2012 International Conference on Frontiers in Handwriting Recognition

A Novel Naive Bayes Voting Strategy for Combining Classifiers

C. De Stefano, F. Fontanella, A. Scotto di Freca
Department of Electrical and Information Engineering (DIEI)
University of Cassino and Southern Lazio
Cassino (FR), Italy
{*destefano, fontanella, a.scotto*}*@unicas.it*

***Abstract*—Classifier combination methods have proved to be an effective tool for increasing the performance in pattern recognition applications. The rationale of this approach follows from the observation that appropriately diverse classifiers make uncorrelated errors. Unfortunately, this theoretical assumption is not easy to satisfy in practical cases, thus reducing the performance obtainable with any combination strategy. In this paper we propose a new weighted majority vote rule which try to solve this problem by jointly analyzing the responses provided by all the experts, in order to capture their collective behavior when classifying a sample. Our rule associates a weight to each class rather than to each expert and computes such weights by estimating the joint probability distribution of each class with the set of responses provided by all the experts in the combining pool. The probability distribution has been computed by using the *naive Bayes* probabilistic model. Despite its simplicity, this model has been successfully used in many practical applications, often competing with much more sophisticated techniques. The experimental results, performed by using three standard databases of handwritten digits, confirmed the effectiveness of the proposed method.**

***Keywords*-Classifier Ensembles; Combining Rules; Naive Bayes Classifiers; Neural Networks.**

I. Introduction

Classifier combination methods have proved to be an effective tool for increasing the performance in pattern recognition applications. The rationale of this approach follows from the observation that the errors produced by appropriately diverse classifiers, such as those based on different classification strategies or working on different feature sets, are likely to be uncorrelated. This implies that, by combining their responses, the overall classification accuracy could be improved [1], [2], [3].

Unfortunately, the theoretical assumption that the classifiers to be combined make uncorrelated errors is not easy to satisfy in practical cases, especially when their number is high. In fact, as the number of classifiers increases, it may happen that a correct classification provided by some classifiers is overturned by the convergence of other classifiers on the same wrong decision, thus reducing the performance obtainable with any combination strategy.

Among the different techniques proposed in the literature, linear combining rules are the most widely used approaches for combining the classification results provided by different experts [4]. In this framework, a natural way of implementing such rules is using some form of voting in which classifier outputs may be simply averaged or weighted by using an estimate of their global reliability [5], [6], [7]. Such an estimate can be computed, for instance, by simply considering the recognition rate of each single expert on a training set or by maximizing the performance of the whole set of experts during a learning phase. Simple and weighted majority vote rules belonging to this last category are used in popular ensemble learning algorithms, such as Bagging [8] and Boosting [9], and have been successfully adopted for a large number of applications.

It should be noted, however, that the introduction of some confidence measures for weighting the results provided by a pool of experts does not allow us to solve in the general case the drawbacks due to correlation among errors. In fact, when the measures are separately computed on each expert, once and forever, the combining rule gives more importance to the responses of top performing classifiers: this fact may significantly reduce some of the advantages of classifiers combination because less performing classifiers are often added to the pool of experts to correctly recognize those samples which have not been adequately learned by the top ones.

Even when the confidence measures are jointly computed evaluating the performance of the whole set of experts, the results may be not satisfactory: experimental studies on the classifier diversity obtained with bagging and boosting have shown that these techniques do not ensure to obtain sufficiently diverse classifiers [10], [11]. In particular, as regards boosting, in [11] it has been observed that while at first steps highly diverse classifiers are obtained, as the boosting process proceeds classifier diversity strongly decreases.

When the experts to be combined directly provide a confidence measure for each classified sample, a weighted majority vote rule may be simply obtained by using such measures as weights. In this case, however, there is a possible lack of consistency because different confidence measures, defined for different classification schemes working on different feature spaces, are compared or combined into an overall measure.

Moving from these considerations, we propose a new

978-0-7695-4774-9/12 $26.00 © 2012 IEEE
DOI 10.1109/ICFHR.2012.166

weighted majority vote rule which try to solve the main drawbacks of the above mentioned techniques. Our rule associates a weight to each class rather than to each expert and computes such weights by estimating the joint probability distribution of each class with the set of responses provided by all the experts in the combining pool. In this way, even less performing experts may effectively contribute to the recognition of samples not adequately learned by the top ones. Moreover, there are no inconsistencies since the reliability of each expert in assigning an input sample to a certain class is jointly evaluated through the probabilistic model.

