# Nearest Neighbor Ensemble

Carlotta Domeniconi Bojun Yan
Information and Software Engineering Department
George Mason University, Fairfax, VA 22030 USA
carlotta@ise.gmu.edu byan@gmu.edu

## Abstract

*Recent empirical work has shown that combining predictors can lead to significant reduction in generalization error. The individual predictors (weak learners) can be very simple, such as two terminal-node trees; it is the aggregating scheme that gives them the power of increasing prediction accuracy. 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 with possibly uncorrelated errors. Interestingly, the approach takes advantage of the high dimensionality of the data. The experimental results show that our technique offers 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 [13, 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 [5, 15, 11, 10]. Breiman's *bagging* [5] and Freund and Schapire's *boosting* [11] 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 [17] 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 [5, 17, 11]. This behavior can be explained in terms of the bias-variance components of the generalization error [6]. 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 [5]. 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% [5]. 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 with possibly uncorrelated errors.

## 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), and to the chosen distance function [12, 14, 9, 8]. 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. It is shown 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.

In this work we use the ADAMENN algorithm to estimate feature relevance, and therefore the corresponding weight vector [9], at any given test point. Other techniques can be considered as well [12, 14, 8]. ADAMENN performs a *Chi-squared* distance analysis to compute a flexible metric for producing neighborhoods that are highly adaptive to query locations. The weights credited to features by ADAMENN 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 employed in ADAMENN avoids zero weight 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. 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.

In addition, we consider the Borda Count method [7]. 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.

## 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 best error rates and standard deviations obtained on five data sets [4]. We also report the best error rate 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 13; 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. 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. Figure 1 plots the error rate as a function of the number of selected features (*NoF*) for *spectf-test*, *lung*, and *sonar*. For the Weight-driven technique and *lung* data the largest value of *NoF* is 50, because four features received very small weights which were approximated to zero (thus, never selected). In some cases the Counting voting method improves performance (with respect to Simple). The Borda technique gives the best result for both Random and Weight-driven algorithms. In most cases, sampling without replacement outperformed sampling with replacement.

**Table 1. Average error rates.**

|  | *liver* | *ionosphere* | *spectf-test* |
|---|---|---|---|
| ($dim$-$N$-$C$) | (6-345-2) | (34-351-2) | (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) |

**Measure of Diversity and Accuracy**. To measure both the accuracy and the diversity of the classifiers, we make use of the Kappa statistic, $\kappa$ [16]. In particular, a *Kappa-Error* diagram [16] 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 $\kappa$ for the two classifiers. Smaller $\kappa$ 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. For lack of space we report only the Kappa-Error diagram for the *spectf-test* data (Figure 2). Both methods show large diversity in this case. In addition, the "intelligent" metric employed by the Weight-driven technique allows to reduce bias, and thus achieve a better error rate.

## 5. Conclusions

We have introduced a mechanism to generate an effective and diverse ensemble of NN classifiers. In our future work we will investigate techniques to locally customize the number of selected features, and further increase the diversity of classifiers.

## References

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

**Table 2. Average error rates.**

| $(dim\text{-}N\text{-}C)$ | *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) | - |

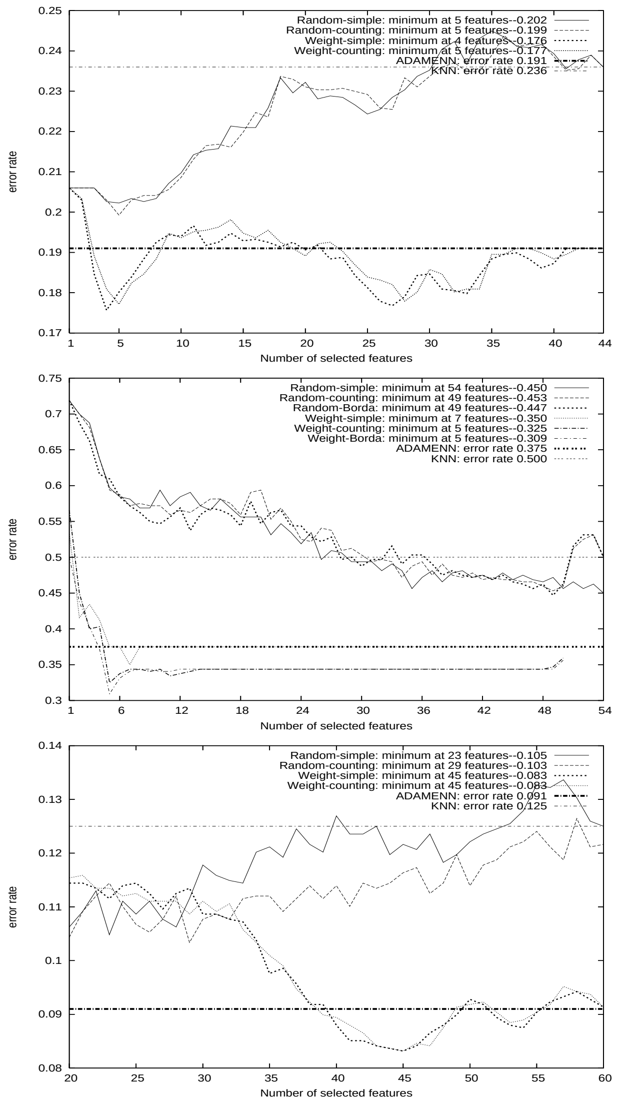

**Figure 1. Error rate as a function of the number of selected features for Random and Weight-driven methods. (Top):** *spectf-test*; **(Middle):** *lung*; **(Bottom):** *sonar*.

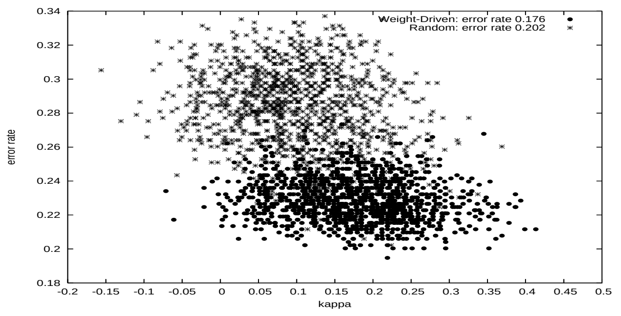

**Figure 2. Kappa-Error diagram for Random and Weight-Driven Subspace methods on** *spectf-test* **data.**

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

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

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

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

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

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

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

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

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

[11] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. *Thirteenth International Conference on Machine Learning*, 1996.

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

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

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

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

[16] D. D. Margineantu and T. Dietterich. Pruning adaptive boosting. *Fourteenth International Conference on Machine Learning*, 1997.

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