Confidence of Classification in Classifier Combining

Štefka, David
2007

Confidence of Classification in Classifier Combining

Post-Graduate Student:
ING. DAVID ŠTEFKA
Institute of Computer Science of the ASCR, v. v. i.
Pod Vodárenskou věží 2
182 07 Prague 8, CZ
stefka@cs.cas.cz

Supervisor:
ING. RNDR. MARTIN HOLEŇA, CSC.
Institute of Computer Science of the ASCR, v. v. i.
Pod Vodárenskou věží 2
182 07 Prague 8, CZ
martin@cs.cas.cz

Field of Study:
Mathematical Engineering

The research reported in this paper was partially supported by the Program "Information Society" under project 1ET100300517.

Abstract

Classifier combining is a succesful method for improving the quality of classification. In this paper, we introduce the concept of confidence of classification and define two confidence measures – the local accuracy and the local class separability. We propose a simple classifier aggregation algorithm which uses the concept of confidence, the Filtered Mean Value Aggregation algorithm. This algorithm outperforms two commonly used methods for classifier combining on two datasets. We show that by incorporating confidence into classifier aggregation algorithms, we can improve state-of-the-art methods for classifier combining.

1. Introduction

The literature shows that a team of multiple classifiers can perform the classification task better than any of the individual classifiers. However, to achieve this, the classifier outputs have to be combined wisely. For this purpose, many methods have been introduced in the literature. These can be grouped into classifier selection and classifier aggregation.

In classifier selection, some rule is used to determine which classifier to use for the current pattern; only this "expert" classifier is then used for the final prediction, and the rest of the team is discarded. In classifier aggregation, outputs of all the classifiers are aggregated into the final decision.

Common drawback of classifier aggregation methods is that they are *global*, i.e., they do not adapt themselves to the particular patterns to classify. In other words, the combination is specified during a training phase, prior to classifying a test pattern. A typical example is that if we use the weighted mean aggregation rule, the weights of the individual classifiers are usually based on the classifiers' accuracies. While it is true that if a classifier has high accuracy, its weight should be higher, still, for the *curent pattern*, some other classifier could be more suitable.

While classifier selection methods use some techniques to determine which classifier is *locally* better than the others, such algorithms select only one classifier, discarding much potentialy useful information, thus reducing the robustness compared to classifier aggregation.

In this paper, we try to identify the strong points of classifier selection techniques and incorporate them info classifier aggregation methods. This will enable us to create novel methods for classifier aggregation which can provide better results than state-of-the-art methods for classifier combining on two datasets.

We introduce the concept of *confidence* of classification, which can be used both as a criterium for classifier selection and for improving classifier aggregation. We define two confidence measures, and propose an algorithm for classifier aggregation which utilizes the concept of confidence. We then show that this algorithm outperforms two commonly used methods for classifier combining.

The paper is structurred as follows: Section 2 describes the basics of classification and classifier combining, and summarizes methods for classifier selection and classifier aggregation. Section 3 then introduces the concept of confidence of classification. Section 4 presents the experimental results. Finally, Section 5 then concludes the paper.

2. Classifier Combining

Throughout the rest of the paper, we use the following notation. Let $\mathcal{X} \subseteq \mathbf{R}^n$ be a n-dimensional *feature space*, an element $\vec{x} \in \mathcal{X}$ of this space is called *pattern*, and let $C_1, \ldots, C_N \subseteq \mathcal{X}$ be disjoint sets called *classes*. The goal of classification is do determine to which class a given pattern belongs. We call a *classifier* any mapping ϕ from the following:

- *possibilistic classifier* – $\phi : \mathcal{X} \rightarrow [0,1]^N$, where $\phi(\vec{x}) = (\mu_1, \ldots, \mu_N)$ are *degrees of classification* to each class.
- *normalized possibilistic classifier* – $\phi : \mathcal{X} \rightarrow [0,1]^N$, where $\phi(\vec{x}) = (\mu_1, \ldots, \mu_N)$, $\sum_i \mu_i = 1$.
- *crisp classifier* – $\phi : \mathcal{X} \rightarrow \{1, \ldots, N\}$, where $\phi(\vec{x})$ is the predicted class label of pattern $\vec{x}$. Crisp classifier can also be defined as a special case of a normalized possibilistic classifier, such that one degree of membership is equal to 1 and the others are equal to 0.

