Classification Based on Combination of Kernel Density Estimators

Mateusz Kobos and Jacek Mańdziuk

Warsaw University of Technology,
Faculty of Mathematics and Information Science,
Plac Politechniki 1, 00-661 Warsaw, Poland
{M.Kobos, J.Mandziuk}@mini.pw.edu.pl

Abstract. A new classification algorithm based on combination of kernel density estimators is introduced. The method combines the estimators with different bandwidths what can be interpreted as looking at the data with different "resolutions" which, in turn, potentially gives the algorithm an insight into the structure of the data. The bandwidths are adjusted automatically to decrease the classification error. Results of the experiments using benchmark data sets show promising performance of the proposed approach when compared to classical algorithms.

Key words: kernel density estimation, classification, density estimators combination

1 Introduction

Classification based on density estimators is one of the basic methods used in machine learning (see e.g. [1] for an introduction to the subject). Among the non-parametric density estimation methods, the most popular are the Gaussian Mixture Model (GMM) and the Kernel Density Estimator (KDE), the latter also called the Parzen windows method. In KDE, in order to get the density estimate in a given point, a distance-based influence of all points from the training set on that point is calculated. A "kernel function" is used to put a relatively greater emphasis on points that are closer than on those which are placed further. Typically, kernel's definition includes a parameter called "bandwidth" which determines how much emphasis is put on the closest points.

A method to estimate the density (but not to make a classification) using a linear combination of predefined GMMs and KDEs was introduced in [2]. The parameters of the combination are computed using the stacking meta-learning method with the EM algorithm. In [3], another fusion of GMM and KDE is proposed. The GMM algorithm is used to assign a weight to each of the predefined KDE models. Yet another meta-learning approach – boosting – is proposed in [4]. The base boosted classifiers are simple algorithms based on KDE; each of the training points gets a different weight in each algorithm's iteration. A different meta-learning approach is proposed in [5]. Authors use ensemble averaging method based on GMMs to estimate the density. Boosting and bagging

meta-learning algorithms are also used in [6] for the density estimation. The EM algorithm is used to maximize training data likelihood but the classification error is not directly optimized. In [7], authors describe an algorithm which uses "Gaussian product kernel estimators" where the bandwidths are chosen independently for each class-dimension combination. What is more, the bandwidths can vary depending on the localization in the feature space.

In this paper, we propose a new classification method which is based on a combination of KDEs. The algorithm is significantly different from each of the above mentioned methods, since it optimizes directly the classification error and does not use explicitly any meta-learning algorithm. The combined kernels bandwidth is not predefined but adjusted to the data.

The method exhibits some similarities to the Ghosh et al. approach [8], where the authors introduced a method which is at heart a binary classifier. They search for an optimal bandwidth using the cross-validation method for each of the 2 classes independently. As a result, they obtain a pair of bandwidth values which can be interpreted as a point in a 2-dimensional bandwidth space. In order to classify a given test point, a couple of density estimations are made. Each of them corresponds to a pair from the neighborhood of the optimal bandwidth values pair in the bandwidth space. The estimation results are transformed in a certain way and their weighted sum is computed. The sum yields the final classes probabilities. The main difference between the method presented in [8] and our approach is that the latter one is simpler (and possibly faster) because the parameter space that is searched is one-dimensional instead of two-dimensional.

The paper is organized as follows: Sect. 2 contains a description of the proposed algorithm, Sect. 3 presents results of the tests on benchmark data sets and comparison with the literature results, Sect. 4 concludes the paper.

2 Algorithm Description

Every classification machine learning algorithm has two modes of operation: training phase and classification/recall phase. In Sections 2.1, 2.2, 2.3 we describe the classification phase and Sect. 2.4 contains a description of the training phase.

