Effective Pruning of Neural Network Classifier Ensembles

Aleksandar Lazarevic and Zoran Obradovic

Center for Information Science and Technology, Temple University,
Room 303, Wachman Hall (038-24), 1805 N. Broad St., Philadelphia, PA 19122, USA,
aleks@ist.temple.edu, zoran@ist.temple.edu

Abstract

Neural network ensemble techniques have been shown to be very accurate classification techniques. However, in some real-life applications a number of classifiers required to achieve a reasonable accuracy is enormously large and hence very space consuming. This paper proposes several methods for pruning neural network ensembles. The clustering based approach applies k-means clustering to entire set of classifiers in order to identify the groups of similar classifiers and then eliminates redundant classifiers inside each cluster. Another proposed approach contains the sequence of the depth-first building the tree of classifiers according to their diversity followed by the process of tree pruning. The novel proposed methods applied to several data sets have shown that by selecting an optimal subset of neural network classifiers, it is possible to obtain significantly smaller ensemble of classifiers while achieving the same or even slightly better generalizability as when using the entire ensemble.

1 Purpose

Recently, neural network ensemble techniques have gained widespread interest among researchers in machine learning community. Among many varieties, the most popular include bagging [1], arcing [2] or boosting [3]. The main idea of these techniques is to generate multiple versions of a predictor. When predictions from these versions are combined, more stable and smother predictions are generated. When applied to neural networks, these techniques can yield dramatic improvements in generalization performance [4, 5]. The reason for this is because neural networks are inherently unstable [1, 6], i.e. small changes in training set and/or parameter selection may produce large changes in performance.

The most of combination methods for classifiers assume that the classifiers forming the classifier ensemble have to be both diverse and accurate. The "diversity" assumption means that the classifiers have to make independent classification errors, in order to improve overall prediction accuracy. Both theoretical [7, 8] and empirical work [9, 10] has shown that a good neural network ensemble is one where the individual networks are both accurate and make errors on different parts of the input space. For example, Hansen and Salamon [7] have shown that a multiple classifier system based on a simple majority combination rule can provide very good improvements in accuracy if combined classifiers make independent errors. In addition, Tumer and Ghosh [11] pointed out that the increase of the accuracy depends much more on error uncorrelation than on the particular adopted combination method.

All above mentioned work indicated that the fundamental need for effective combining methods is to design an ensemble of **independent classifiers**. However, most of the previous work has either focused on combining the outputs of multiple trained networks or only indirectly addressed the necessity for generating a good set of networks. In addition, in some real life applications a number of classifiers in an ensemble required to achieve a reasonable accuracy may be enormously large [12, 13].

Margineantu and Dietterich [13] observed that in some domains the boosting algorithm, although effective at generating diverse set of classifiers, requires a large amount of memory for storing all the classifiers in order to lower the prediction error. More specifically, they have noticed that in the *letter* dataset, AdaBoost algorithm [3] requires about 200 C4.5 trees [14] to achieve very good generalization accuracy. They asked if all 200 decision trees are necessary, and according to their findings, by examining the diversity and accuracy of the available classifiers, it is possible to find a subset of classifiers that achieves similar level of performance as the entire set. They proposed an interesting method of pruning the boosting ensemble using a statistics called the Kappa measure [15, 16]. Tamon and Xiang [12] offered a slight modification to the Kappa method, such that the weights of unpruned classifiers are modified with the weights of the pruned ones.

In this paper, a novel method for pruning classifiers from ensembles based on a clustering approach is proposed. For pruning classifiers, a method of distributing their voting

weights is implemented. In addition, a depth-first technique of building a tree with the most "independent" classifiers is proposed. The proposed pruning methods applied to several data sets indicate that by selecting an optimal subset of neural network classifiers, it is possible to obtain the same or even slightly better generalizability as when using the entire ensemble.

2 Method

2.1. Classifier Ensembles

In order to improve the global accuracy, an ensemble of classifiers must be both accurate and diverse. To create an ensemble of classifiers in this study, two approaches are used.

The first one is based on bagging, where a very popular statistical re-sampling technique called bootstrap [17] is used to generate multiple training sets on which the individual networks from an ensemble are constructed. However, the diversity of individual classifiers were tuned not only by varying samples of the training set, but also through different initial weight settings, learning algorithms, number of hidden neurons and number of training epochs.

- Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$
- Let $B = \{(i,y): i = 1,\ldots,m, y \neq y_i\}$
- Initialize the distribution D_1 over the examples, such that $D_1(i) = 1/m$.
- For t = 1, 2, 3, 4, … *T*
  1. Train a weak learner L_t 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 $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2) \cdot (1 - h_t(x_i, y) + h_t(x_i, y_i))$
  5. Update D_t : $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{w_t}$ where Z_t is a normalization constant chosen such that D_{t+1} is a distribution.
- Output the final hypothesis:
  $$h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$$

Figure 1. The AdaBoost.M2 algorithm

A second approach used standard AdaBoost.M2 procedure [3], since the additional randomization techniques used in bagging did not significantly improve the ensemble performance. The AdaBoost.M2 algorithm, shown at Figure 1, proceeds in a series of *T* rounds. In each round, a weak learning algorithm is called and presented with a different distribution D_t that is altered by emphasizing particular training examples. The distribution is updated to give wrong classifications higher weights than correct classifications. The entire weighted training set is given to the weak learner to compute the weak hypothesis h_t. A classifier weight β is computed (for each trial), which is used in the final weighted vote. At the end, all weak hypotheses are combined into a single hypothesis h_{fn}.

2.2. The measures for pruning classifiers

The pruning method is defined as a procedure that takes as input a training set S, the combining method and the set *H* of all obtained classifiers h_t, $t = 1, \ldots, T$. The objective of the pruning method is to identify the minimal subset of classifiers that achieves the best prediction accuracy. In order to maintain the prediction accuracy achieved by an entire ensemble, we need to keep both accurate and diverse classifiers. However, the diversity of classifiers is more important and can be achieved by selecting the classifiers that make different errors, i.e. disagree on different parts on the input space.

As a measure of disagreement between two classifiers, Kappa statistic [15, 16] and correlation error [18] are used. First, the Kappa measure between two classifiers h_i and h_j, where h_i, h_j: $X \rightarrow Y$, is defined. Consider the following $|Y| \times |Y|$ contingency table or matrix $M_{m \times m}$. For elements $a, b \in Y$, define $M_{a,b}$ to contain the number of examples $x \in S$ for which $h_i(x) = a$ and $h_j(x) = b$. If h_i and h_j are identical on the data set *S*, then all non-zero counts will appear along the diagonal. If h_i and h_j are very different, then there should be a large number of counts off the diagonal. Let

$$\Theta_1 = \frac{\sum_{a=1}^{C} M_{a,a}}{m}$$

be the probability that the two classes agree, where *C* is the number of classes, and *m* is the number of examples in the training set *S*. Lets also define

$$\Theta_2 = \sum_{a=1}^{C} (\sum_{b=1}^{C} \frac{M_{a,b}}{m} \cdot \sum_{b=1}^{C} \frac{M_{b,a}}{m})$$

to be the probability that two classifiers agree by chance, given the observed counts in the table. The Kappa measure of disagreement between classifiers h_i and h_j is now defined as:

$$\kappa(h_i, h_j) = \frac{\Theta_1 - \Theta_2}{1 - \Theta_2}.$$

A value of $\kappa = 0$ implies that $\Theta_1 = \Theta_2$, and the two classifiers are considered to be different (or independent). A value of $\kappa = 1$ implies that $\Theta_1 = 1$, which means that the two classifiers agree on every example. It is possible for κ to be negative although it was noted that this rarely occurs [13].

A correlation error [18] has been also used in our approach to analyze and explain the performance and disagreement between two classifiers. For two classifiers h_i and h_j, where h_i, h_j: X → Y, we compute the prediction vectors Y_i and Y_j that classifiers h_i and h_j make respectively. The correlation error between two classifiers h_i and h_j is defined as the correlation coefficient between two vectors Y_i and Y_j.

2.3. The pruning algorithms

Given a set H of all classifiers h_t, $t = 1, \ldots, T$, our goal is to eliminate redundant classifiers that do not affect the improvement in the total prediction accuracy. Denote with $h_i(x)$ the prediction that the classifier h_i makes for the instance $x \in S$. It is apparent that the prediction that the classifier h_i makes for the entire training set S can be represented as a vector Y_i. The vector Y_t contains m classification values, one for each of data examples from the training set S.