Normalized possibilistic classifiers are sometimes called *probabilistic* [1]. However, they do not need to be based on probablility theory, so we will call them normalized possibilistic. Other types of classifiers, such as *rank classifier* [2], can be defined, but we deal with crisp and possibilistic classifiers only in the rest of the paper. The conversion of a possibilistic classifier ϕ_p to a crisp classifier ϕ_c is called *hardening*:

$$\phi_c(\vec{x}) = \arg\max_{i=1,\ldots,N}\{\mu_i\}, \qquad (1)$$

where $\phi_p(\vec{x}) = (\mu_1, \ldots, \mu_N)$.

In classifier combining, we create a team of classifiers, let each of the classifiers predict independently, and then combine the classifiers' outputs into one final classifier. This combined classifier can perform its classification task better than any of the individual classifiers in the team. Methods which use more or less this idea can be found under many names in the literature – *classifier combining*, *classifier aggregation*, *classifier fusion*, *classifier selection*, *mixture of experts*, *classifier ensembles*, etc. Basically, there are two main approaches to classifier combination:

- *classifier selection*, where we use some rule to determine which classifier to use for the current pattern; only this "expert" classifier is then used for the final prediction
- *classifier aggregation*, where all the classifiers in the team are used for the final decision

Classifier combining consists of two steps – first, we create a team of classifiers, and then we adopt some strategy to combine the classifiers' outputs into the final decision. The former step is common for both classifier selection and aggregation (algorithms for creating a team of classifiers are descibed in Sec. 2.1), while for the latter, different algorithms are needed (these are described in Sec. 2.2 and 2.3).

2.1. Ensemble Methods

If the team of classifiers consists only of classifiers of the same type, which differ only in their parameters, dimensionality, or training sets, the team is usually called an *ensemble* of classifiers. That is why the methods which create a team of classifiers are sometimes called *ensemble methods*. The restriction to classifiers of the same type is not essential, but it ensures that the outputs of the classifiers are consistent.

Well-known methods for ensemble creation are *bagging* [3], *boosting* [4], error correction codes [5], or *multiple feature subset* (MFS) methods [6]. These methods try to create an ensemble of classifiers which are both *accurate* and *diverse* (in the sense that they predict differently).

Diversity of the ensemble is thought to be a cruical issue for classifier combining; however, there is no generally accepted measure of diversity. In [7], 10 diversity measures are studied, resulting in the suggestion to use the Q statistics because of its simplicity.

2.2. Classifier Selection

Crisp classifiers are not much appropriate for classifier combining, because they do not provide information about degree of classification to each class. For these classifiers, only simple techniques like voting or single best selection can be used. That's the reason why we restrict to possibilistic classifiers in this paper. In the rest of the paper, we suppose that we have constructed an ensemble $(\phi_1, \ldots, \phi_r)$ of r possibilistic classifiers using some of the methods described in Sec. 2.1.

Classifier selection algorithms [8, 9, 10] use some criterion to determine which classifier is most suitable for the current pattern, and the output of this classifier is taken as the final result. The criterion for selection can be some global property of the ensemble, as in *single best selection* (SBS), or some local property, as in *dynamic best selection* (DBS).

In SBS, the criterion for selection is usually the validation error rate of the individual classifiers. The classifier with the lowest validation error rate is used for prediction of all the patterns (i.e. the other classifiers are entirely discarded). In DBS, the classifier optimizing some local criterion (for example local accuracy of the classifier in neighborhood of the current pattern) is selected for the prediction.

