Adaptive Boosting for Spatial Functions with Unstable Driving Attributes*

Aleksandar Lazarevic[1], Tim Fiez[2], Zoran Obradovic[1]

[1] School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA
{alazarev, zoran}@eecs.wsu.edu
[2] Department of Crop and Soil Sciences, Washington State University, Pullman, WA 99164-2752, USA
tfiez@wsu.edu

Abstract. Combining multiple global models (e.g. back-propagation based neural networks) is an effective technique for improving classification accuracy by reducing a variance through manipulating training data distributions. Standard combining methods do not improve local classifiers (e.g. k-nearest neighbors) due to their low sensitivity to data perturbation. Here, we propose an adaptive attribute boosting technique to coalesce multiple local classifiers each using different relevant attribute information. In addition, a modification of boosting method is developed for heterogeneous spatial databases with unstable driving attributes by drawing spatial blocks of data at each boosting round. To reduce the computational costs of k-nearest neighbor (k-NN) classifiers, a novel fast k-NN algorithm is designed. The adaptive attribute boosting applied to real life spatial data and artificial spatial data show observable improvements in prediction accuracy for both local and global classifiers when unstable driving attributes are present in the data. The "spatial" variant of boosting applied to the same data sets resulted in highly significant improvements for the k-NN classifier, making it competitive to boosted neural networks.

Keywords: multi-strategy learning, boosting, attribute representation, spatial databases, fast k-NN classifier.

1. Introduction

Many large-scale data analysis problems involve an investigation of relationships between attributes in heterogeneous databases. Large data sets very often exhibit attribute instability, such that the set of relevant attributes is not the same through the entire data space. This is especially true in spatial databases, where different spatial regions may have completely different characteristics.

It is known in machine learning theory that combining multiple classifiers is an effective technique for improving prediction accuracy. There are many general combining algorithms such as bagging [1], boosting [2], or Error Correcting Output

* Partial support by the INEEL University Research Consortium project No. C94-175936 to T. Fiez and Z. Obradovic is gratefully acknowledged.

Codes (ECOC) [3] that significantly improve global classifiers like decision trees, rule learners, and neural networks. These algorithms may manipulate the training patterns individual classifiers use (bagging, boosting) or the class labels (ECOC). An ensemble of classifiers must be both diverse and accurate in order to improve accuracy of the whole. Diversity is required to ensure that all the classifiers do not make the same errors. In order to increase the diversity of combined classifiers for spatial heterogeneous databases with attribute instability, one cannot assume that the same set of attributes is appropriate for each single classifier. For each training sample, drawn in a bagging or boosting iteration, a different set of attributes is relevant and therefore the appropriate attribute set should be used by each single classifier in an iteration. In addition, the application of different classifiers on spatial databases, where the data are highly spatially correlated, may produce spatially correlated errors. In such situations the standard combining methods might require different schemes for manipulating the training instances in order to keep the diversity of classifiers.

In this paper, we propose a modification of the AdaBoost algorithm [2] for combining multiple classifiers to improve overall classification accuracy. In each boosting round we try to maximize the local information for a drawn sample by changing attribute representation through attribute selection, attribute extraction and appropriate attribute weighting methods [4]. In order to exploit the spatial data knowledge, a modification of the boosting method appropriate for heterogeneous spatial databases is proposed, where at each boosting round spatial data blocks are drawn instead of the standard approach of sampling single instances.

The influence of these adjustments to single classifiers is not the same for local classifiers (e.g. k–nearest neighbor) and global classifiers (e.g. artificial neural networks). It is known that standard combining methods do not improve simple local classifiers due to correlated predictions across the outputs from multiple combined classifiers [1, 3]. We show that prediction of combined nearest neighbor classifiers can be decorrelated by selecting different attribute representations for each sample and by sampling spatial data blocks. The nearest neighbor classifier is often criticized for slow run-time performance and large memory requirements, and using multiple nearest neighbor classifiers could further worsen the problem. Therefore, we used a novel fast method for k-nearest neighbor classification to speed up the boosting process. We also test the influence of changing attribute representation on global classifiers like neural networks.

2. Related Work

The nearest neighbor classifier [6] is one of the oldest and simplest methods for performing general, non-parametric classification. A common extension is to choose the most common class among the k nearest neighbors. Despite its simplicity, the k-nearest neighbor classifier (k-NN) can often provide similar accuracy to more sophisticated methods such as decision trees or neural networks. It’s advantages include ability to learn from a small set of examples, and to incrementally add new information at runtime.

