On Error Correlation and Accuracy of Nearest Neighbor Ensemble Classifiers

Carlotta Domeniconi and Bojun Yan
Information and Software Engineering Department
George Mason University
carlotta@ise.gmu.edu byan@gmu.edu

Abstract

Recent empirical work has shown that combining predictors can lead to significant reduction in generalization error. Unfortunately, many combining methods do not improve nearest neighbor (NN) classifiers at all. This is because NN methods are very robust with respect to variations of a data set. In contrast, they are sensitive to input features. We exploit the instability of NN classifiers with respect to different choices of features to generate an effective and diverse set of NN classifiers. Interestingly, the approach takes advantage of the high dimensionality of the data. We investigate techniques to decorrelate errors while keeping the individual classifiers accurate. We analyze the results both in terms of error rates and error correlations. The experimental results show that our technique can offer significant performance improvements with respect to competitive methods.

1 Introduction

An ensemble of classifiers succeeds in improving the accuracy of the whole when the component classifiers are both diverse and accurate. Diversity is required to ensure that the classifiers make uncorrelated errors. If each classifier makes the same error, the voting carries that error into the decision of the ensemble, thereby gaining no improvement. In addition, accuracy is required to avoid poor classifiers to obtain the majority of votes. These requirements have been quantified. Under simple voting and error independency conditions, if all classifiers have the same probability of error, and such probability is less than 50%, then the error of the ensemble decreases monotonically with an increasing number of classifiers [15, 2].

One way to generate an ensemble with the required properties is to train the classifiers on different sets of data, obtained by sampling from the original training set [6, 18, 13, 8]. Breiman's *bagging* [6] and Freund and Schapire's *boosting* [13] are well known examples of successful iterative methods for improving the predictive power of classifier learning systems. Bagging uses sampling with replacement. It generates multiple classifiers by producing replicated samples of the data. To classify an instance, a vote for each class j is recorded by every classifier that chooses it, and the class with the most votes is chosen by the aggregating scheme. Boosting uses adaptive sampling. It uses all instances at each repetition, but maintains a weight for each instance in the training set that reflects its importance as a function of the errors made by previously generated hypotheses. As for bagging, boosting combines the multiple classifiers by voting, but unlike bagging boosting assigns different voting strengths to component classifiers on the basis of their accuracy.

Experimental evidence [21] proved that both bagging and boosting are quite effective in reducing generalization error, with boosting providing in general higher improvements. Dramatic error reductions have been observed with decision trees such as CART and C4.5 [6, 21, 13]. This behavior can be explained in terms of the bias-variance components of the generalization error [7]. The variance component measures the scatter in the predictions obtained from using different training sets, each one drawn from the same distribution. The effect of combination is to reduce the variance, that is what both bagging and boosting achieve. In addition, boosting does something more. By concentrating the attention of the weak learner on the harder examples, it challenges the weak learner algorithm to perform well on these harder parts of the sample space, thereby reducing the bias of the learning algorithm.

It turns out that sampling the training set is not effective with NN classifiers [6]. To gain some insights as to why this is the case, let us analyse the conditions under which the bagging procedure is effective. As observed above, bagging reduces the variance component of the generalization error. When the weak learner is unstable with respect to variations in the training set, perturbing the training data can cause significant variability in the resulting predictor. Thus, bagging the ensemble improves accuracy in this case. Suppose the weak learner is the NN classifier. It has been shown that

the probability that any given training point is included in a data set bootstrapped by bagging is approximately 63.2% [6]. It follows that the nearest neighbor will be the same in 63.2% of the nearest neighbor classifiers. Thus, errors are highly correlated, and bagging becomes ineffective.

The fact that NN methods are very robust with respect to variations of the data set makes ensemble methods ineffective. In contrast, NN methods are sensitive to input features. In this paper we exploit such instability of NN classifiers with respect to different choices of features, to generate an effective and diverse set of NN classifiers. We explore the challenge of producing NN classifiers with decorrelated errors which are, at the same time, accurate. In general, the balance where the gains due to decreased correlations outweigh the losses due to reduced information available to individual classifiers must be found to provide the best results.

2 Ensemble of Nearest Neighbors in Weight-Driven Subspaces

As discussed above, kNN methods are very robust with respect to variations of the data set. The stability of nearest neighbor classifiers to variations in the training set makes ensemble methods obtained by bootstrapping the data ineffective. In contrast, kNN techniques are sensitive to features (i.e., intolerant to irrelevant features) [19], and to the chosen distance function [14, 16, 12, 11]. As such, in order to achieve diversity and accuracy with nearest neighbor classifiers, we ought to sample the feature space, to which the kNN method is highly sensitive. The idea is then to exploit the instability of NN classifiers with respect to different choices of features to generate a diverse set of NN classifiers with (possibly) uncorrelated errors.