2.1 Introduction

The problem of classification is to create a decision rule $d(\mathbf{x}) : \mathbb{R}^d \to \{\omega_1, \omega_2, \ldots, \omega_c\}$ to classify a d-dimensional observation (point) $\mathbf{x}$ into one of c classes ω_i. The rule is usually built using the observations from a training set, its robustness is tested on the observations from a testing set. One possibility to create such a rule is to employ the Bayesian classifier of the form

$$d_B(\mathbf{x}) = \arg\max_{w_i} \hat{P}(\omega_i|\mathbf{x}) = \arg\max_{w_i} \frac{\hat{p}(\mathbf{x}|\omega_i)\hat{P}(\omega_i)}{\hat{p}(\mathbf{x})} \quad , \tag{1}$$

where $\hat{P}(\omega_i|\mathbf{x})$ is a posterior probability estimator of class ω_i, $\hat{P}(\omega_i)$ is a prior estimator of class ω_i (in practice, it is equal to the fraction of observations from

a given class in the training set), $\hat{p}(\mathbf{x}|\omega_i)$ is an estimated probability density function of class ω_i, and $\hat{p}(\mathbf{x}) = \sum_{i=1}^{c} \hat{p}(\mathbf{x}|\omega_i)\hat{P}(\omega_i)$ is a normalization factor. All of the quantities in this formula are simple to compute except for the class probability density estimate $\hat{p}(\mathbf{x}|\omega_i)$.

One of the most popular density estimators is the Kernel Density Estimator. When applied in the Bayesian classifier, it has the form of

$$\hat{p}_h(\mathbf{x}|\omega_i) = \frac{1}{|\mathcal{D}_i|} \sum_{\mathbf{x}' \in \mathcal{D}_i} \frac{1}{h^d} \phi\Big(\frac{\mathbf{x} - \mathbf{x}'}{h}\Big), \tag{2}$$

where $\mathcal{D}_i$ is the set of observations belonging to class ω_i, the function $\phi(\mathbf{x}) : \mathbb{R}^d \to [0, \infty)$ is a density function called "kernel function" and h is a smoothing factor called "bandwidth".

One of the most popular choices for the kernel function is the Gaussian kernel. In the general case, it has the form of

$$\phi(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\mathbf{\Sigma}|^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x} - \mu)\Big] \ , \tag{3}$$

where the covariance matrix $\mathbf{\Sigma}$ is responsible for the hyper-ellipsoidal shape of the kernel (cf. [1, Sect. 2.5.2]). Generally, the shape of the kernel should be adjusted to match the layout of the training points in the feature space. There are two approaches to reach this goal. The first one is to set an appropriate shape of the kernel i.e. to adjust the covariance matrix to match the data. The second one is to let the covariance matrix be fixed and equal to the identity matrix $\mathbf{\Sigma} = \mathbf{I}$ (the shape of the kernel will be circular in this case) and to transform the feature space instead (this method was used e.g. in [7]). We have chosen the second approach because it makes the algorithm simpler to analyze and is less computationally intensive. During the experiments, two different transformations were used: standardization and whitening, the latter implemented with Principal Component Analysis (PCA).

2.2 Combination of Estimators

The kernel's bandwidth parameter specifies how smooth the resulting estimation of density function will be. The larger the factor, the smoother and less concentrated on the training points the estimation of the density function. We can also interpret the bandwidth as a "resolution" of the data view – the larger the bandwidth, the smaller resolution and the more general view on the data (i.e. the density assumes similar values even in distant points of the space, which makes them difficult to distinguish).

The main idea presented in this paper is to combine a certain number of KDEs with different bandwidths which are selected to match the analyzed data set. Such an approach of looking at the data with different "resolutions" should give a better insight into the data structure and result in a better classification than a method using a single "resolution". In this "multi-resolution" case,

the formula for posterior probability estimator will be an average of different-bandwidth KDEs: $\hat{p}(\mathbf{x}|\omega_i) = \frac{1}{E}\sum_{j=1}^{E}\hat{p}_{h_j}(\mathbf{x}|\omega_i)$, where E is the number of KDEs, and h_j is individual bandwidth of j-th KDE.

