes. We generated some 10 dimensional random vectors $\mathbf{X} = (X_1, X_2, \ldots, X_{10})$, where $X_1, X_2, \ldots, X_{10}$ are independent $U(0, 1)$ variables. For the first half of the data, we assigned $\mathbf{X}$ to class-1 if $X_1 > 0.5$ and to class-2 otherwise. For the rest half, we assigned $\mathbf{X}$ to class-1

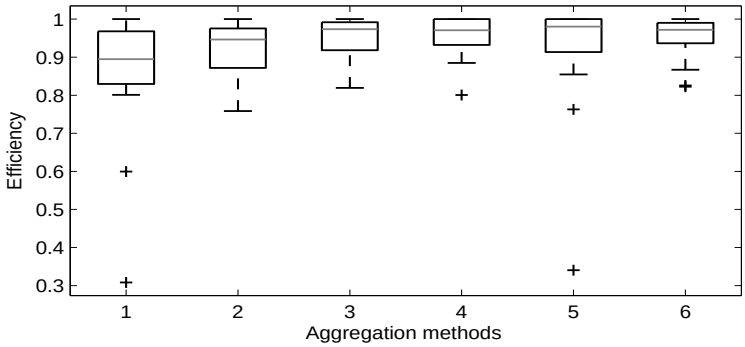

Figure 5: Efficiencies of locally adaptive and non-adaptive weighing scheme : [1] MSLCV-2, [2] MSLCV-2 (local), [3] MSCV-2, [4] MSCV-2 (local), [5] Stacking, [6] Stacking (local).

if $(X_1 - 0.5)(X_2 - 0.5)(X_3 - 0.5) > 0$, to class-2 otherwise. For each $\mathbf{X}$ in the second half, we increased the value of X_4 by 10 so that the generated data form two distinct clusters. Now, in one part of the measurement space, since the Bayes classifier is linear, LDA is expected to work better than NN-classifiers, whereas in the other part, NN-classifiers are expected to outperform LDA. So, if we hybridize LDA with the NN-classifiers, it would be appropriate to put more weight on LDA in the first part and small weight on it in the other part. In order to have this locally adaptive nature in aggregation, for classification of $\mathbf{x}$, different weights $\omega_{\mathbf{X}}(\mathbf{x}_i)$ can be assigned to different data points $\mathbf{x}_i$ depending on their distances $\|\mathbf{x}_i - \mathbf{x}\|$ from $\mathbf{x}$. Using these weights, one can compute the weighted error rate or the weighted likelihood function for different classifiers to put locally adaptive weights on them. Similarly, for an adaptive version of stacking, we can perform weighted linear regression under the non-negativity constraint. Here, we used a simple method for computing ω. We assigned $\omega = 1$ for all points in a neighborhood of $\mathbf{x}$ and $\omega = 0$ for the rest. Following Hastie and Tibshirani (1996), we chose a neighborhood containing $\max(50, n/5)$ observations subject to a maximum of 200.

We generated 500 training and test sets each of size 100 and 200 respectively, and computed the average error rates for adaptive and non-adaptive versions of stacking, MSCV-2 and MSLCV-2 (these three methods were chosen because of their computational efficiency). While non-adaptive versions of MSLCV-2, MSCV-2 and stacking had average error rates of 31.84%, 31.26% and 31.40%, those for their adaptive versions were 31.09%, 30.12% and

30.77%, respectively. In view of corresponding standard errors (0.15% for all these methods), the improvement was statistically significant in all these cases. When we considered it as a four class problem, average error rates for non-adaptive versions were 28.86%, 28.62%, 28.47% respectively, whereas those for adaptive versions were 28.59%, 27.67% and 28.21%.

We observed the same phenomenon in Example-5 discussed in Section-2. Recall that hybridization of QDA and NN-classifiers led to the best result in this example. Hybrid-LCV and Hybrid-CV yielded average error rates of 24.78% and 23.00% respectively. In this example, non-adaptive multiscale methods could not provide significant improvement in terms of error rates, but adaptive versions of MSLCV-2, MSCV-2 and stacking yielded average error rates of 22.76%, 22.18% and 21.86%, respectively, with corresponding standard errors of 0.19%, 0.17% and 0.16%.