In [3], the outputs of multiple nearest neighbor classifiers, each having access only to a random subset of features, are combined using simple voting. In [17], instead, the class membership decision is delayed until the aggregation phase. It is shown [3] that random feature selection can increase the diversity without increasing the error rates. This fact results in accuracy improvements on a variety of data sets. However, as also pointed out in [3], the technique has some major drawbacks that cause the degradation in performance observed in some cases. While the *random* selection of features is likely to increase diversity among the classifiers, it gives no guarantee that the selected features carry the necessary *discriminant* information. If they don't, poor classifiers will be generated, and the voting will increase the generalization error.

To reduce the risk of discarding discriminant information, while preserving a reasonable degree of diversity, we propose to perform *adaptive* sampling over the feature space. In particular, in order to keep the bias of individual classifiers low, we use feature relevance to guide the sampling mechanism. This process has the potential of producing accurate classifiers in disagreement with each other. While it is expected that the level of diversity obtained by this adaptive mechanism may be lower than the diversity given by random sampling, the higher accuracy of the individual classifiers should allow the ensemble to improve performance. It is interesting to observe that, since the method uses subsets of features, it will be effective for problems with a large number of dimensions, which is often the case for many applications. Although it defies common sense, sampling in feature space takes advantage of the high dimensionality of the data. The experimental results we present support this conjecture.

2.1 Learning Feature Weights. In this work we use the ADAMENN algorithm to estimate feature relevance, and therefore the corresponding weight vector [12], at any given test point. Other techniques can be considered as well [14, 16, 11]. For completeness, we provide here a brief description of the ADAMENN algorithm. ADAMENN performs a *Chi-squared* distance analysis to compute a flexible metric for producing neighborhoods that are highly adaptive to query locations. Let $\mathbf{x}$ be the nearest neighbor of a query $\mathbf{x}_0$ computed according to a distance metric $D(\mathbf{x}, \mathbf{x}_0)$. The goal is to find a metric $D(\mathbf{x}, \mathbf{x}_0)$ that minimizes $E[r(\mathbf{x}_0, \mathbf{x})]$, where

$$r(\mathbf{x}_0, \mathbf{x}) = \sum_{j=1}^{J} P(j|\mathbf{x}_0)(1 - P(j|\mathbf{x})). \tag{2.1}$$

Here $P(j|\mathbf{x})$ is the class conditional probability at $\mathbf{x}$. That is, $r(\mathbf{x}_0, \mathbf{x})$ is the finite sample error risk given that the nearest neighbor to $\mathbf{x}_0$ by the chosen metric is $\mathbf{x}$. It can be shown [12] that the weighted *Chi-squared* distance

$$D(\mathbf{x}, \mathbf{x}_0) = \sum_{j=1}^{J} \frac{[P(j|\mathbf{x}) - P(j|\mathbf{x}_0)]^2}{P(j|\mathbf{x}_0)} \tag{2.2}$$

approximates the desired metric, thus providing the foundation upon which the ADAMENN algorithm computes a measure of local feature relevance, as shown below.

We first notice that $P(j|\mathbf{x})$ is a function of $\mathbf{x}$. Therefore, we can compute the conditional expectation of $P(j|\mathbf{x})$, denoted by $\overline{P}(j|x_i = z)$, given that x_i assumes value z, where x_i represents the ith component

of $\mathbf{x}$. That is,

$$\begin{aligned}\overline{P}(j|x_i = z) &= E[P(j|\mathbf{x})|x_i = z] \\ &= \int P(j|\mathbf{x})p(\mathbf{x}|x_i = z)d\mathbf{x}.\end{aligned}$$

Here $p(\mathbf{x}|x_i = z)$ is the conditional density of the other input variables defined as

$$p(\mathbf{x}|x_i = z) = p(\mathbf{x})\delta(x_i - z)/\int p(\mathbf{x})\delta(x_i - z)d\mathbf{x},$$

here $\delta(x - z)$ is the Dirac delta function having the properties $\delta(x - z) = 0 \quad if \quad x \neq z$ and $\int_{-\infty}^{\infty}\delta(x - z)dx = 1$. Let

$$r_i(\mathbf{z}) = \sum_{j=1}^{J}\frac{[P(j|\mathbf{z}) - \overline{P}(j|x_i = z_i)]^2}{\overline{P}(j|x_i = z_i)}. \tag{2.3}$$

$r_i(\mathbf{z})$ represents the ability of feature i to predict the $P(j|\mathbf{z})$s at $x_i = z_i$. The closer $\overline{P}(j|x_i = z_i)$ is to $P(j|\mathbf{z})$, the more information feature i carries for predicting the class posterior probabilities locally at $\mathbf{z}$.