The other underlying idea is to make the estimators' bandwidths related in some way. It is proposed that bandwidths of different estimators decrease in an exponential manner. This way it is possible to combine vastly different data view "resolutions". As a result, bandwidth is a function of the following form:

$$h_j(a) = h_{\min} + a^j(h_{\max} - h_{\min}) \ , \tag{4}$$

where $j \in \{1, \ldots, E\}$ is an estimator number, $a \in [0, 1]$ is a parameter determining how fast the bandwidths decrease, $[h_{\min}, h_{\max}]$ is a range of bandwidth values.

2.3 Bandwidth Range

The first question to be answered is what is a "sensible" bandwidth range in (4). If one uses a very small bandwidth, numerical problems occur. If a given testing point is far from any other point in the training set, then every class-conditional density in the given point will be close to zero and, as a result, $\hat{p}(\mathbf{x})$ in the denominator of the Bayes classifier formula (1) will be close to zero. For a sufficiently small bandwidth, the value will be smaller than the computer's machine precision and, as a consequence, assumed to be equal to zero. This, in turn, will make the formula impossible to evaluate. The next problem is that for small bandwidths one can get unreliable and possibly misleading information for classification [8, Sect. 2.2]. To deal with these problems, we decided to take an approach similar, but not the same, to the one presented in [8]. We chose a small ratio $\frac{1}{\xi}$ of the smallest non-zero percentile of the pairwise distances of the transformed data points from the training set as the lower limit. The ξ is a radius of a sphere which contains 99% of kernel's probability mass – it is assumed that outside this sphere, the kernel's influence on the overall density is negligible. In the presented algorithm, the Gaussian kernel is used, and for such a kernel it can be shown that $\xi = \sqrt{F^{-1}_{\chi^2(d)}(0.99)}$, i.e. it is a square root of inverted χ^2 distribution function with d degrees of freedom in point 0.99.

The proposed solution is not always sufficient to solve the above mentioned problems. In extreme situations, an outlier point can be situated far away from all the training points, and in such case the algorithm would not be able to evaluate the formula (1) even for larger bandwidths. We can note that in such situations (i.e. outlier point or, equivalently, a very small bandwidth) the KDE classification result mimics the result of Nearest Neighbor algorithm [9, p.251]. Thus, if the denominator of the formula (1) is equal to zero (i.e. the value is smaller than the computer's machine precision), the probabilities that would be yielded by the Nearest Neighbor algorithm are returned as the classification result.

The upper limit of the bandwidth range, on the other hand, is set to be the 99-th percentile of pairwise distances of the transformed data points in the

training set. As can be seen, when calculating the lower and the upper limits, small and large percentiles are used instead of simply using the minimum and the maximum. The reason is making the calculations resistant to outlier points, which could unnecessarily widen the bandwidth range. Apart from that, the estimation of both lower and upper limits is rather conservative.

2.4 Algorithm Training

In this section the training phase in which the parameters of KDEs combination (especially a) are computed is described. Parameter a in (4) is selected to minimize the classification error on the training set. The classification error that is minimized is the Mean Squared Error (MSE) defined as

$$\mathrm{MSE}(\hat{P}(\cdot), \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \sum_{i=1}^{c} (\hat{P}(\omega_i|\mathbf{x}) - \mathbf{t}_i(\mathbf{x}))^2 \ , \tag{5}$$

where $\mathcal{D}$ is the data set on which the error is computed, $\hat{P}(\omega_i|\mathbf{x})$ is the algorithm's posterior probability estimation for class ω_i, and $\mathbf{t}_i(\mathbf{x})$ is a vector whose i-th component, where i corresponds to the actual class ω_i, is 1 and all other components are 0.