2.3. Classifier Aggregation

For classifier aggregation, the output of the ensemble $(\phi_1, \ldots, \phi_r)$ for input pattern $\vec{x}$ can be structured to a $r \times N$ matrix, called *decision profile* (DP):

$$DP(\vec{x}) = \begin{pmatrix} \phi_1(\vec{x}) \\ \phi_2(\vec{x}) \\ \vdots \\ \phi_r(\vec{x}) \end{pmatrix} = \begin{pmatrix} \mu_{1,1} & \mu_{1,2} & \cdots & \mu_{1,N} \\ \mu_{2,1} & \mu_{2,2} & \cdots & \mu_{2,N} \\ & & \ddots & \\ \mu_{r,1} & \mu_{r,2} & \cdots & \mu_{r,N} \end{pmatrix} \tag{2}$$

The $i-$th row of $DP(\vec{x})$ is an output of the corresponding classifier ϕ_i, and the $j-$th column contains the degrees of classification of $\vec{x}$ to the corresponding class C_j given by all the classifiers.

Many methods for aggregating the ensemble of classifiers into one final classifier have been reported in the literature. A good overview of the commonly used aggregation methods can be found in [1]. These methods comprise simple arithmetic rules (sum, product, maximum, minimum, average, weighted average, see [1, 11]), fuzzy integral [1, 12], Dempster-Shafer fusion [1, 13], second-level classifiers [1], decision templates [1], and many others.

In this paper, we introduce the concept of *confidence* of classification, which can be used both as a criterion for classifier selection, and for improving classifier aggregation by filtering the worst classifiers in the team. The concept of confidence is described in the next section.

3. Confidence Classifiers

The classifiers defined in Sec. 2 (both crisp and possibilistic) give us information about the *evidence* of classification (i.e., degrees of classification) of the current pattern $\vec{x}$. This is all we need to know if we are classifying patterns using a single classifier. However, in classifier combining, we have a team of classifiers, and the information about "how can we trust the output of classifier ϕ_i" could be very useful. For this purpose, we introduce a concept of *confidence* of classification.

The concept of confidence is not new to classifier combining – in classifier selection, the criteria for selection can be viewed as some confidence measures. In weighted mean classifier aggregation, the individual classifiers' error rates (which can again be viewed as some confidence measure) are used to adapt the weights of the individual classifiers etc. In this paper, we try to generalize different methods which use this approach, and incorporate all of them into the concept of confidence. This enables us to create general algorithms for classifier aggregation, which use some properties of classifier selection, improving both classifier aggregation and classifier selection. This is what makes the approach novel.

Suppose we have a classifier ϕ, and a pattern $\vec{x}$ to classify. The confidence of classification of the pattern $\vec{x}$ using classifier ϕ is a real number in the unit interval $[0, 1]$, and we model it by a mapping $\kappa_\phi : \mathcal{X} \to [0, 1]$. The mapping κ_ϕ will be called *confidence measure*, and the tuple (ϕ, κ_ϕ) will be called *confidence classifier*.

The confidence of classification $\kappa_\phi(\vec{x})$ can be any property estimating the degree to which we can trust the output of ϕ for current pattern $\vec{x}$. In this paper, we will use the following two confidence measures:

- *local accuracy* with parameter k – LA(k)
  LA(k) is commonly used criterion for classifier selection [10]. The confidence of classification of $\vec{x}$ using ϕ is defined as the estimate of local accuracy of ϕ near $\vec{x}$. Let $N_k(\vec{x})$ denote the set of k nearest neighbors from the training (or validation) set, closest to $\vec{x}$ under Euclidean metric. Then $\kappa_\phi^{LA(k)}(\vec{x})$ is defined as the ratio of the number of patterns from $N_k(\vec{x})$ classified correctly by ϕ to the number of all patterns from $N_k(\vec{x})$.
- *local class separability* – (LCS)
  This approach is based on the fact that if the degree of classification to some class is high, and degrees of classification to the remaining classes are low, then the classification is probably right, i.e., the confidence should be high. On the other hand, if all the degrees of classification have similar values, then the confidence should be low. Let ϕ be a normalized possibilistic classifier, $\phi(\vec{x}) = (\mu_1, \ldots, \mu_N)$. Then the LCS confidence of classification is defined using the fololowing formula:

$$\kappa_\phi^{LCS}(\vec{x}) = \frac{1}{(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |\mu_i - \mu_j| \tag{3}$$

Proposition 1 *Let ϕ be a normalized possibilistic classifier, i.e. $\sum_{i=1}^{N} \mu_i = 1$. Then $\kappa_\phi^{LCS}(\vec{x}) \in [0, 1]$.*

Proof: Let $C = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |\mu_i - \mu_j|$. Without loss of generality, let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_N$ – under such conditions the absolute values vanish. It is easy to show that

$$C = \sum_{i=1}^{N-1} \mu_i + \sum_{i=1}^{N-2} \mu_i + \cdots + \mu_1 - \\ - \sum_{i=2}^{N} \mu_i - \sum_{i=3}^{N} \mu_i - \cdots - \mu_N, \quad (4)$$

hence

$$C \leq \sum_{i=1}^{N-1} \mu_i + \sum_{i=1}^{N-2} \mu_i + \cdots + \mu_1, \quad (5)$$

and because $\sum_{i=1}^{N-j} \mu_i \leq 1 \; \forall j = 1, \ldots, N-1$, we get

$$C \leq (N-1), \quad (6)$$

which proves that $\kappa_\phi^{LCS}(\vec{x}) \leq 1$. The fact that $\kappa_\phi^{LCS}(\vec{x}) > 0$ is trivial. ■

We give some examples of LCS for different outputs of normalized possibilistic classifiers:

- $\phi(\vec{x}) = (1, 0, 0, 0)$ – the degree of classification to one of the classes is maximal, and the others are 0. The confidence should be high, and indeed, $\kappa_\phi^{LCS}(\vec{x}) = 1$.
- $\phi(\vec{x}) = (0.8, 0, 0.2, 0)$ – there is some small ambiguity in the classification. The confidence drops to $\kappa_\phi^{LCS}(\vec{x}) = 0.86$.
- $\phi(\vec{x}) = (0.5, 0.5, 0, 0)$ – the degrees of classification to the first and second class are the same. $\kappa_\phi^{LCS}(\vec{x}) = 0.66$
- $\phi(\vec{x}) = (0.4, 0.4, 0.2, 0)$ – ambiguity increases, but still $\mu_4 = 0$. $\kappa_\phi^{LCS}(\vec{x}) = 0.46$
- $\phi(\vec{x}) = (0.4, 0.4, 0.1, 0.1)$ – all the degrees of classification are > 0. $\kappa_\phi^{LCS}(\vec{x}) = 0.4$
- $\phi(\vec{x}) = (0.25, 0.25, 0.25, 0.25)$ – all the degrees of classification are the same, confidence should be minimal. $\kappa_\phi^{LCS}(\vec{x}) = 0$

The examples above show that LCS expresses some measure of confidence of classification using normalized possibilistic classifiers. However, the formula (3) can not be used for non-normalized classifiers:

- $\phi(\vec{x}) = (1, 0, 0, 0)$ – $\kappa_\phi^{LCS}(\vec{x}) = 1$. This is as expected.
- $\phi(\vec{x}) = (1, 1, 0, 0)$ – in this case, we do not know to which of the classes C_1 or C_2 to classify, so the confidence should be lower; however $\kappa_\phi^{LCS}(\vec{x}) = 1.33$.

  This behavior implies that (3) has to be modified for non-normalized classifiers. However, all the classifiers we used in our experiments were normalized, so we used LCS in the form of (3).