We can now define a measure of feature relevance for $\mathbf{x}_0$ as

$$\bar{r}_i(\mathbf{x}_0) = \frac{1}{K_0}\sum_{\mathbf{z}\in N(\mathbf{x}_0)} r_i(\mathbf{z}), \tag{2.4}$$

where $N(\mathbf{x}_0)$ denotes the neighborhood of $\mathbf{x}_0$ containing the K_0 nearest training points, according to a given metric. $\bar{r}_i$ measures how well on average the class posterior probabilities can be approximated along input feature i within a local neighborhood of $\mathbf{x}_0$. Small $\bar{r}_i$ implies that the class posterior probabilities will be well captured along dimension i in the vicinity of $\mathbf{x}_0$. Note that $\bar{r}_i(\mathbf{x}_0)$ is a function of both the test point $\mathbf{x}_0$ and the dimension i, thereby making $\bar{r}_i(\mathbf{x}_0)$ a local relevance measure.

To formulate the measure of feature relevance as a weighting scheme, we first define

$$R_i(\mathbf{x}_0) = \max_j\{\bar{r}_j(\mathbf{x}_0)\} - \bar{r}_i(\mathbf{x}_0),$$

i.e., the more relevant dimension i is, the larger R_i becomes. An exponential weighting scheme is then given by

$$w_i(\mathbf{x}_0) = \exp(R_i(\mathbf{x}_0))/\sum_{l=1}^{q}\exp(R_l(\mathbf{x}_0)).$$

Such weights are real values between 0 and 1, and their sum equals 1. Therefore, they define a probability distribution over the feature space that can be employed in our adaptive sampling mechanism. In addition, the exponential weighting scheme avoids zero values. As such, for each test point and each classifier of the ensemble, any given feature has a non zero probability to be selected. This property guarantees a certain level of diversity among the classifiers of our ensemble. For the details on how to estimate the unknown quantities involved in the feature relevance measure, see [12]. We point out that ADAMENN outperforms decision trees, and other well known locally adaptive classifiers on a variety of data sets [12]. In addition, it has shown accuracy results similar to support vector machines in a variety of cases. Thus, being able to improve upon its performance is a significant objective to achieve.

2.2 Ensemble Algorithm. The general formulation of our approach is as follows:

Input: Number-Of-Classifiers (NoC), Number-Of-Features (NoF), k, test point $\mathbf{x}_0$;

Compute the weight vector $\mathbf{w}_0$ reflecting feature relevance at $\mathbf{x}_0$ (e.g., using the ADAMENN algorithm);

- For 1 to NoC:
  1. Sample NoF features *with* or *without replacement*, according to the probability distribution given by the weight vector $\mathbf{w}_0$;
  2. Use selected features ($SelF$) only (and their weights) to compute the k closest neighbors, according to the weighted Euclidean distance: $D(\mathbf{x}_0, \mathbf{y}) = \sqrt{\sum_{i\in SelF} w_{0i}(x_{0i} - y_i)^2}$;
  3. Classify test point using kNN rule;
- Apply the voting scheme in use among the NoC classifiers.

Output: Decision of the ensemble.

The algorithm has three input parameters: The Number-Of-Classifiers to combine, the Number-Of-features to be selected, and the size k of the neighborhoods. The values of these parameters can be determined based on cross-validation accuracy estimated on the training set for the whole ensemble. When sampling with replacement is used, if a feature is selected more than once, say t times, its weight is multiplied by a factor t for distance computation.

3 Voting Methods

The classifiers can be combined using a simple majority voting. We also investigate an alternative mechanism to combine the classifiers. Instead of computing the most frequent class label within the neighborhood of the test

point, we keep all estimated class posterior probabilities. That is, for each classifier, all class labels of the k nearest neighbors are recorded. After NoC iterations, the test point is assigned to the class that has the most frequent occurrency. This voting scheme selects the class with the largest expected posterior probability in the ensemble. As such, it takes into account not only the "winner" of each classifier, but also the margin of the win. The class with the largest overall margin will be selected by the ensemble.

As a simple example, suppose we have three classifiers and two classes (positive and negative). For a given test point $\mathbf{x}_0$, the recorded labels of its five nearest neighbors ($k = 5$) are as follows: Classifier 1: 2 positives and 3 negatives; Classifier 2: 2 positives and 3 negatives; Classifier 3: 4 positives and 1 negative. The expected class posterior probabilities, estimated by the ensemble, are: $E[P(+|\mathbf{x}_0)] = 8/15$ and $E[P(-|\mathbf{x}_0)] = 7/15$. Thus, the ensemble chooses the positive class. Note that, although the negative class is the winner for the majority of classifiers, its overall margin of win is two. The positive class wins only once but with a (larger) margin of three. Simple voting predicts a negative label in this example.

In addition, we consider the Borda Count method [10]. It is a positional-scoring technique: each candidate class gets 0 points for each last place vote received, 1 point for each next-to-last point vote, and so on up to $C - 1$ points for each first place vote (where C is the number of classes). The candidate class with the largest point total wins the election. When $C = 2$, the Borda Count method reduces to a simple majority voting technique. It is often used for polls which rank sport teams or academic institutions.