Our unsupervised approach aims to identify the groups (clusters) of classifiers that make errors on the same or similar input patterns and then to eliminate the redundant classifiers from these clusters. Consider all T prediction vectors Y_t that T classifiers make. Each of these vectors Y_t may be treated as a data pattern with m attributes. Therefore, a clustering algorithm is applied to the set that contains T patterns, each with m attributes.

In order to partition the set of classifiers into subsets containing similar classifiers, the standard k-means algorithm [19] is employed. Here, data set $P = \{Y_1, \ldots, Y_T\}$, is partitioned into k clusters by finding k points $\{M_j\}_{j=1}^{k}$ such that

$$\frac{1}{n_j} \sum_{Y_i \in Y} (\min_j d^2(Y_i, M_j))$$

is minimized, where $d^2(Y_i, M_j)$ usually denotes the Euclidiean distance between Y_i and M_j, although other distance measures can be used. The points $\{M_j\}_{j=1}^{k}$ are known as *cluster centroids* or cluster *means*.

To identify the optimal number of clusters we gradually increased the number of clusters k until the diversity between the clusters of classifiers starts to deteriorate. The diversity between clusters of classifiers is defined as the measure of disagreement between the cluster *centroids*. When the optimal number k of the clusters are obtained, according to the initial assumption the agreement among the classifiers from the same cluster is large. Therefore, the majority of them can be eliminated according to their accuracy on the validation set or according to their disagreement with the retained ones. The algorithm for pruning the classifiers inside each of the clusters is represented in Figure 2.

The algorithm starts from the least accurate classifier from each cluster and keeps the classifier if its disagreement with the remaining ones overcomes some prespecifed threshold and if it is sufficiently accurate. We were removing the classifiers from the cluster until the prediction accuracy of the entire classifier ensemble on the validation set starts to decrease.

- **For** each of the k clusters
  - **Sort** the classifiers h_t according to the accuracy
  - **Repeat**
    - For *classifier* h_t from (the least accurate) **to** (the most accurate)
    - Compute the measure of disagreement between *classifier* h_t and the most accurate classifier
    - **If** (disagreement greater than prespecified threshold)
      - Remove *classifier* h_t

      else
      - Compute accuracy on the validation set without considering *classifier* h_t
      - **If** (the new computed accuracy > the old accuracy) Remove *classifier* h_t

      end if
  - **Until** (the accuracy increases)

Figure 2. Pruning classifiers inside each cluster

In addition to simple elimination of classifiers inside each cluster, a method of distributing their voting weights is implemented. Each classifier h_t in an ensemble has its weight β_t used in a final weighted vote. In our simple clustering approach, the weights of all removed classifiers are set to zero, since they do not take part in the weighted vote. Instead, it is possible to transfer the voting weights β_t of the pruned classifiers such that each unpruned hypothesis receives a fraction of the weight proportional to its similarity to the pruned hypothesis. Therefore, in this algorithm, each pruned classifier computes the set of distances from itself to the collection of the unpruned classifiers. The pruned classifier then distributes its voting weight using the distribution of distances, after performed normalization. More weight is given to the classifiers that are closer (similar or $\kappa \sim 1$) to the pruned classifier. Our assumption is that the process of distributing the voting weights helps produce a more authentic final ensemble, especially when the pruning rate is high. The algorithm of distributing the voting weights is shown in Figure 3.

- For all classifiers h_t with weights β_t inside a cluster
- If *classifier* h_t is selected for elimination
  - Compute the measure of disagreement d_i between h_t and all other classifiers h_i, $i = 1, \ldots T$.
  - Normalize the measures $d_i = d_i / D_T$, where D_T is a normalization constant such that d_i is a distribution
  - Transfer the $d_i \cdot \beta_t$ part of the voting weight to the unpruned classifier h_i, such that the new weight of the classifier h_i is $\beta_i + d_i \cdot \beta_t$.