The advantage of LCS over LA is its lower time complexity. While LA needs to find the set of neighbors, and to classify all of them, LCS performs only a simple arithmetic operation on a vector of length N.

State-of-the-art methods for classifier combining do not use both evidence and confidence of classification heavily. In classifier selection, confidence is used to select a classifier, and the evidence of other classifiers is discarded. Simple algorithms for classifier aggregation (mean value, product, maximum, minimum, etc.) use the evidence of classification only, and they disregard the confidence. Advanced classifier aggregation methods (weighted mean, fuzzy integral, etc.) incorporate confidence into aggregation, but only global confidence measures (i.e., measures independent on the current pattern, e.g. based on validation accuracy of the classifiers) are commonly used.

However, by incorporating local confidence measures (like LA or LCS) into such algorithms, their performance could be improved. To show this, we propose a simple classifier aggregation algorithm, which utilizes the concept of confidence of classification, the Filtered Mean Value Aggregation algorithm, and study its performance on two datasets. The details are given in the next section.

4. Experiments

To show that the concept of confidence of classification can improve state-of-the-art methods for classifier combining, we developed a simple algorithm, called Filtered Mean Value Aggregation (FMVA), and compared it to two other methods, Dynamic Best Selection (DBS) and Mean Value Aggregation (MVA), on two datasets from the UCI repository [14] – the Pima and Balance datasets.

The algorithms used in the experiments are described in the next section.

4.1. Algorithm Description

Let $(\phi_1, \ldots, \phi_r)$ be a team of classifiers, (2) the output of the team for a pattern $\vec{x}$.

1. *Mean Value Aggregation* – MVA is an classifier aggregation method. MVA computes mean value of degree of classification to each class, i.e. the

aggregated degree of classification to class C_j is defined as the average of the degrees of classification to class C_j through all the classifiers in the team:

$$\mu_j = \frac{1}{r} \sum_{i=1}^{r} \mu_{i,j}. \qquad (7)$$

2. *Filtered Mean Value Aggregation* – FMVA is a modification of MVA, the difference being that prior to computing the mean value, classifiers with confidence of classification of the current pattern lower than some threshold T are discarded. If $T = 0$, FMVA coincides with MVA. If there are no classifiers with confidence higher than T (i.e., all the classifiers would be discarded), T is lowered to the value of maximal confidence in the team.

3. *Dynamic Best Selection* – DBS is a classifier selection algorithm. From the team $(\phi_1, \ldots, \phi_r)$, the classifier with the maximal confidence κ_{max} is selected. If there is more than one classifier with confidence κ_{max}, a random one among them is selected.

4.2. Experimental Results

For the experiments, we used an ensemble of classifiers $(\phi_1, \ldots, \phi_r)$, constructed using the Multiple Feature Subset method, i.e., we created classifiers with all possible combinations of features (all 1-D classifiers, all 2-D classifers, etc.). As the Balance dataset is 4-D, the resulting ensemble consisted of 15 classifiers, and as the Pima dataset is 8-D, the resulting ensemble consisted of 255 classifiers.

For the Pima dataset, the base classifiers were Bayesian classifiers [15], for the Balance dataset, we used Fuzzy k-NN classifiers [16].

The combination of the ensemble was done using the MVA (classifier aggregation), FMVA (with threshold T increasing from 0.1 to 1.0 – i.e., with increasing classifier-selection-like behavior), and DBS (classifier selection) methods. As confidence measures for FMVA and DBS, we used both LA(20) and LCS. All the algorithms were implemented using the Java programming language.

The results from experimental testing on the Pima and Balance datasets are shown in Fig. 1 and 2, respectively. We measured mean test error rate and standard deviation of test error rate (in %) from 10-fold crossvalidation.

From the figures, we can see that FMVA performs most often better than both of the other two methods. For the Pima dataset, MVA achieves about 26% error rate, DBS with LA(20) confidence measure about 28%, DBS with LCS confidence measure about 25%. By fine-tuning the threshold for FMVA, we can achieve less than 24% error rate.