4 Experimental Results

We have conducted experiments to compare the accuracy and diversity of Random and Weight-Driven feature subspace methods. Both sampling *with* and *without replacement* have been used. The three voting schemes described above (*Simple*, *Counting*, and *Borda*) were used to compute the decision of the ensemble ($NoC = 200$ classifiers). Tables 1-2 show the error rates and standard deviations obtained on five data sets [5]. We also report the error rates of ADAMENN and kNN using Euclidean distance. The characteristics of each data set (number of dimensions, number of data (N), and number of classes (C)) are given in parenthesis. Leave-one-out cross-validation was used to generate training and test data in each classifier. We have tested values of k between 1 and 5; for NoF we considered values from 1 (or higher, for data with a larger dimensionality) to the total number of dimensions. For each combination of parameter values, the experiment was repeated 10 times, and the average error rate was computed. For all methods compared, validation of parameters was performed over the training data. Tables 3-4 specify the parameter values for the error rates given in Tables 1-2. For each data set and each technique, Tables 3-4 show: the value of k, if sampling with (1) or without (0) replacement was performed, and the number of selected features (NoF).

The results show that our Weight-driven approach offers significant accuracy improvements (over both ADAMENN and the Random approach) for the three data sets with a larger number of dimensions (*spectf-test*, *lung*, *sonar*). For *liver* and *ionosphere* the Random and Weight approaches give similar performances. This result provides evidence that bootstrapping features using an "intelligent" distance metric (Weight-Driven method) takes advantage of the high dimensionality of the data. Thus, it provides an effective method to dodge the curse-of-dimensionality phenomenon. Figures 1-2-3-4-5 plot the error rate as a function of the number of selected features (NoF) for all five data sets considered here. For the Weight-driven technique, and for *ionosphere* and *lung* data, the largest values of NoF are 23 and 50, respectively (see Figures 2-4). This is because eleven and four features, respectively for the first and second data set, received very small weights which were approximated to zero (thus, were never selected). The plots show the robustness of the Weight-driven approach as the number of selected features increase. On the contrary, the error rate of the Random approach can be quite sensitive to the value of the NoF parameter. In particular, the results for the *ionosphere*, *spectf-test*, and *sonar* data clearly demonstrate the drawback of the Random approach: as the fraction of selected features *not* carrying discriminant information increases, poor classifiers are generated, and the voting increases the generalization error.

In some cases the Counting voting method improves performance (with respect to Simple). The Borda technique on *lung* data gives the best result for both Random and Weight-driven algorithms. (Note that we test the Borda voting method only on *lung* data since the other data sets all involve two classes. In such case, the Borda count method reduces to simple voting.) In most cases, sampling without replacement outperformed sampling with replacement (see parameter values in Tables 3-4).

4.1 Measure of Diversity and Accuracy

To measure both the accuracy and the diversity of the classifiers, we make use of the Kappa statistic, κ [9, 20]. In particular, a *Kappa-Error* diagram [20] allows us to visualize the diversity and the accuracy of an ensemble

of classifiers. A Kappa-Error diagram is a scatterplot where each point corresponds to a pair of classifiers. The x coordinate is the value of κ for the two classifiers. Smaller κ values indicate a larger diversity: $\kappa = 0$ when the agreement of the two classifiers equals that expected by chance, and $\kappa = 1$ when the two classifiers agree on every example. The y coordinate is the average error rate of the two classifiers.

We report the Kappa-Error diagrams for the five data sets in Figures 6-7-8-9-10. As expected, the level of diversity obtained with the Weight-driven technique is in general lower than the diversity given by the Random approach. However, the "intelligent" metric employed by the Weight-driven technique allows to reduce bias, and thus achieve a better error rate.

4.2 Reduction of Error Correlations In light of the results provided by the Kappa statistic, we explore the possibility of decorrelating errors by introducing new elements of diversification among the NN classifiers. The benefits and pitfalls of reducing the correlation among classifiers, especially when the training data are limited, are well discussed in [22]. Here we face the challenge of reaching a trade-off between error decorrelation and accuracy in the context of NN classifiers.

We first allow each classifier to customize the number of selected features at each query point $\mathbf{x}_0$. This is achieved as follows. We sort the weight components of $\mathbf{w}_0$ in non increasing order: $w_{01}, \ldots, w_{0q}$ (q is the number of dimensions). The classifier chooses NoF_0 (number of selected features at $\mathbf{x}_0$) to be such that