Recently, researchers have begun testing methods for improving classification accuracy by combining multiple versions of a single classifier, also known as an ensemble approach. Unfortunately, many combining methods do not improve the k-NN classifier. For example, when experimenting with bagging, Breiman [1] found no difference in accuracy between the bagged k-NN classifier and the single model approach. It is also shown that ECOC will not improve classifiers that use local information due to high error correlation [3].

A popular alternative to bagging is boosting, which uses adaptive sampling of patterns to generate the ensemble. In boosting [2], the classifiers in the ensemble are trained serially, with the weights on the training instances set adaptively according to the performance of the previous classifiers. The main idea is that the classification algorithm should concentrate on the difficult instances. Boosting can generate more diverse ensembles than bagging does, due to its ability to manipulate the input distributions. However, it is not clear how one should apply boosting to the k-NN classifier for the following reasons: (1) boosting stops when a classifier obtains 100% accuracy on the training set, but this is always true for the k-NN classifier, (2) increasing the weight on a hard to classify instance does not help to correctly classify that instance as each prototype can only help classify its neighbors, not itself. Freund and Schapire [2] applied a modified version of boosting to the k-NN classifier that worked around these problems by limiting each classifier to a small number of prototypes. However, their goal was not to improve accuracy, but to improve speed while maintaining current performance levels.

Although there is a large body of research on multiple model methods for classification, very little specifically deals with combining k-NN classifiers. Ricci and Aha [5] applied ECOC to the k-NN classifier (NN-ECOC). Normally, applying ECOC to k-NN would not work since the errors in two-class problems would be perfectly correlated. However, they found that applying attribute selection to the two-class problems decorrelated errors if different attributes were selected. Unlike this approach, Bay's Multiple Feature Subsets (MFS) method [6] uses random attributes when combining individual classifiers by simple voting. Each time a pattern is presented for classification, a new random subset of attributes is selected for each classifier.

Although it is known that boosting works well with global classifiers like neural networks, there have been several experiments in selecting different attribute subsets as an attempt to force the classifiers to make different and hopefully uncorrelated errors. Tumer and Ghosh [7] found that with neural networks, selectively removing attributes could decorrelate errors. Unfortunately, the error rates in the individual classifiers increased, and as a result there was little or no improvement in the ensemble. Cherkauer [8] was more successful, and was able to combine neural networks that used different hand selected attributes to achieve human expert level performance in identifying volcanoes from images.

3. Methodology

3.1 Adaptive Boosting for k-NN Classifiers

We follow the generalized procedure of AdaBoost.M2 [2]. The modified algorithm is shown in Fig. 1. It maintains a distribution D_t over the training examples, which can be initially uniform. The algorithm proceeds in a series of T rounds. In each round, the entire weighted training set is given to the weak learner to compute weak hypothesis h_t. The distribution is updated to give wrong classifications higher weights than correct classifications.

- Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, k\}$
- Initialize the distribution D_1 over the examples, such that $D_1(i) = 1/m$.
- For t = 1, 2, 3, 4, … T
  0. Find relevant feature information for distribution D_t
  1. Train weak learner using distribution D_t
  2. Compute weak hypothesis h_t: $X \times Y \rightarrow [0, 1]$
  3. Compute the pseudo-loss of hypothesis h_t:

$$\varepsilon_t = \frac{1}{2} \cdot \sum_{(i,y) \in B} D_t(i, y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$

  4. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$
  5. Update D_t : $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{(1/2) \cdot (1 - h_t(x_i, y_i) + h_t(x_i, y_i))}$
     where Z_t is a normalization constant chosen such that D_{t+1} is a distribution.
- Output the final hypothesis: $h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$

Fig. 1. The scheme of modified AdaBoost.M2 algorithm

Since at each boosting iteration t we have different training samples drawn according to the distribution D_t, at the beginning of the "for loop" in Fig. 1 we modify the standard algorithm by adding **step 0.**, wherein we choose a different attribute representation for each sample. Different attribute representations are realized through attribute selection, attribute extraction and attribute weighting processes through boosting iterations. This is an attempt to force individual classifiers to make different and hopefully uncorrelated errors.