The training phase of the algorithm consists of several steps. 1) As the first step, the sequence of the training instances is randomly permuted. The randomization is required for the cross-validation folds (which are created from the data in later steps) to be independent as much as possible (e.g. we do not want to cumulate all of the samples from one class in one fold) which is a standard requirement in the cross-validation method. It is worth noting that, apart from this step, the algorithm is completely deterministic. 2) In the next step, the data is transformed. The transformation parameters (e.g. for standardization transformation: sample expected values and sample standard deviations) are saved – they will be used later to build the classification error function. 3) Next, using the transformed data, a "sensible" bandwidth range (as it was described in Sect. 2.3) is calculated. The range will be searched for the optimal bandwidth value. 4) Finally, in the last step, the value of a that minimizes the 10-fold stratified cross-validation estimator of a classification error function is searched for. The minimization is performed in a exhaustive way with a grid-search method. In this method, we compute the value of the cross-validation estimator function in 100 equidistant points from range $[0, 1]$ (see Fig. 1 for an example of examined error values and location of the optimal a). As a result of the training phase, the optimal a is obtained along with the transformation parameters, transformed data and bandwidth range. These values will be used later in the classification phase.

The construction of 10-fold stratified cross-validation estimator of the classification error function needs some further explanation. First, 10 splits of the training data are created. Each split consists of two disjoint data sets: the fitting set $\mathcal{D}$ which is used to train the classifier, and the validation set $\mathcal{D}^v$ which is

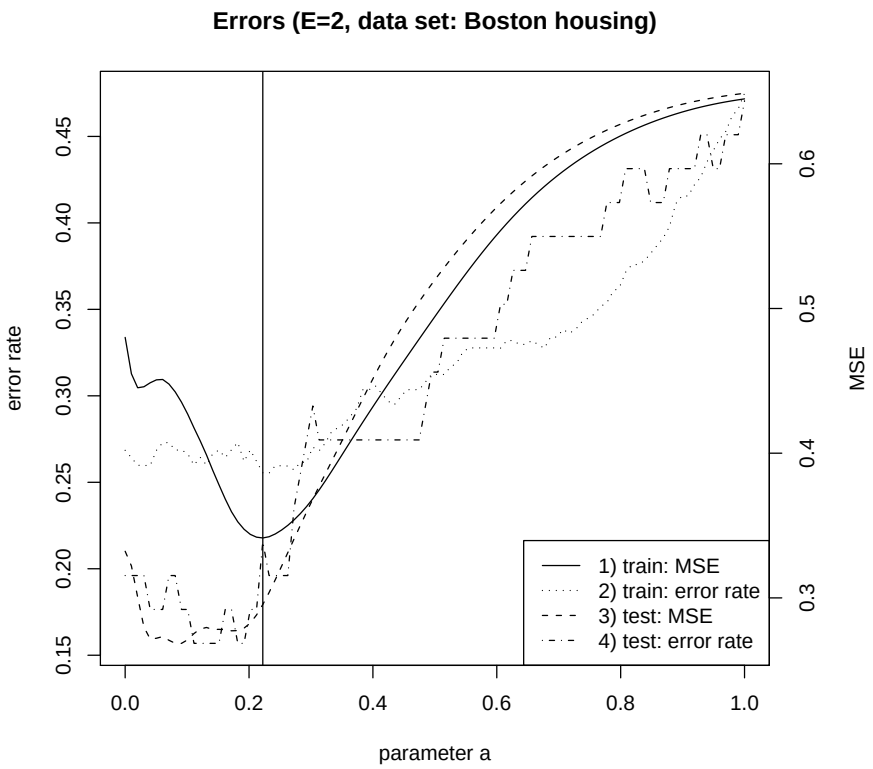

Fig. 1. An example of classification errors for different parameter a values. The algorithm with 2 KDEs and standardization transformation is tested on the *Boston housing* data set. Examined error types: cross-validation MSE on the training set (*line 1*), cross-validation error rate on the training set (*line 2*), MSE on the test set (*line 3*), error rate on the test set (*line 4*). The optimal a value found by the algorithm is equal to the global minimum of the cross-validation MSE error on the training set (*vertical line*).