$$\sum_{i=1}^{NoF_0} w_{0i} \leq f, and \sum_{i=1}^{NoF_0+1} w_{0i} > f,$$

where $f \in (0, 1)$ is an input parameter. Basically, f is the fraction of the total weight (which is always one) captured by NoF_0. The fewer the relevant features, the smaller NoF_0. The actual features are then selected as before, by sampling according to the probability distribution given by the weight vector $\mathbf{w}_0$. We have experimented with the following values of f: 0.2, 0.4, 0.6, 0.8, and 0.9. The error rates corresponding to the cross-validated values of f are shown in Tables 5-6. We observe that the error rates get worst (considerably for lung data), except for *liver* and Weight (Counting). We noticed that these error rates all correspond to $f = 0.9$ (interestingly, except for *liver* and Weight (Counting) where $f = 0.6$). This suggests that the classifiers are highly correlated (as confirmed by the correlation measurements given below). For *liver* data, the value $f = 0.6$ offers the best trade-off between diversity and accuracy. For *lung* data, we observed that 13 points (out of 32) used less that five features. This may have affected the results, since previously the combination of seven or five features gave the best error rates.

In an effort to decorrelate the errors, we have also generated ensembles of a mixture of Random and Weight-driven classifiers. (Two percentage combinations were tested: 50% of each kind; 60% Weight-driven and 40% Random.) The resulting error rates are given in Tables 7-8. Simple voting was used to combine the classification results. As values of the parameters for each kind of classifier, we used the cross-validated ones given in Tables 3-4. Only for *sonar* data a better error rate with respect to both previous error rates of Random and Weight was achieved.

To analyze these results, we have measured the correlation of errors of two classifiers (indexed by 1 and 2) on each class i:

$$\delta^i_{1,2} = \frac{cov(\eta^i_1(\mathbf{x}), \eta^i_2(\mathbf{x}))}{\sigma_{\eta^i_1}\sigma_{\eta^i_2}},$$

where $\eta^i_j(\mathbf{x})$ is the error value (0 or 1) on $\mathbf{x} \in C_i$ of classifier j, and $\sigma_{\eta^i_j}$ is the standard deviation of $\eta^i_j(\mathbf{x})$, computed over all $\mathbf{x} \in C_i$. To account for all classes:

$$\delta_{1,2} = \sum_{i=1}^{C} \delta^i_{1,2} P(i),$$

where $P(i)$ is the prior probability of class i. In our experiments we consider equal priors, and thus

$$\delta_{1,2} = 1/C \sum_{i=1}^{C} \delta^i_{1,2}$$

gives the total error correlation between classifiers 1 and 2.

Sample results for *liver* and *sonar* are given in Tables 9-10. For each data set we summarize the average error correlation values computed between five classifiers. We also report the corresponding error rates of the ensembles. In each case simple voting is used. Weight-C is for Weight-driven with customized number of features. As expected, the Random approach gives very low correlation values. Decreasing the value of f for the Weight-C method, clearly decreases the correlation of errors. Though, in most cases, the gains due to the decreased correlations is outweighed by the loss of information due to the reduced number of features retained. The mixture method is effective in decreasing correlations, though it didn't offer significant accuracy improvements with respect to both the Random and Weight approach alone. Nevertheless, this analysis suggests that combining mixture of different classifiers

Table 1: Average error rates.

(*dim-N-C*)	*liver* (6-345-2)	*ionosphere* (34-351-2)	*spectf-test* (44-267-2)
kNN	32.5	13.7	23.6
ADAMENN	30.7	7.1	19.1
Random (Simple)	29.4 (0.5)	5.8 (0.2)	20.2 (0.4)
Random (Counting)	28.6 (0.5)	5.7 (0.2)	19.9 (0.4)
Weight (Simple)	29.3 (0.5)	6.3 (0.2)	17.6 (0.4)
Weight (Counting)	29.9 (0.5)	6.3 (0.2)	17.7 (0.4)

Table 2: Average error rates.

(*dim-N-C*)	*lung* (54-32-3)	*sonar* (60-208-2)
kNN	50.0	12.5
ADAMENN	37.5	9.1
Random (Simple)	45.0 (0.5)	10.5 (0.3)
Random (Counting)	45.3 (0.5)	10.3 (0.3)
Random (Borda)	44.7 (0.5)	-
Weight (Simple)	35.0 (0.5)	8.3 (0.3)
Weight (Counting)	32.5 (0.5)	8.3 (0.3)
Weight (Borda)	30.9 (0.5)	-

(e.g., each kind performing a different form of adaptive sampling) may offer error rate improvements as well. In our future work we plan to consider other adaptive feature sampling mechanisms, and random projection as well.

5 Conclusions

We have introduced a mechanism to generate an effective and diverse ensemble of NN classifiers. Our analysis shows the difficulty of reaching a good balance between error decorrelation and accuracy in the context of NN classifiers. In an effort to further reduce correlations without increasing error rates, we will consider multiple adaptive mechanisms to sampling in feature space.