Error correlation is related to Breiman's [1] concept of stability in classifiers. Nearest neighbor classifiers are stable to the patterns, so bagging and boosting generate poor k-NN ensembles. Nearest neighbor classifiers, however, are extremely sensitive to the attributes used. Our approach attempts to use this instability to generate a diverse set of local classifiers with uncorrelated errors. At each boosting round, we perform one of the following methods to determine a suitable attribute space for use in classification.

To eliminate irrelevant and highly correlated attributes, regression-based attribute selection was performed through performance feedback [4] forward selection and backward elimination search techniques based on linear regression mean square error

(MSE) minimization. The *r* most relevant attributes are selected according to the selection criterion at each round of boosting, and are used by the k-NN classifiers.

In contrast to attribute selection where a decision is target-based, variance-based dimensionality reduction through attribute extraction is also considered. Here, linear Principal Components Analysis (PCA) [4] was employed. Each of the k-NN classifiers uses the same number of new transformed attributes. Another possibility is to choose an appropriate number of newly transformed attributes which will retain some predefined part of the variance.

The attribute weighting method used in the proposed method is based on a 1-layer feedforward neural network. First, we try to perform target value prediction for the drawn sample with defined a 1-layer feedforward neural network using all attributes. It turns out that this kind of neural network can discriminate relevant from irrelevant attributes. Therefore, the neural networks interconnection weights are taken as attribute weights for the k-NN classifier.

To further experiment with attribute stability properties, miscellaneous attribute selection algorithms [4] were applied on the entire training set and the most stable attributes were selected. Then the standard boosting method was applied to the k-NN classifiers using the identified fixed set of attributes at each boosting iteration. When boosting is applied with attribute selection at each boosting round, the attribute occurrence frequency is monitored in order to compare the most stable selected attributes. When attribute subsets selected through boosting iterations become stable, this can be an indication to stop the boosting process.

3.2 Spatial Boosting for k-NN Classifiers

Spatial data represent a collection of attributes whose dependence is strongly related to a spatial location where observations close to each other are more likely to be similar than observations widely separated in space. Explanatory attributes, as well as the target attribute in spatial data sets are very often highly spatially correlated. As a consequence, applying different classification techniques on such data is likely to produce errors that are also spatially correlated [12]. Therefore, when applied to spatial data, the boosting method may require different partitioning schemes than simple weighted selection which doesn't take into account the spatial properties of the data. Rather than drawing *n* data points according to the distribution D_t (Fig. 1), the proposed method draws (n/M^2) spatial data blocks (squares of size M points $\times$ M points) according to the new distribution SD_t. The distribution SD_t is modified in such a way that all data points *dp* inside the same spatial block have the same $SD_t(dp)$. This is done through simple median M $\times$ M filtering over the data points inside the spatial block. Using this approach we hope to achieve more decorrelated classifiers whose integration can further improve model generalization capabilities for spatial data.

3.3 Adaptive Attribute and Spatial Boosting for Neural Network Classifiers

Although standard boosting can increase prediction accuracy of artificial neural network classifiers, we experimented with changing attribute representation and

spatial block drawing to see if adaptive attribute and spatial boosting can further improve accuracy of an ensemble of classifiers. The most stable attributes used in standard boosting of k-NN classifiers were also used here for the same purpose. At each boosting round we performed attribute selection and attribute extraction processes, since the attribute weighting method seemed to be "redundant" when training neural network classifiers. We trained multilayer (2-layered) feedforward neural network classification models with the number of hidden neurons equal to the number of input attributes. We also experimented with different numbers of hidden neurons. The neural network classification models had the number of output nodes equal to the number of classes (3 in our experiments), where the predicted class is from the output with largest response. We used two learning algorithms: resilient propagation [10] and Levenberg-Marquardt [11]. The experiments for testing attribute stability through the boosting were repeated as well, and they were used to determine the proper number of boosting iterations.

3.4 The Fast k-NN Algorithm

The quality of k-NN generalization depends on which *k* instances are deemed least distant, which is determined by its distance function. We consider two distance functions in our work: standard Euclidean and Mahalanobis distance.

To speed up the long-lasting boosting process, a fast k-NN classifier is proposed. For *n* training examples and *d* attributes our approach requires preprocessing which takes O($d \cdot n \cdot \log n$) steps to sort each attribute separately. However, this is performed only once, and we trade off this initial time for later speedups.