In this study, we have computed the conditional probability distribution of each class by using the *naive Bayes* probabilistic model. We have chosen this model because of both its simplicity and its effectiveness. In fact, it has been shown in the literature that, despite the strong assumption of independence among variables, this approach results remarkably successful in practice, often competing with much more sophisticated techniques [12], [13]. Naive Bayes has proved to be effective in many practical applications, including text classification, intrusion detection systems and systems performance management [14], [15].

The main advantage of the naive Bayes approach comes from the computational efficiency of the learning procedure, which exhibits a linear computational complexity with respect to the number of random variables to be modeled (the number of experts in our case).

The experimental results, performed by using three standard databases of handwritten digits, confirmed the effectiveness of the proposed method.

The remainder of the paper is organized as follows: Section 2 illustrates the naive Bayes Classifier, Section 3 presents the architecture of the method, Section 4 reports the experimental results, while some concluding remarks are eventually left to Section 5.

II. The Naive Bayes Classifier

A naive Bayes Classifier (nBC) is a probabilistic classifier which applies the Bayes' theorem with a strong (naive) assumption: it is assumed that the features describing the objects to be classified are statistically independent each other. Such statistical model is often referred in the literature as the "independent feature model" [16]. As anticipated in the Introduction, in spite of this strong assumption, nBc have demonstrated to be very effective in many real world–applications.

Given a feature space $\mathcal{X}$ in which the objects to be classified are represented by m feature variables $f_1, \ldots, f_m$, the nBC is based on the Bayes' theorem:

$$p(c|f_1, \ldots, f_m) = \frac{p(c)p(f_1, \ \ldots, f_m|c)}{p(f_1, \ \ldots, f_m)} \tag{1}$$

where c is the class variable. Usually, the denominator of the above equation is not considered since it is constant with respect to the class variable c.

According to the underlying assumption of the nBC, namely the statistical independence of the features, the above equation can be rewritten in the following way:

$$p(c|f_1, \ldots, f_m) = \frac{1}{Z}p(c)\prod_{i=1}^{m} p(f_i|c) \tag{2}$$

where $Z = p(f_1, \ \ldots, f_m)$, is a scaling factor not depending on c. According to the maximum likelihood estimation, the model parameters, are estimated from a training set of data (say $\mathcal{D}$). The class prior is estimated as follows: if n_k is the number of samples in $\mathcal{D}$ belonging to the class k and $n_{\mathcal{D}}$ is the total number of samples in $\mathcal{D}$, then $P(c = k) = n_k/n_{\mathcal{D}}$. As concerns the estimation of the feature distribution parameters, two distinct approaches can be followed depending on whether the features assume continuous or discrete values. Details can be found in [13].

Once the above parameters have been estimated, nBC classifies an unknown sample x represented by a feature vector $\mathbf{x} = (f_1, \ldots, f_m)$ according to the *maximum a posteriori* decision rule:

$$e(\mathbf{x}) = \arg\max_k p(c = k)\prod_{i=1}^{m} p(f_i|c = k) \tag{3}$$

Despite the fact that the independence assumptions may be in some cases inaccurate, nBC has several properties that make it very effective in practical applications. For instance, the factorization of the class conditional feature distributions means that each distribution can be independently estimated. This helps to avoid the the curse of dimensionality problem, i.e. the need for very large data sets, whose number of samples scale exponentially with the number of features.

III. The Combiner Architecture

Consider the responses $e_1, \ldots, e_L$ provided by a set of L experts $E_1, \ldots, E_L$ for an input sample x in a N class problem, and assume that such responses constitute the input to the combiner, as shown in Fig.1. Each expert E_i gives as output the label of the class assigned to the sample x, in the set $C = \{C_1, \ldots, C_N\}$, while the combiner provides the final classification result. In this stage the combiner can be defined as a higher level classifier that works on a L-dimensional discrete-values feature space. The combiner uses a supervised learning strategy, which consists in observing both the responses $\{e_1, \ldots, e_L\}$ and the true class label c for each sample of a training set, in order to compute the conditional probability $p(c|e_1, \ldots, e_L)$.