Table 3: Parameter values for error rates in Table 1: *k - with (1) or without (0) replacement - NoF.*

	liver	*ionosphere*	*spectf-test*
Random (Simple)	5-1-3	1-0-5	1-0-5
Random (Counting)	3-1-4	1-0-5	1-0-5
Weight (Simple)	5-1-5	1-0-8	5-0-4
Weight (Counting)	3-1-6	1-0-7	1-0-7

Table 4: Parameter values for error rates in Table 2: *k - with (1) or without (0) replacement - NoF.*

	lung	*sonar*
Random (Simple)	5-1-54	1-0-23
Random (Counting)	3-0-49	1-1-29
Random (Borda)	3-0-49	-
Weight (Simple)	3-0-7	1-0-45
Weight (Counting)	3-0-5	1-0-45
Weight (Borda)	3-0-5	-

Table 5: Weight-driven approach with customized number of features: Average error rates.

	liver	*ionosphere*	*spectf-test*
Weight (Simple)	30.3 (0.5)	8.3 (0.3)	18.7 (0.4)
Weight (Counting)	29.1 (0.5)	8.3 (0.3)	18.9 (0.4)

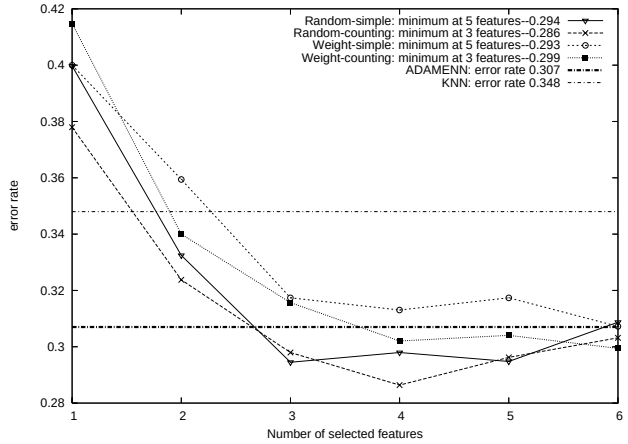

Figure 1: Liver data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

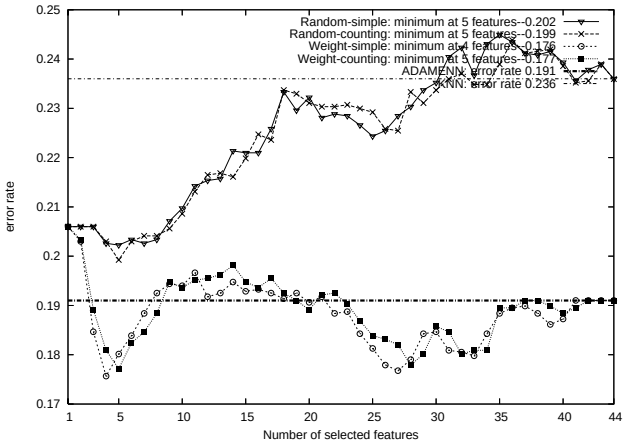

Figure 3: Spectf data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

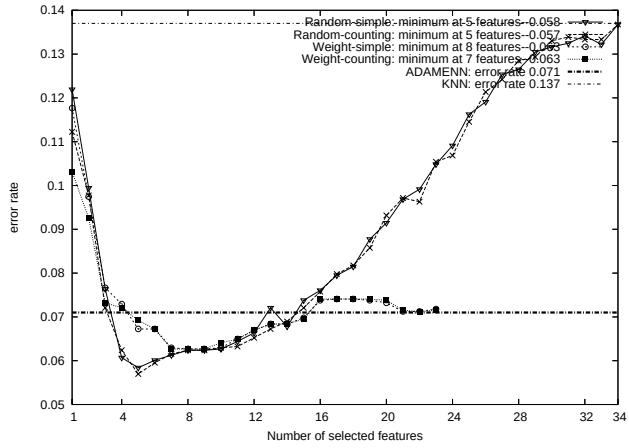

Figure 2: Ionosphere data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

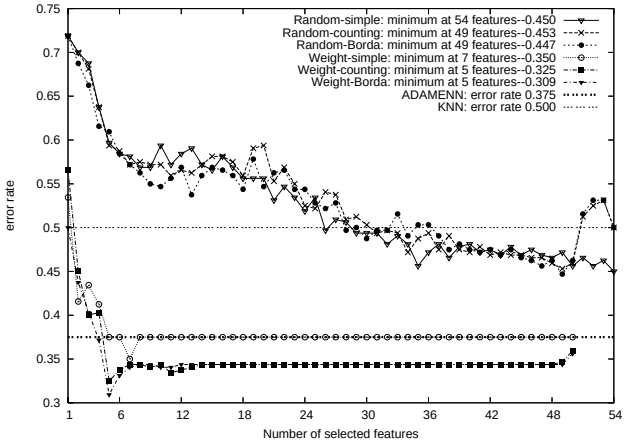

Figure 4: Lung data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

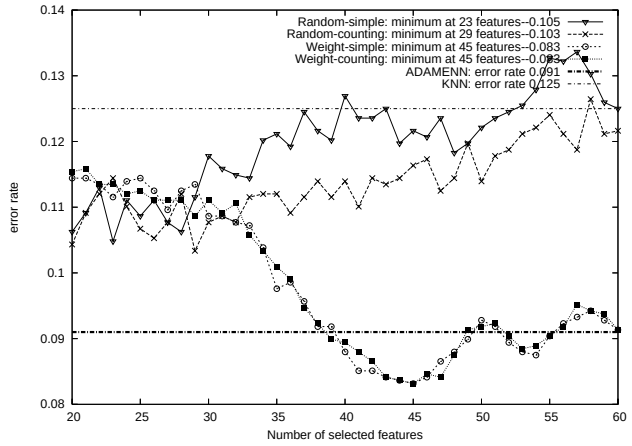

Figure 5: Sonar data: Error rate as a function of the number of selected features for Random and Weight-driven methods.