The main idea of the proposed algorithm will be presented is as follows.

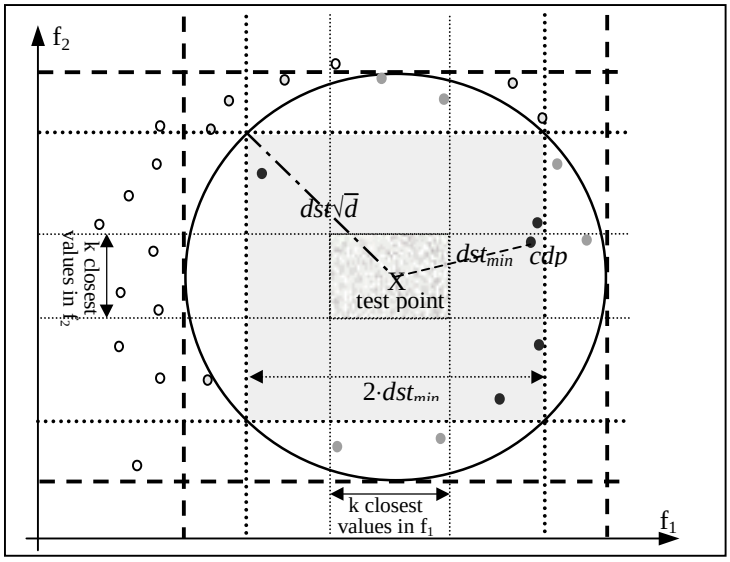

Fig. 2. The used hyper-rectangle, hypersphere and hypercubes in the fast k-NN

Initially, we form a hyper-rectangle with boundaries defined by the extreme values of the *k* closest values for each attribute (Fig. 2 – small dotted lines). If the number of training instances inside the identified hyper-rectangle is less than *k*, we compute the distances from the test point to all of *d·k* data points which correspond to the *k* closest values for each of *d* attributes, and sort them into non-decreasing array *sx*. We take the nearest training example *cdp* with the distance dst_{min}, and form a hypercube with boundaries defined by this minimum distance dst_{min} (Fig. 2 - larger dotted lines). If the hypercube doesn't contain enough (*k*) training points, form the hypercube of a side $2 \cdot sx(k+1)$. Although this hypercube contains more than *k* training examples, we need to find the one which contains the minimal number of training examples greater than *k*. Therefore, if needed, we search for a minimal hypercube by binary halving the index in non-decreasing array *sx*. This can be executed at most *log k* times, since we are reducing the size of the hypercube from $2 \cdot sx(k+1)$ to $2 \cdot sx(1)$. Therefore the total time complexity of our algorithm is $O(d \cdot \log k \cdot \log n)$, under the assumption that $n > d \cdot k$, which is always true in practical problems.

If the number of training instances inside the identified hyper-rectangle is greater than *k*, we also search for a minimal hypercube that contains at least *k* and at most *2·k* training instances inside the hypercube. This was accomplished through binary halving or incrementing the side of a hypercube. After each modification of a hypercube's side, we compute the number of enclosed training instances and modify the hypercube accordingly. By analogous reasoning as in the previous case, it can be shown that binary halving or incrementing the hypercube's side will not take more than *log k* time, and therefore the total time complexity is still $O(d \cdot \log k \cdot \log n)$.

When we find a hypercube which contains the appropriate number of points, it is not necessary that all *k* nearest neighbors are in the hypercube, since some of the closer training instances to the test points could be located in a hypersphere of identified radius $dst\sqrt{d}$ (Fig. 2). Since there is no fast way to compute the number of instances inside the sphere without computing all the distances, we embed the hypersphere in a minimal hypercube and compute the number of the training points inside this surrounding hypercube. The number of points inside the surrounding hypercube is much less than the total number of training instances and therefore classification speedups of our algorithm.

4. Experimental results

Our experiments were performed using spatial data from a 220 ha field located near Pullman, WA. All attributes were interpolated to a 10x10 m grid resulting in 24,598 patterns. The Pullman data set contained x and y coordinates, 19 soil and topographic attributes and the corresponding crop yield. We also performed the experiments on an artificial data set made using our spatial data simulator [13] corresponding to 5 heterogeneous data distributions, each having different relevant attributes for yield generation. The data set had 5 relevant and 5 irrelevant attributes.