We used these locally adaptive aggregation techniques to analyze the 18 benchmark data sets used in this article, and the results are summarized using box plots in Figure 5. This figure shows that the overall performance of the adaptive versions were somewhat better than their non-adaptive analogs, especially in the case of MSLCV-2.

4. Concluding remarks

In this article, we have developed and studied hybrid classification methods to improve the performance of parametric and nonparametric classifiers. When the underlying distributions are close to the assumed parametric models, hybrid methods usually perform better than nonparametric classifiers and match the performance of parametric methods. But unlike parametric classifiers, hybrid methods provide automatic safeguards against parametric model mis-specifications. When the true population distributions are far from the assumed parametric models, hybrid classifiers perform substantially better than parametric methods and yield error rates either lower or comparable to that of nonparametric classifiers. Also, in some case, specially when the parametric classifier is better in one part of the measurement space and the nonparametric classifiers are better in other part, hybrid methods can outperform both of them. Using several simulated and benchmark data sets, in this article, we have amply demonstrated these important features of hybrid classifiers.

The aggregation methods discussed in Section 3 are simple and easy to implement. They combine the results obtained by different classifiers, which are expected to have reasonable diversities among themselves. So, intuitively it seems more advantageous to use the aggregation technique, and our analysis of benchmark data sets also supports this intuition. The MSLCV algorithm based on the MCMC technique can also be used for classification with missing values. In the missing value problem, one either ignores the full observation or uses EM type algorithm to replace the missing value by the expected value of the variable. But instead of fixing this value, one can consider the results for different choices of the missing value and aggregate the results. In the same spirit, it can be used for semi-supervised classification as well. However, its performance on such problems needs to be investigated.

Acknowledgement

We are thankful to three anonymous referees for their careful reading of earlier versions of the paper and providing us with several helpful comments.

References

[1] Alpaydin, E. and Kayank, C. (1998) Cascading classifiers. *Kybernetika*, **34**, 369-374.

[2] Breiman, L. (1996a) Stacked regressions. *Machine Learning*, **24**, 49-64.

[3] Breiman, L. (1996b) Bagging predictors. *Machine Learning*, **24**, 123-140.

[4] Chaudhuri, P. and Marron, J. S. (1999) SiZer for exploration of structures in curves. *J. Amer. Statist. Assoc.* **94**, 807-823.

[5] Chaudhuri, P. and Marron, J. S. (2000) Scale space view of curve estimation. *Ann. Statist.* **28**, 408-428.

[6] Chaudhuri, P., Ghosh, A. K. and Oja, H. (2009) Classification using hybridization of parameteric and nonparametric classifiers. *IEEE Trans. Pattern Anal. Machine Intell.*, **31**, 1153-1164.

[7] Cover, T. M. and Hart, P. E. (1967) Nearest neighbor pattern classification, *IEEE Trans. Info. Theory*, **13**, 21-27.

[8] Dzeroski, S. and Zenko, B. (2004) Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, **54**, 255-273.

[9] Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Ann. Eugen.*, **7**, 179-188.

[10] Friedman, J. H., Hastie, T. and Tibshirani, R. (2000) Additive logistic regression : a statistical view of boosting (with discussion). *Ann. Statist.*, **28**, 337-374.

[11] Ghosh, A. K. and Chaudhuri, P. (2005) On data depth and distribution free discriminant analysis using separating surfaces. *Bernoulli*, **11**, 1-27.

[12] Ghosh, A. K., Chaudhuri, P. and Murthy, C. A. (2005) On visualization and aggregation of nearest neighbor classifiers. *IEEE Trans. Pattern Anal. Machine Intell.*, **27**, 1592-1602.

[13] Ghosh, A. K., Chaudhuri, P. and Sengupta, D. (2006) Classification using kernel density estimates : multi-scale analysis and visualization. *Technometrics*, **48**, 120-132.