Once this conditional probability has been learned from a set of training data, the combiner classifies each unknown sample by using a weighted voting strategy.

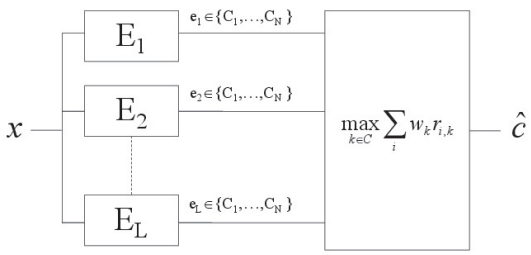

Figure 1. The Combiner scheme.

Before introducing the proposed weighted voting rule, let us briefly recall the main properties of two basic voting rules, namely the majority vote and the weighted majority vote. According to the Majority Vote rule (hereafter MjV) the class $\hat{c}$ of an unknown sample x is computed as:

$$\hat{c} = \max_{k \in C} \sum_{i} r_{i,k}$$

where $r_{i,k}$ is a function whose value is 1 when the classifier E_i classify the sample x as belonging to the class k, and 0 otherwise. Such voting strategy is the most *democratic* one, since each classifier contributes in the same manner to the final decision. As previously discussed, however, the higher the correlation among classifiers, the lower the performance of the rule.

The weighted Majority Vote rule (hereafter wMjV) is less *democratic* then the previous one, since it uses some reliability measures for weighting the responses provided by the experts. These measures, are generally obtained by considering the recognition rate of each expert on a training set [3]. In this case the class $\hat{c}$ assigned to an unknown sample x is:

$$\hat{c} = \max_{k \in C} \sum_{i} R_i r_{i,k}$$

where R_i represents the reliability of the expert E_i. Note that the weights are preliminary computed during a training phase and their values do not change in the operative phase. When the experts to be combined provide also a reliability value, together with the classification result, it should be also possible in principle to use such measures as weights. As discussed in the introduction, however, this choice may lead to inconsistencies because the reliability measures refer to different classification schemes working on different feature spaces and their values cannot be directly compared.

In this study, we propose a new weighted majority vote rule, which uses the conditional probability $p(c|e_1, \ldots, e_L)$ for weighting the votes of each class to be recognized rather than the vote of each expert. According to the Naive Bayes theory, such conditional probability may be computed through Eq. 2, but considering as random variables to be modeled the set of classifier responses $\{e_1, \ldots, e_L\}$ instead of the set of features $\{f_1, \ldots, f_m\}$. Thus Eq. 2 can be rewritten as:

$$p\left(c|e_1, \ldots, e_L\right) = \frac{1}{Z}\, p(c) \prod_{i=1}^{L} p(e_i|c) \tag{4}$$

where $Z = p(e_1, \ \ldots, e_L)$, is a scaling factor not depending on c and the random variables e_i assume discrete values as they represent the responses provided by the experts. Moreover it is assumed that variables e_i are independent each other, but all dependent on c. Under these assumptions, $Z = \prod_{i=1}^{L} p(e_i)$ and $p(e_i|c)$ is obtained by counting the occurrences of each value of e_i for each value of the true class c in a training set. In the Bayesian Network jargon, the result of each single count represents a *parameter* and the whole process of computing all the parameters is defined as *parameter learning*. In case of a N class problem, $N * N$ parameters must be computed for each expert. Considering that:

$$p(c, e_1, \ldots, e_L) = p(c|e_1, \ldots, e_L)\, Z$$

and that the term Z does not depends on c, without loosing generality, we can consider the joint probability $p(c, e_1, \ldots, e_L)$ instead of the conditional probability $p(c|e_1, \ldots, e_L)$. From Eq. 4 we obtain:

$$p(c, e_1, \ldots, e_L) = p(c) \prod_{i=1}^{L} p(e_i|c)$$

Thus, the weight w_k associated to a class k is:

$$w_k(e_1, \ldots, e_L) = p(c = k) \prod_{i=1}^{L} p(e_i|c = k) \tag{5}$$

An high value for the the weight $w_k(e_1, \ldots, e_L)$ means that the set of responses $(e_1, \ldots, e_L)$ provided by the experts for an unknown sample is very frequent in the training set in correspondence of the class k.