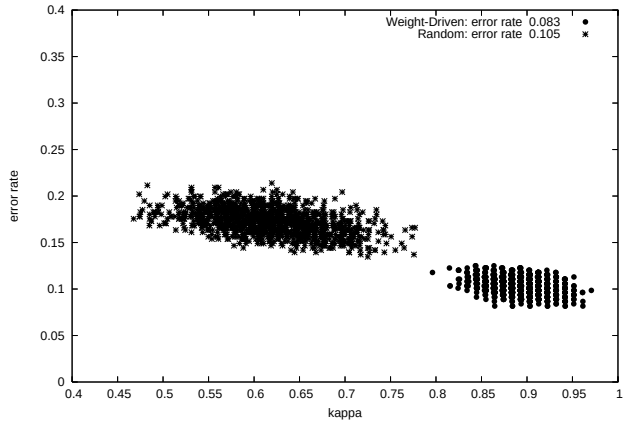

Figure 6: Sonar data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

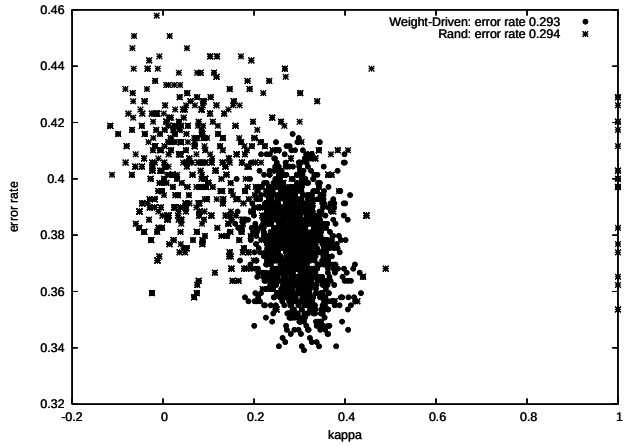

Figure 7: Liver data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

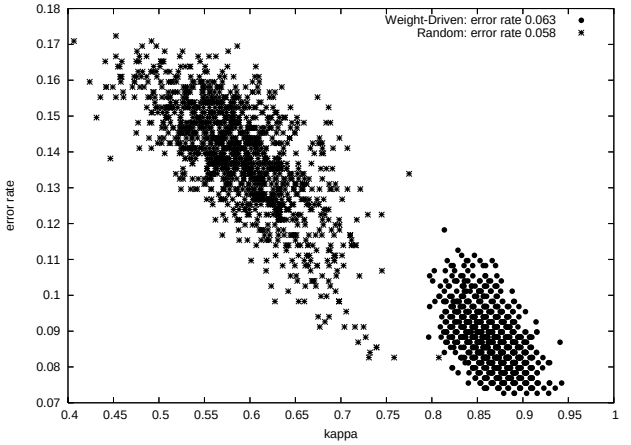

Figure 8: Ionosphere data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

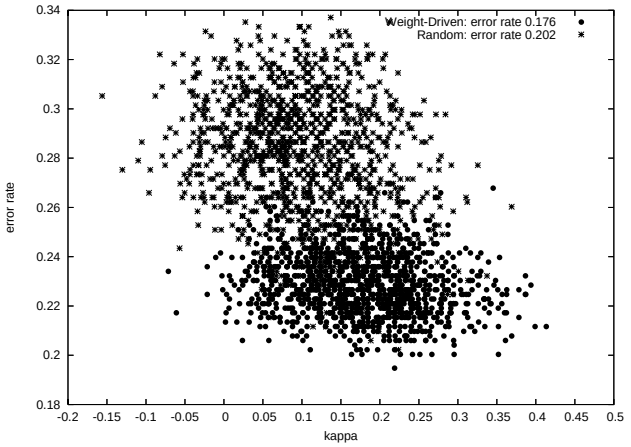

Figure 9: Spectf data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

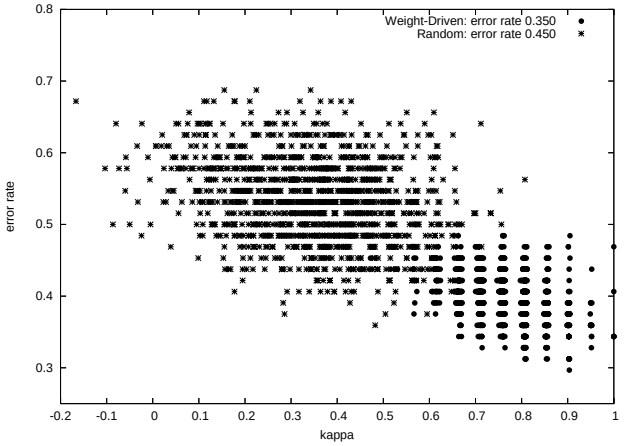

Figure 10: Lung data: Kappa-Error diagram for Random and Weight-Driven Subspace methods.

Table 6: Weight-driven approach with customized number of features: Average error rates.

	lung	*sonar*
Weight (Simple)	46.9 (0.5)	8.7 (0.3)
Weight (Counting)	43.4 (0.5)	8.6 (0.3)
Weight (Borda)	43.4 (0.5)	-

Table 7: Mixture of Weight-driven and random classifiers: Average error rates.

	liver	*ionosphere*	*spectf-test*
Simple voting	30.8 (0.5)	6.0 (0.3)	17.8 (0.4)

Table 8: Mixture of Weight-driven and random classifiers: Average error rates.

	lung	*sonar*
Simple voting	37.5 (0.5)	8.1 (0.3)

Table 9: Average error correlation values and Average error rates: *Liver* data.

	Error Correlation	*Error rate*
Random	0.12	29.4
Weight	0.23	29.3
Weight-C ($f = 0.9$)	0.74	30.3
Weight-C ($f = 0.8$)	0.41	31.4
Weight-C ($f = 0.6$)	0.21	31.6
Mixture	0.11	30.8

Table 10: Average error correlation values and Average error rates: *Sonar* data.

	Error Correlation	*Error rate*
Random	0.34	10.5
Weight	0.69	8.3
Weight-C ($f = 0.9$)	0.72	8.7
Weight-C ($f = 0.8$)	0.66	10.2
Weight-C ($f = 0.6$)	0.42	11.4
Mixture	0.43	8.1

References

[1] Agresti, A. (1990). *Categorical data analysis*. John Wiley & Sons, New York.

[2] Ali, K. M., & Pazzani, M. J. (1996). Error reduction through learning multiple descriptions. *Machine Learning*, **24**:173-202.

[3] Bay, S. D. (1999). Nearest neighbor classification from multiple feature subsets. *Intelligent Data Analysis*, **3**(3):191-209.

[4] Bishop, Y. M. M., Fienberg, S. E., & Holland, P. W. (1975). *Discrete multivariate analysis: theory and practice*. MIT Press, Cambridge.