For the Pullman data set the miscellaneous attribute selection methods were used to identify the 4 most relevant attributes (Table 1) and the most stable attributes (4, 7, 9, 20) were used for the standard boosting method.

Table 1. Attribute selection methods used to identify 4 most stable attributes

Attribute Selection Methods						
Forward Selection			*Selected Attributes*	*Branch and Bound*		*Selected Attributes*
Inter-class distance	Minkowski order	1	7, 9, 10, 11	Probabilistic distance	Bhatacharya	4, 7, 10, 14
		3	3, 4, 5, 7		Mahalanobis	7, 9, 11, 20
	Euclidean		3, 4, 5, 7		Patrick-Fischer	13,17,20,21
	Chebychev		3, 4, 5, 7			
Probabilistic distance	Bhatacharya		3, 4, 8, 9	*Backward Elimination*		*Selected Attributes*
	Mahalanobis		7, 9, 11, 20			
	Divergence metric		3, 4, 8, 9	Probabilistic distance	Bhatacharya	4, 7, 9, 14
	Patrick-Fischer		13,16,20,21		Mahalanobis	7, 9, 11, 20
Minimal Error Probability, k-NN with Resubstitution			4, 7, 11, 19		Patrick-Fischer	13,17,20,21
Linear Regression Performance Feedback			5, 9, 7, 18	Linear Regression Performance Feedback		7, 9, 11, 20

For the k-NN classifier experiments, the value of *k* was set using cross validation performance estimates on the entire training set. The selected attributes during the boosting iterations were monitored and their frequency was computed. The attribute frequency during the boosting rounds for backward elimination is shown in Fig. 3. PCA was also used to reduce data dimensionality at each boosting round. Here, projections to 4 dimensions explained most of the variance and there was little improvement from additional dimensions. For the attribute weighting method, we used the attribute weights given by a neural network. For each of these attribute representation changing methods, boosting was applied to k-NN classifiers and the classification accuracies for 3 equal size classes are given in Table 2.

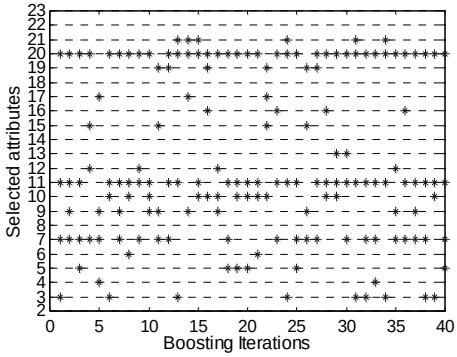

Fig. 3. Attribute stability during boosting on k-NN classifiers

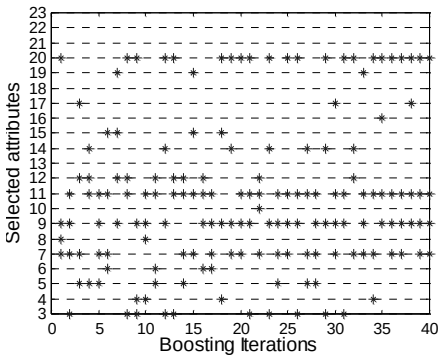

Fig. 4. Attribute stability during boosting on Levenberq-Marquardt algorithm

Table 2. Comparative analysis of overall classification test accuracies for 3-class problems

Number of Boosting Rounds	Standard Boosting on k-NN	Attribute Boosting on k-NN with				Boosting on	
		Forward Selection	Backward Elimination	PCA	Attribute Weighting	Levenberg-Marquardt	
						Standard Boosting	Backward Elimination
8	38.2	40.9	38.5	42.4	43.0	43.6	47.5
16	39.5	41.3	38.8	42.4	43.9	44.1	47.8
24	38.8	41.9	42.1	44.5	44.8	44.8	48.3
32	38.5	41.8	43.5	45.1	46.1	45.5	48.8
40	39.3	42.1	42.8	43.4	44.3	44.9	48.5

Analyzing the data from Table 2, the methods of adaptive attribute boosting outperformed the standard boosting model. The results indicate that 30 boosting rounds were usually sufficient to maximize prediction accuracy. After this many iterations, attribute selection somewhat stabilized although attribute selection during boosting was less stable for k-NN (Fig. 3) than for neural networks (Fig. 4). For k-NN after approximately 30 boosting rounds the attributes became fairly stable with attributes 7, 11 and 20 obviously more stable than attributes 3 and 9 which also appeared in later iterations. The prediction accuracies for k-NN classifier experiments using Mahalanobis distance were worse than those using k-NN classifier with Euclidean distance, and are not reported here. The results show that our approach is promising. For each method of changing attribute representation, we achieve better prediction accuracy than using the standard boosting method.