used to compute the classification error of the trained algorithm. For each split, the classification MSE (5) defined as $\mathrm{Error}(a) = \mathrm{MSE}(\hat{P}(\cdot; a), \mathcal{D}^v)$ is calculated, where a is the bandwidths' decrease parameter from (4), $\hat{P}(\omega_i|\mathbf{x}; a)$ is the posterior probability estimator from (1) dependent on a. The MSE function is used instead of direct use of the error rate function (i.e. misclassification ratio) because it seems to be less affected by the random dependencies in the data (see Fig. 1). The function uses the training set transformation computed in step 2 of the training phase to transform the data in each of the cross-validation splits. The reason the transformation is not computed for every split's fitting set independently, as would a standard cross-validation procedure suggest, is that we

want the bandwidths calculated for every split to correspond to the same values in the original non-transformed space. If the transformations were calculated independently, the same bandwidth value would correspond to different bandwidth values in each split in the original space. The priors used in each split are also estimated on the whole training set (for similar reasons).

3 Experiments

The efficacy of the proposed algorithm was compared with the results published in [10] and [8]. In [10], the authors consider 33 classification algorithms and verify them using different data sets, which establishes a broad comparison base for the proposed algorithm. Among the data sets examined in [10], the ones that matched the proposed algorithm (i.e. sets with numerical attributes only) were chosen. The raw error rates used for comparison were retrieved from the article's appendix available at one of the author's website. In [8], on the other hand, the authors compare the algorithm they introduced with literature results. The same data is used in this paper to compare our method with that of [8].

The following data sets were used in the experiments: *Boston housing* (Boston housing, used in [10]), *breast cancer* (Wisconsin breast cancer data set, collected at the University of Wisconsin by W.H. Wolberg [11], used in [10]), *glass* (forensic glass data, used in [8]), *Indian diabetes* (PIMA Indian diabetes, used in [10]), *liver disorders* (BUPA liver disorders, used in [10]), *Ripley's synthetic* (Ripley's synthetic data, used in [8]), *satellite image* (StatLog satellite image, used in [10]), *sonar* (sonar data, used in [8]), *vehicle silhouette* (StatLog vehicle silhouette, used in [10]). All of the data sets except for *Ripley's synthetic* were downloaded from the UCI Machine Learning Repository [12]; the *Ripley's synthetic* data set was downloaded from [13]. In the cited articles, some of the original data sets were preprocessed and we executed the same preprocessing steps. When testing our algorithm, we followed the methodology used in adequate articles with an exception for the holdout experiments (for data sets with a selected testing set). The holdout experiments were executed 10 times instead of once, because our algorithm is non-deterministic and repeating the experiment several times results in a more unbiased quality estimation.

3.1 Results

During the experiment, the algorithm was tested with data standardization transformation and number of estimators equal to: 1 (*E=1*), 2 (*E=2*), 5 (*E=5*), or the number of classes in the data set (*E=cl.no.*). For comparison, the algorithm was also tested with data whitening transformation and the number of estimators equal to: 1 (*PCA E=1*), or the number of classes in the data set (*PCA E=cl.no.*).

Four of the algorithms yielded results better than the literature ones on at least one data set (see Table 1), and one of them (*E=2*) yielded results better than the literature ones on 2 data sets. On average, the results yielded by *E=2*,

E=5, *E=cl.no.* were better than the results of the simplest version *E=1* (average differences equal to .0035, .0018, .002, resp.); the results yielded by *PCA E=1*, *PCA E=cl.no.* were worse (average differences equal to -.01, -.012, resp.).