The proposed weighted majority vote rule, denoted as Naive Bayes weighted Majority Vote rule (hereafter NB-wMjv) computes the class $\hat{c}$ of the unknown sample x by using the formula:

$$\hat{c} = \max_{k \in C} \sum_{i} w_k r_{i,k} \tag{6}$$

Summarizing, our system operates in two different modalities: during the learning phase, a training set of samples is used for learning the parameters of the Naive Bayes Classifier. In the operative phase, for each input sample x the set of responses $(e_1, \ldots, e_L)$ provided by the experts is

used to compute the weights $w_k(e_1, \ldots, e_L),\ 1 \leq k \leq N$ and then the final decision is taken according to the Eq. 6.

The rationale of our approach is to jointly analyze the responses provided by all the experts, in order to capture the collective behavior of the whole system when classifying a sample. This approach tries to solve some of the main questions in combining classifiers: should the decision provided by each expert be considered equally reliable, or should the decision delivered by the most competent experts be accepted without giving importance to the majority consensus? According to our method, the decision of each expert must be evaluated only in the context of the decisions provided by all the other experts, without considering the performance of each single classifier. In this way, even if for certain samples the majority of the experts provide a wrong decision, the whole set of responses may constitute a pattern adequately learned by the Naive Bayes Classifier, for which a high probability (not necessarily the highest) may be assigned to the correct class. Consider, for instance, a case in which the classifiers E_4 and E_7 confuse the class C_1 with the class C_3 in 20% and in 40% of the cases, respectively. After the parameter learning we have $p(e_4 = C_1|c = C_3) = 0.2$ and $p(e_7 = C_1|c = C_3) = 0.4$: these values contribute to increase the probability to obtain a correct decision when classifying a sample belonging to the class C_3, even if both the experts E_4 and E_7 provided a wrong result.

Finally, we want to remark that the our approach try to overcome one of the main drawbacks related to the estimation of the joint probability distribution of the true class and the ensemble responses, e.g. the need of large sets of data, which may be not available in real world applications. In fact, as above discussed, in case of a N class problem the Naive Bayes Classifier requires, for each expert, the learning of $N * N$ parameters whose values are estimated during a training phase looking at the set of responses provided by that expert for each sample of a training set. When a set of responses provided by the experts represents a pattern not adequately learned by the probabilistic model, more classes may have similar joint probabilities, thus making ineffective a maximum a posteriori probability (MAP) rule. According to our weighted majority vote combining rule, instead, when more classes exhibit similar joint probabilities, the final decision mainly depends on the number of votes received by such classes. On the contrary, when a set of responses provided by the experts represents a pattern adequately learned by the probabilistic model, there is just one class exhibiting a high joint probability value: this value will be used to weight the votes received by such class thus making very likely that it will be chosen as final decision.

IV. Experimental Results and Discussion

In order to ascertain the effectiveness of the proposed approach, three real world datasets involving handwritten characters have been taken into account.

The following datasets have been used for testing the proposed approach: Mfeat, Optodigit and Pendigit. These datasets are publicly available from the UCI machine learning repository [17]. From each dataset, we have randomly extracted three sets of data: TR1, TR2 and TS. TR1 and TR2 have been used as training sets for the classifiers and for the combiner respectively, while TS has been used for the whole system performance evaluation. Dataset partitions are summarized in Table I. In the following an accurate description of each data set is given.

The Mfeat dataset (Multiple Features Data Set) contains 2000 instances of handwritten digits, 200 for each digit, extracted from a collection of Dutch utility maps. Data are described by using six different sets of features, totaling 649 features. Each set of features has been used to describe all the handwritten digits, and arranged in a separate dataset. For each dataset, the type of features and their number are the following: Fourier coefficients of the character shapes (76 features); Zernike moments (47); morphological features (6); Karhunen-Love coefficients (64); pixel averages in 2 x 3 windows (240); profile correlations (216). Starting from the provided datasets we generated a dataset (DS) obtained by merging all the descriptions included in the previous ones, in such a way to describe each sample by the whole set of 649 available features.

The second considered dataset is the Optical Recognition of Handwritten Digits Data Set (Optodigit). It contains 5620 samples equally distributed among the ten classes. Each sample is described by 64 features. Such data have been obtained by preprinted forms, extracting normalized 32x32 bitmaps of handwritten digits. Each bitmap is divided into non-overlapping blocks of 4x4 and the number of black pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range $[0, 16]$. As a consequence, a character is represented by a feature vector of 64 elements where each element contains a value of a 8x8 matrix.