[14] Gilks, W. R., Richardson, S. and Spiegelhalter, D. J. (1996) *Markov Chain Monte Carlo in Practice*, Chapman and Hall, London.

[15] Glad, I. (1998) Parametrically guided nonparametric regression. *Scand. J. Statist.*, **25**, 649-668.

[16] Godtliebsen, F., Marron, J. S. and Chaudhuri, P. (2002) Significance in scale space for bivariate density estimation. *J. Comput. Graph. Statist.*, **11**, 1-22.

[17] Hastie, T. and Tibshirani, R. (1996) Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Anal. Machine Intell.*, **18**, 607-616.

[18] Hastie, T., Tibshirani, R. and Friedman, J. H. (2009) *The Elements of Statistical Learning : Data Mining, Inference and Prediction*. Springer Verlag, New York.

[19] Hjort, N. L. and Glad, I. (1995) Nonparametric density estimation with a parametric start. *Ann. Statist.*, **23**, 882-904.

[20] Hjort, N. L. and Jones, M. C. (1996) Locally parametric nonparametric density estimation. *Ann. Statist.*, **24**, 1619-1647.

[21] Holmes, C. C. and Adams, N. M. (2002) A probabilistic nearest neighbor method for statistical pattern recognition. *J. Royal Statist. Soc., Series B*, **64**, 295-306.

[22] Hoti, F. and Holmstrom, L. (2004) A semiparametric density estimation approach to pattern classification. *Pattern Recognition*, **37**, 409-419.

[23] Jacobs, R. A., Jordon, M. I., Nowlan, S. J. and Hinton, G. E. (1991) Adaptive mixtures of local experts. *Neural Computation*, **3**, 79-87.

[24] Johnson, R. A. and Wichern, D. W. (1992) *Applied Multivariate Statistical Analysis*. Prentice Hall, New Jersey.

[25] Jones, M. C., Linton, O. and Nielsen, J. P. (1995) A simple and effective bias reduction method for density and regression estimation. *Biometrika*, **82**, 327-338.

[26] Kayank, C. and Alpaydin, E. (2000) Multistage cascading of multiple classifiers : one man's noise is other man's data. *Proc. 17th Inter. Conf. Mach. Learn.*, Stanford, U.S.A.

[27] Kittler, J., Hatef, M., Duin, R. P. W. and Matas, J. (1998) On combining classifiers. *IEEE Trans. Pattern Anal. Machine Intell.*, **20**, 226-239.

[28] Lawson, C. L. and Hanson, R. J. (1995) *Solving Least Squares Problems*, SIAM Publications.

[29] Loftsgaarden, D. O. and Quesenberry, C. P. (1965) A nonparametric estimate of a multivariate density function. *Ann. Math. Statist.*, **36**, 1049-1051.

[30] MacLachlan,G. and Krishnan, T. (1997) *The EM Algorithm and Extensions*. Wiley, New York.

[31] Olkin, I. and Spiegelman, C. H. (1987) A Semiparametric approach to density estimation. *J. Amer. Statist. Assoc.*, **82**, 858-865.

[32] Peterson, G. E. and Barney, H. L. (1952) Control methods used in a study of vowels. *J. Acoust. Soc. Amer.*, **24**, 175-185.

[33] Schapire, R. E., Fruend, Y., Bartlett, P. and Lee, W. (1998) Boosting the margin : a new explanation for the effectiveness of voting methods. *Ann. Statist.*, **26**, 1651-1686.

[34] Silverman, B. W. (1986) *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London.

[35] Ting, K. M. and Witten, I. H. (1999) Issues in stacked generalization. *J. Art. Intell. Res.*, **10**, 271-289.

[36] Wolpert, D. H. (1992) Stacked generalizations. *Neural Networks*, **5**, 241-259.

[37] Zhou, Z. H. and Yu, Y. (2005) Ensembling local learners through multimodal perturbations. *IEEE Trans. Systems, Man, Cybern. B*, **35**, 725-735.

[38] Zhu, J., Rosset, R., Zhou, H. and Hastie, T. (2005) Multi-class adaboost. *Tech. Report, Dept. of Stat., Univ. of Michigan.*