Table 1. Comparison of experimental results. Error rate (misclassification error) for different algorithm versions and different data sets was measured. The results that are better than the best literature result (*best lit.*) are marked with an asterisk. All of the results are given with the same number of decimal places as in the respective literature source.

data set	best lit.	E=1	E=2	E=5	E=cl.no.	PCA E=1	PCA E=cl.no.
Boston housing	.221	.239	.247	.249	.245	.243	.247
breast cancer	.0278	.0323	.0323	.0323	.0323	.0661	.0661
glass	.236	.252	.233*	.247	.247	.271	.308
Indian diabetes	.221	.251	.256	.259	.256	.264	.269
liver disorders	.279	.405	.365	.365	.365	.321	.310
Ripley's synthetic	.090	.105	.094	.097	.094	.105	.094
satellite image	.098	.097*	.095*	.095*	.096*	.292	.288
sonar	.135	.194	.222	.216	.222	.181	.181
vehicle silhouette	.145	.286	.285	.284	.286	.208	.205

When comparing the classification results, it might be helpful to check at which quantile is a given result situated among the literature results (Fig. 2) (the quantiles were calculated using `ecdf` function in the R environment [14]). This method potentially allows to assess if the improvement of the result is meaningful (e.g. the same error rate improvement for a difficult data set can be more important than for a simple one). As can be seen in Fig. 2 the results of all of the algorithm's versions were situated among the top-50% literature results in 4 out of 9 examined data sets (except for *PCA E=1* where the ratio was 5 out of 9). On average, the quantile results yielded by *E=2*, *E=cl.no.* were better than the results of the simplest version *E=1* (average differences equal to .0168, .0101, resp.); the results yielded by *E=5*, *PCA E=1*, *PCA E=cl.no.* were worse (average differences equal to -.0034, -.11, -.0797, resp.).

In summary, the results yielded by all of the algorithm's versions are promising when compared to the literature results. Although the error rate on some of the data sets was high, it can be argued that according to the no free lunch theorem [1, Sect. 9.2.1] no single classifier can achieve great results on all of the problems. Furthermore, the algorithms that used the whitening transformation generally yielded worse results than the others. On the other hand, the whitening transformation improved the results on some of the data sets (see e.g. results on data set *liver disorders* in Fig. 2), so we can conclude that this transformation can improve or worsen the results depending on the data. What is more, although various algorithm's versions excelled in the classification of various data sets, the version which seemed to be generally the best is the one which uses two KDEs with standardization transformation (*E=2*). This version yielded the