The third dataset taken into account is Pendigit. It contains 11220 samples, and has been created by collecting 250 samples from 44 writers. The samples written by 30 writers has been used for building the training set, while the digits written by the remaining 14 writers has been used as test set. The data has been collected by using a pressure sensitive tablet with an integrated LCD display. Each sample is represented by a set of (x, y) coordinates of the pen, recorded at a fixed time interval (100 milliseconds). The data collected from the tablet has been normalized in order

Table I
Dataset partition sizes.

Dataset	TR1	TR2	TS
Mfeat	700	650	650
Optodigit	1911	1911	1911
Pendigit	3747	3747	3498

Table II
COMPARISON RESULTS AMONG THE NBWMJV RULE AND THE MAJORITY VOTE (MJV) AND WEIGHTED MAJORITY VOTE (WMJV) RULES.

Dataset	Ens.	Best	Avg.	MjV	wMjV	NBwMjv
Mfeat	25	97.54	96.02	97.54	97.69*	**97.85**
	50	96.92	95.58	97.38	97.54*	**97.69**
Optodigit	25	96.74	95.61	97.33	97.38*	**98.05**
	50	96.74	96.12	97.16	97.22*	**97.50**
Pendigit	25	96.99	95.98	96.86	96.91*	**97.06**
	50	96.99	96.09	97.06	**97.06**	97.03*

to make them invariant to translations and scale distortions. Note that, because of the writer speed or the digit, feature vectors may be of different lengths, e.g., feature vectors describing the '1' digit are shorter than the ones describing the '8'. In order to represent all the samples by feature vectors having the same length, the original feature vectors have been spatially resampled by using a linear interpolation between pairs of points. In our data a spatial resampling of 8 points has been performed, as a consequence each digit is represented by a sequence of 8 points $(x_1, y_1), \ldots, (x_8, y_8)$, regularly spaced in arc length.

In our experiments, we have considered the neural network classifier architecture of Learning Vector Quantization (LVQ) [18]. The LVQ networks have been trained by using the well known and largely used Frequency Sensing Competitive Learning (FSCL) algorithm. In order to induce diversity among the learned experts, we have trained networks having a different number of neurons in the output layer. In practice, for each dataset 50 classifiers have been learned, each starting from a different random initialization and a diverse number of output neurons. In particular, the neurons number has been varied from 50 to 1 per class, generating a pool of experts with different recognition rate. From this set of classifiers two pools have been extracted: the first one is made of the first 25 classifiers, i.e. those having a number of output neurons per class in the range $50 - 25$; the second one includes all 50 trained classifiers.

In Table II the best and average test accuracy results of the single classifiers making up the pools are reported (Best and Avg columns). The set of responses provided by these two pools have been used for learning the parameters of the our combining rule, namely the probability distributions of the nBC classifier (eq. 2). The test accuracy rates achieved by the proposed rule are shown in the last columns of Table II (NBwMjv header). Moreover, this table also reports the recognition rates of the two voting strategies mentioned above: majority vote (MjV header) and weighed majority vote (wMjV header). As concerns the weights of the wMjV rule, we used the recognition rates obtained by each classifier on the training set TR2. For each ensemble, the values in bold highlight the best test results, while the second best outcomes are starred.

From the table it can be observed that in all cases but one (Pendigit 25), the proposed approach achieves better performance than those obtained by the two compared combining rules. Even if, in absolute terms, the improvement may seem negligible, it should be considered with respect to the maximum achievable one. Indeed, for all datasets high test accuracy rates ($\approx 97\%$) are obtained by the best two methods, this implies that also little improvements may represent a significant part of the maximum possible improvement. For example, if we consider the Mfeat 25 data, the absolute improvement obtained (0.16%) amounts to the 7% of the maximum possible improvement. In the case of Optodigit 25 results, the absolute enhancement (0.67%) represents the 25% of the best possible one.

Table III
COMPARISON AMONG NBWMJV AND BAGGING AND BOOSTING.