The frequency of selected attributes during the boosting rounds when boosting was applied to neural network classification models is presented in Fig. 4. The best results were obtained with applied backward elimination attribute selection using the Levenberq-Marquardt algorithm (Table 2). It appears that monitoring selected attributes can be a good criterion for early stopping of boosting, since after the selected attribute subsets become stable no significant improvements in prediction accuracy was noticed. In this case it was even more evident that attributes stabilized after approximately 30 boosting rounds. During the boosting iterations we were selecting the 4 and 5 most important attributes, and the number of hidden neurons in a 2-layer feedforward neural network was equal to the number of input attributes. We noticed that further increasing the number of hidden neurons did not improve prediction accuracy probably because of overfitting.

Since our data have a spatial dimension, we also performed experiments with a modified spatial boosting method. Applying the spatial boosting method to a k-NN classifier, we achieved much better prediction than using the previous two boosting methods on a k-NN classifier (Table 3). Furthermore, when applying spatial boosting with attribute selection at each round, the prediction accuracy was increased slightly as the size (M) of the spatial block was increased (Table 3). No such improvements were noticed for spatial boosting with fixed attributes or with the attribute weighting method, and therefore the classification accuracies for just M = 5 are given.

Applying spatial boosting on neural network classifiers resulted in no enhancements in classification accuracies. Moreover, for pure spatial boosting

without attribute selection we obtained slightly worse classification accuracies than using "non-spatial" boosting. This phenomenon is due to spatial correlation of our attributes, which means that data points close in the attribute space are probably close in real space. Since k-NN examines this local information, it gains from spatial data blocks unlike neural networks which do not consider any kind of spatial information during the training. Therefore, one of our current concerns will be to find a technique to include spatial knowledge into the training of neural networks classifiers.

Table 3. Overall accuracy of spatial boosting on a 3-class real-life test data using k-NN

Number of Boosting Rounds	Spatial Boosting for k-NN with					
	Fixed Attribute Set	Backward Elimination Attribute Selection				Attribute Weighting
	M = 5	M = 2	M = 3	M = 4	M = 5	M = 5
8	46.4	45.8	47.7	48.1	47.8	45.2
16	46.6	46.2	47.6	48.1	47.7	45.6
24	46.7	46.7	47.9	48.2	48.2	45.8
32	46.9	46.9	48.1	48.4	47.9	46.3
40	47.0	47.2	48.1	47.9	47.8	45.9

Although we achieved promising results on the real life data, we repeated all experiments for the more controllable artificial data set, which had 5 clusters similar in attribute space. Each of these clusters had a different set of relevant attributes used for yield generation. The best results for boosting of k-NN and neural network classifiers are shown in Table 4.

Table 4. Comparative analysis of overall classification accuracies for 3-class problems on artificial test data set with 5 clusters (BE stands for Backward Elimination, LM for Levenberg-Marquardt algorithm)

Number of Boosting Rounds	Boosting Applied to k-NN Classifier		Boosting Applied to LM Neural Networks		Spatial Boosting for k-NN with				
					Fixed Attribute Set	Backward Elimination Attribute Selection			
	Standard	BE	Standard	BE	M = 5	M=2	M=3	M=4	M=5
8	57.9	57.5	65.3	66.1	65.6	64.6	65.3	65.4	66.0
16	59.1	59.1	66.7	67.2	65.5	65.2	65.9	65.2	66.7
24	57.6	58.7	67.1	69.3	65.8	65.5	65.9	65.8	67.0
32	58.3	58.5	68.8	69.2	66.0	65.4	66.2	66.1	67.6
40	58.2	59.2	69.8	69.4	66.1	65.3	66.4	66.7	68.1

The adaptive attribute boosting results show no improvements in prediction accuracy, which was due to properties of the artificial data set. Each different region has not only different relevant attributes related to yield class but also a different number of relevant attributes. Since we are not sure of the number of relevant attributes for each region, we need to select at least the 4 or 5 most important attributes at each boosting round. However, the total number of relevant attributes in the data set is 5 as well, and therefore we could not achieve any attribute instability.