For the Balance dataset, the improvement is even more apparent – MVA achieves about 18.5%, DBS with LA(20) nearly 20%, DBS with LCS nearly 25%, while FMVA can be fine-tuned to approx. 14% for both LA(20) and LCS.

In all of the four figures, we can see the following trend: with increasing T, the error rate first decreases to some point, and then it starts to increase again. This can be interpreted as follows: if T is too low, classifiers with low confidence (which probably yield incorrect predictions) are used in the aggregation, confusing the rest of the team. If the threshold is too high, there is only a small number of classifiers (or just one in the extreme case) used in the aggregation, and the team is less robust to outliers. For some value of T, these two aspects balance, resulting in enough classifiers with reasonably good confidence.

What could be somewhat surprising on the first sight, is the relatively big gap between DBS and FMVA with $T = 1$, which is particularly apparent for the Balance dataset. This is in contrast with the guess that these two algorithms should perform comparably. However, this notion is incorrect – in DBS, always only one classifier is used, while in the case of FMVA with $T = 1$, there is usually more than one classifier with confience equal to one (or less than one if T has to be lowered), so the prediction is always based on aggregation of some small number of classifiers. As the figures show, even such detail can improve the classification slightly.

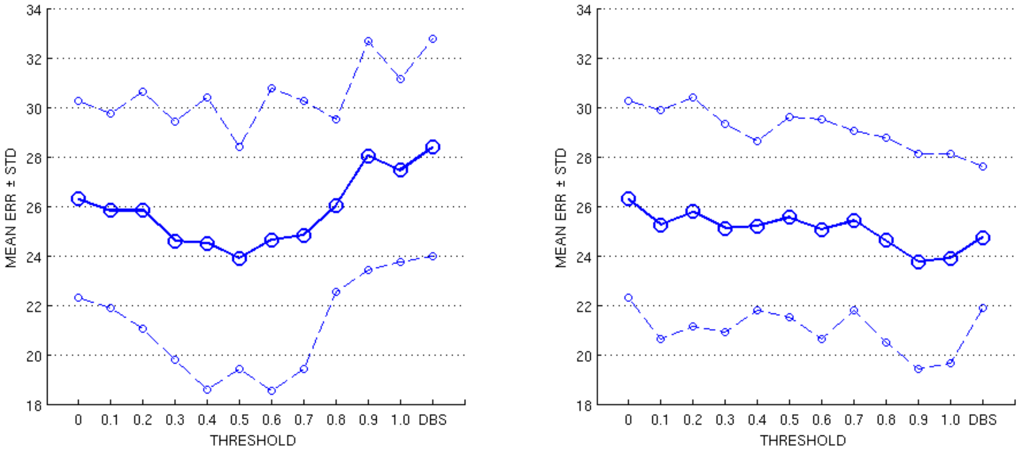

Figure 1: Mean ± standard deviation of the test error rate for the Pima dataset for MVA ($Threshold = 0$), FMVA ($Threshold = 0.1 - 1$), and DBS. Two confidence measures were used – LA(20) (left) and LCS (right).