Dataset	Ens.	Bag	Boost	NBwMjv
Mfeat	25	96.60*	93.16	**97.85**
	50	96.63*	93.15	**97.69**
Optodigit	25	95.60*	90.57	**98.05**
	50	95.65*	90.41	**97.50**
Pendigit	25	90.87*	86.67	**97.06**
	50	90.89*	87.14	**97.03**

For the sake of comparison the proposed approach has been also compared with two well known algorithms for building classifier ensembles: bagging [8] and boosting [9]. In order to perform a fair comparison, for both algorithms we used as base classifier an LVQ network. Moreover, as regards the parameter values, all but the number of neurons in the output layer, have been set to the same values used for training the experts employed in our approach. As mentioned above, in our experimental setup the number of neurons in the LVQ output layer has been varied in order to induce expert diversity. But in the case of the bagging and boosting algorithms this option is unavailable, since all the parameters of the base classifier must be set to a single value. In all the experiments performed, such value has been set to 20 neurons per class. The achieved results by bagging, boosting and NBwMjv are summarized in Table III. The values in bold highlight the best test results, while the starred values represent the second best outcomes are starred. From the table it can be seen that the proposed approach always obtains the best result, while the second best results are always achieved by the bagging algorithm. Moreover, it is

worth noting that, the accuracy trends of the bagging and boosting algorithms is similar to those of NBwMjv, Mjv and wMjv: there are no significant difference between the 25 and 50 classifiers pools.

V. Conclusions

We presented a novel weighted majority vote rule for combining the responses provided by a pool of classifiers. The proposed rule has been devised in such a way that even less performing experts can effectively contribute to correctly classify those samples that have not been adequately learned by the top ones. This remarkable result is obtained by estimating the joint probability distribution of each class with the set of responses to be combined. These distributions have been computed by using the naive Bayes probabilistic model, which assumes the statistical independence of the variables to be modeled. When more classes exhibit similar joint probabilities, the final decision mainly depends on the number of votes received by such classes. On the contrary, when there is just one class exhibiting a high joint probability value, this class will be likely assigned to the input sample. This model has been chosen because its computational complexity is linear with respect to the number of experts to be combined. The experimental results confirmed the effectiveness of the proposed method.

Acknowledgment

Many people deserve thanks for making the repository a success. Foremost among them are the donors and creators of the databases and data generators. Special thanks should also go to the past librarians of the repository: David Aha, Patrick Murphy, Christopher Merz, Eamonn Keogh, Cathy Blake, Seth Hettich, and David Newman.

References

[1] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, 1994.

[2] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *PAMI, IEEE Transactions on*, vol. 20, no. 3, pp. 226–239, 1998. [Online]. Available: http://dx.doi.org/10.1109/34.667881

[3] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[4] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *PAMI, IEEE Transactions on*, vol. 27, no. 6, pp. 942–956, 2005.

[5] L. Lam and C. Y. Suen, "Increasing classifiers for majority vote in ocr. theoretical considerations and strategies," in *Workshop on Frontiers in Handwriting Recognition*, 1994, pp. 245–254.

[6] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, May 1992. [Online]. Available: http://dx.doi.org/10.1109/21.155943

[7] A. Della Cioppa, C. De Stefano, and A. Marcelli, "An adaptive weighted majority rule for combining multiple classifiers," in *International Conference on Pattern Recognition 2002*, 2002, pp. 1034–1038.

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

[9] Y. Freund and R. Shapire, "Experiments with a new boosting algorithm," in *Proceedings of ICML96*, 1996, pp. 148–156.

[10] L. Kuncheva, M. Skurichina, and R. P. W. Duin, "An experimental study on diversity for bagging and boosting with linear classifiers," *Information Fusion*, vol. 3, no. 4, pp. 245–258, 2002.

[11] L. Kuncheva and C. Shipp, "An investigation into how adaboost affects classifier diversity," in *Proceedings of IPMU02*, 2002.

[12] I. Rish, "An empirical study of the naive bayes classifier," in *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[13] H. Zhang, "The optimality of naive bayes," in *Proceedings of FLAIRS 2004*, V. Barr and Z. Markov, Eds. AAAI Press, 2004.

[14] N. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of SAC 04*. ACM, 2004, pp. 420–424.

[15] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of ICML06*, W. W. Cohen and A. Moore, Eds., vol. 148. ACM, 2006, pp. 161–168.

[16] R. O. Duda, D. G. Stork, and P. E. Hart, *Pattern classification*. New York: Wiley, 2000.

[17] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[18] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.