To avoid forcing the standard boosting method to be inferior to our method, we used all 5 relevant attributes from the data set for standard boosting. If we select the 5 best attributes during each boosting iteration, it is obvious that we will achieve similar results. Therefore, we were selecting the 4 most relevant attributes knowing that for some drawn samples we would lose beneficial information. Nevertheless, we obtained similar classification accuracies as the standard boosting method, but reached the "bounded" final prediction accuracy in a smaller number of boosting iterations. This could be very important in order to reduce the time needed for the latest boosting rounds. Instead of post-pruning the boosted classifiers [14] we can try to on-line settle the appropriate number of boosting iterations.

Classification accuracies of spatial boosting for k-NN on the artificial data set were again much better than without using spatial information and comparable to neural networks. Here, the classification accuracy improvements from increasing the size (M) of the spatial blocks were more apparent than for real-life data due to the higher spatial correlation of the artificial data.

5. Conclusions and Future Work

Results from two spatial data sets indicate that the proposed algorithm for combining multiple classifiers can result in significantly better predictions over existing classifier ensembles, especially for heterogeneous data sets with attribute instabilities. By manipulating the attribute representation used by individual classifiers at each boosting round, we showed that classifiers could be more decorrelated thus leading to higher prediction accuracy. The attribute stability test served as a good indicator for proper stopping of further boosting iterations. Testing of the proposed method seems to indicate that a smaller number of iterations is needed in order to achieve the same final prediction accuracy. The new boosting method proposed for spatial data showed promising results for k-NN classifiers making it competitive with highly non-linear and powerful models like neural networks.

In this paper, we concentrated on improving the accuracy of the global classifier. Although the new fast k-NN classifier significantly reduces the computational requirements, an open research question is to further increase the speed of ensembles of k-NN classifiers for high-dimensional data.

Although the performed experiments provide evidence that the proposed approach can improve predictions of ensemble of classifiers, further work is needed to examine the method for more heterogeneous data sets with more diverse attributes. In addition, we are working to extend the method to regression based problems.

References

1. Breiman, L.: Bagging predictors, Machine Learning 24, 123-140, (1996)
2. Freund, Y., and Schapire, R. E.: Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, pp. 325-332, (1996)
3. Kong, E. B., Dietterich, T. G.: Error-correcting output coding corrects bias and variance, In Proc. of the twelfth National Conference on Artificial Intelligence, 725-730, (1996)

4. Liu, L. and Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining, Kluwer Academic Publishers, Boston (1998)
5. Ricci, F., and Aha, D. W.: Error-correcting output codes for local learners, In Proceedings of the 10th European Conference on Machine Learning (1998)
6. Bay, S. D.: Nearest Neighbor Classification from Multiple Feature Subsets. Intelligent Data Analysis. 3(3):191-209, (1999)
7. Tumer, K., and Ghosh, J.: Error correlation and error reduction in ensemble classifiers, Connection Science 8, 385-404, (1996)
8. Cherkauer, K. J.: Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In P. Chan, (Ed.): Working Notes of the AAAI Workshop on Integrating Multiple Learned Models, 15-21, (1996)
9. Bishop, C., Neural Networks for Pattern Recognition, Oxford University Press, (1995)
10. Riedmiller, M., Braun, T.: A Direst Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, Proceedings of the IEEE International Conf. on Neural Networks, San Francisco, (1993)
11. Hagan, M., Menhaj, M.B.: Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neura Networks 5 (1994) 989-993
12. Vucetic, S., Fiez, T., Obradovic, Z.: A Data Partitioning Scheme for Spatial Regression, Proceedings of the IEEE/INNS Int'l Conf. on Neural Networks, Washington, D.C., (1999)
13. Pokrajac, D., Fiez, T. and Obradovic, Z.: A Spatial Data Simulator for Agriculture Knowledge Discovery Applications, in preparation.
14. Margineantu, D. D., and Dietterich, T. G.: Pruning adaptive boosting, In Proceedings of the 14th International Conference on Machine Learning, (1997)