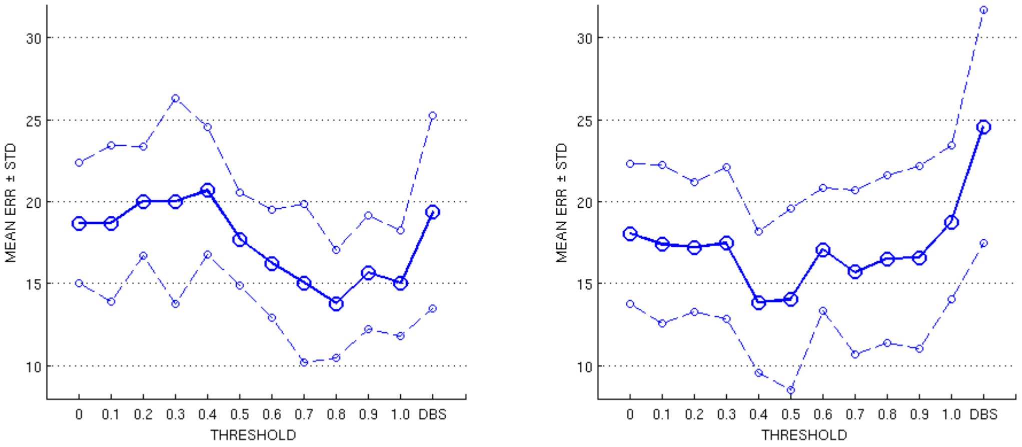

Figure 2: Mean ± standard deviation of the test error rate for the Balance dataset for MVA ($Threshold = 0$), FMVA ($Threshold = 0.1 - 1$), and DBS. Two confidence measures were used – LA(20) (left) and LCS (right).

5. Summary

In this paper, we intorduced the concept of confidence of classification, which can be used both as a criterium for classifier selection and for modifying classifier aggregation methods. We defined two confidence measures (the local accuracy and the local class separability), and introduced simple algorithm for classifier aggregation which uses the concept of confidence of classification – the Filtered Mean Value Aggregation algorithm. Experimental results show that even such a simple modification of the Mean Value Aggregation algorithm can yield improvements in the classification.

However, the concept of confidence of classification can be incorporated into many classifier combining techniques, possibly resulting in very successful methods. In addition, other confidence measures than those reported in this article can be used to further improve the algorithms. This is the topic of our future research.

References

[1] L. I. Kuncheva, J. C. Bezdek, and R. P. W. Duin, "Decision templates for multiple classifier fusion: an experimental comparison.," *Pattern Recognition*, vol. 34, no. 2, pp. 299–314, 2001.

[2] O. Melnik, Y. Vardi, and C.-H. Zhang, "Mixed group ranks: Preference and confidence in clas-

sifier combination.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 8, pp. 973–981, 2004.

[3] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[4] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *International Conference on Machine Learning*, pp. 148–156, 1996.

[5] L. I. Kuncheva, "Using diversity measures for generating error-correcting output codes in classifier ensembles," *Pattern Recogn. Lett.*, vol. 26, no. 1, pp. 83–90, 2005.

[6] S. D. Bay, "Nearest neighbor classification from multiple feature subsets," *Intelligent Data Analysis*, vol. 3, no. 3, pp. 191–209, 1999.

[7] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles," *Machine Learning*, vol. 51, pp. 181–207, 2003.

[8] X. Zhu, X. Wu, and Y. Yang, "Dynamic classifier selection for effective mining from noisy data streams," in *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, (Washington, DC, USA), pp. 305–312, IEEE Computer Society, 2004.

[9] M. Aksela, "Comparison of classifier selection methods for improving committee performance.," in *Multiple Classifier Systems*, pp. 84–93, 2003.

[10] K. Woods, J. W. Philip Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 4, pp. 405–410, 1997.

[11] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 3, pp. 226–239, 1998.

[12] L. I. Kuncheva, "Fuzzy versus nonfuzzy in combining classifiers designed by boosting," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 729–741, 2003.

[13] M. R. Ahmadzadeh and M. Petrou, "Use of Dempster-Shafer theory to combine classifiers which use different class boundaries.," *Pattern Anal. Appl.*, vol. 6, no. 1, pp. 41–46, 2003.

[14] C. B. D.J. Newman, S. Hettich and C. Merz, "UCI repository of machine learning databases," 1998. www.ics.uci.edu/∼mlearn/MLRepository.html.

[15] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[16] J. M. Keller, M. R. Gray, and J. A. Givens, Jr., "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 4, pp. 580–585, 1985.