[5] Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases University of California, Department of Information and Computer Science.

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

[7] Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation* **11**:1493-1517.

[8] P. Chan, P., & Stolfo, S. (1995). A comparative evaluation of voting and meta-learning on partitioned data. *Twelfth International Conference on Machine Learning*.

[9] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psycological Measurement*, **20**(1):37-46.

[10] de Borda, J. (1781). Memoire sur les elections au scrutin, historie de l'academie royale des sciences. Paris.

[11] Domeniconi, C. & Gunopulos, D. (2002). Adaptive nearest neighbor classification using support vector machines. *Advances in Neural Information Processing Systems 14*, MIT Press.

[12] Domeniconi, C., Peng, J., & Gunopulos, D. (2002). Locally adaptive metric nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(9):1281-1285.

[13] Y. Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. *Thirteenth International Conference on Machine Learning*.

[14] Friedman, J. H. (1994). Flexible metric nearest neighbor classification. *Technical Report*, Department of Statistics, Stanford University.

[15] Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**:993-1001.

[16] Hastie, T., & Tibshirani, R. (1996). Discriminant adaptive nearest neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(6):607-615.

[17] Ho, T. K. (1998). Nearest Neighbor in random subspaces. *Joint IAPR International Workshop on Advances in Pattern Recognition*.

[18] Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. *Thirteen International Conference on Machine Learning*.

[19] Langley, P., & Iba, W. (1997). Average-case analysis of a nearest neighbor algorithm. *Thirteenth International Conference on Machine Learning*.

[20] Margineantu, D. D., & Dietterich, T. (1997). Pruning adaptive boosting. *Fourteenth International Confer-*

ence on Machine Learning.

[21] J. R. Quinlan, J. R. (1996). Bagging, boosting and C4.5. *Fourteenth National Conference on Artificial Intelligence*.

[22] Tumer, K., & Ghosh, J. (1996). Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3-4), pp 385-404.