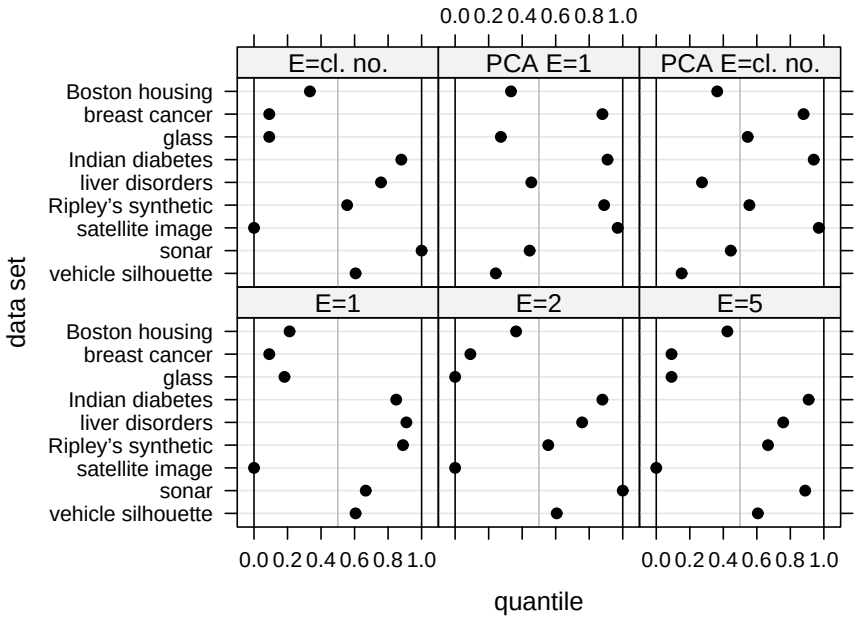

Fig. 2. Comparison of the experimental results with the literature results. Each panel contains relative error rates of a certain algorithm version on different data sets. Each point corresponds to a quantile position of the experiment result among the literature results. For example, in the panel *E=1*, 4 results are among 50% best literature results and 5 results are among 50% worst literature results. Points at quantile 0 correspond to the results that are as good as the best literature result or better; points at quantile 1 correspond to the results that are as bad as the worst literature result or worse.

results that were better than the best literature results on two data sets: *glass* and *satellite image*. The results were also better, on average, than the results of the simplest version which uses one KDE (*E=1*); this observation justifies application of 2-element KDEs combination instead of a simpler version with a single KDE.

4 Conclusions and Future Work

A new classification algorithm based on a combination of Kernel Density Estimators, where the classification error is minimized directly is presented in the paper. The algorithm performs well on the benchmark data sets when compared to the literature results; one of the algorithm's versions yields the results which are better on two data sets than the best ones reported in the literature. These results confirm the algorithm's potential, especially in the domains related to

the examined data sets. It will be the object of further research to examine the detailed characteristics of these data sets, and, as a result, to determine which algorithm version matches best each data set domain.

The next steps of the algorithm's development involve applying a numerical optimization method (e.g. one of the pseudo-Newton algorithms) instead of the exhaustive optimization currently employed, which should result in significantly shorter training times. Other extension paths concern testing other KDEs combination functions, applying other kernel types (e.g. p-Gaussian), and removing non-typical observations (similarly to [15]) from the training set which may lead to better classification results.

Acknowledgments. The authors would like to thank prof. Jan Mielniczuk for valuable suggestions concerning the algorithm's design and development.

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. 2nd edn. Wiley-Interscience Publication (2000)
2. Smyth, P., Wolpert, D.: Linearly combining density estimators via stacking. Machine Learning 36, 59–83 (1999)
3. Marchette, D.J., Priebe, C.E., Rogers, G.W., Solka, J.L.: Filtered kernel density estimation. Computational Statistics 11, 95–112 (1996)
4. Di Marzio, M., Taylor, C.C.: On boosting kernel density methods for multivariate data: density estimation and classification. Statistical Methods and Applications 14, 163–178 (2005)
5. Ormoneit, D., Tresp, V.: Averaging, maximum penalized likelihood and bayesian estimation for improving gaussian mixture probability density estimates. In: IEEE Transactions on Neural Networks, vol. 9, pp. 639–650 (1998)
6. Ridgeway, G.: Looking for lumps: boosting and bagging for density estimation. Computational Statsistics and Data Analysis 38, 379–392 (2002)
7. Cooley, C.A., MacEachern, S.N.: Classification via kernel product estimators. Biometrika 85, 823–833 (1998)
8. Ghosh, A.K., Chaudhuri, P., Sengupta, D.: Classification using kernel density estimates: Multiscale analysis and visualization. Technometrics 48, 120–132 (2006)
9. Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley, New York (1992)
10. Lim, T.S., Loh, W.Y., Shih, Y.S.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning 40, 203–228 (2000)
11. Mangasarian, O., Wolberg, W.: Cancer diagnosis via linear programming. Siam News 23, 1–18 (1990)
12. Asuncion, A., Newman, D.: UCI Machine Learning Repository (2007)
13. Ripley, B.: Pattern recognition and neural networks datasets collection, `http://www.stats.ox.ac.uk/pub/PRNN/` (1996)
14. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2008)
15. Dendek, C., Mańdziuk, J.: Improving performance of a binary classifier by training set selection. In: Kurková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part 1. LNCS, vol. 5163, pp. 128–135. Springer, Heidelberg (2008)