Figure 3. Distributing the voting weights

Finally, the method of organizing the classifiers into a tree of classifiers is proposed. The most accurate classifier is identified and the matrix of disagreements among all classifiers obtained through the combining process is constructed. The most accurate classifier is chosen to be the root of the classifier tree (Figure 4). The first level of the classifier tree corresponds to the classifiers that most disagree (they are the least correlated) with the parent classifier, which is the root of the tree (Figure 4). Which classifiers belong to the first level of the classifier tree depends on the prespecified threshold of disagreement among the classifiers. If the disagreement between the parent classifier and the considered classifier prevails the prespecified threshold, the considered classifier is put into the classifier tree as a child node of the parent classifier. The tree nodes of further levels are built in an analogous way considering all remaining classifiers as possible children nodes and each of the classifier from the previous level as parent nodes. It is important to note that using this depth-first building the classifier tree, not all classifiers will necessary be included in the tree.

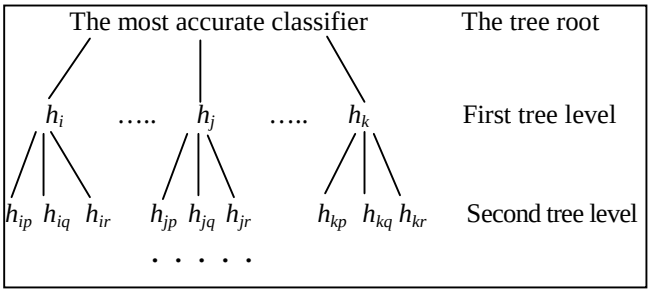

Figure 4. The tree of classifiers

Pruning classifiers was accomplished in the bottom-up manner, starting from the bottom level of the classifier tree and from the least accurate classifiers in that level. The classifiers are removed from the ensemble gradually one by one, only if a new subset of classifiers without removed classifier achieves the prediction accuracy that is not worse than the prediction accuracy of the ensemble containing that classifier.

3 Results

3.1. Experimental Setup

Our experiments were performed on three UCI data sets [20] (diabetes, glass, waveform) and on a synthetic spatial data set. For diabetes data set, the task was to classify from diagnostic data (e.g. blood pressure, result of glucose tolerance test, etc.), whether a female Pima Indian is diabetes positive or not. The data set contains 768 examples with 8 continuous attributes and 2 output classes. The problem may be complex since some values for the input patterns are not available.

The glass data set was from a criminological investigation. In that problem, the glass left at the scene of crime is classified in order to use this as an evidence. The glass data set includes 214 instances with 9 continuous attributes and 6 output classes. For the Waveform set, 10,000 instances with 21 continuous attributes and three equally sized classes are generated.

Finally, a synthetic spatial data set was generated using our spatial data simulator [21] such that the distributions of generated data resembled the distributions of real life spatial data. The obtained spatial data stemmed from a homogeneous distribution and had 6561 data examples with five continuous attributes and three equal size classes.

As classifiers, we trained multilayer (2-layered) feedforward neural network models with the number of hidden neurons equal to the number of input attributes. The neural network classification models had the number of output nodes equal to the number of classes, where the predicted class is from the output with the largest response. We used two learning algorithms: resilient propagation [22] and Levenberg-Marquardt [23]. Non-linear neural network models generally have a large variance, meaning that their accuracy can largely differ over different weight's initial conditions and choice of training data. In such situations using the neural network models may effect in significant errors in the estimation of the level of pruning classifiers. In order to alleviate the effect of neural network instability in our experiments, the prediction accuracy is averaged over 10 trials of considered algorithms for combining classifiers.

3.2. Experimental Results

We first performed *k*-means based clustering approach for pruning. During the identification of the optimal number of clusters, we measured the diversity between the cluster representatives. As cluster representatives we used the classifiers that are the closest to the cluster centroids. When the diversity between cluster representatives started to deteriorate, we assumed that the optimal number of clusters is discovered.

The classifiers inside the obtained clusters are pruned such that the ensemble does not achieve worse prediction accuracy than the entire ensemble of all classifiers. The results of clustering based approach for four different data sets are shown in Table 1.

Analyzing the data from Table 1, it is evident that the clustering based method for pruning classifiers could eliminate 62 – 68% of classifiers without deteriorating the prediction accuracy of the pruned ensemble. For some data sets (diabetes, glass, synthetic), the pruning classifiers could even achieve the negligible improvement in prediction accuracy.

Table 1. The accuracy and pruning level achieved by pruning initial ensemble with T classifiers using the *clustering approach*

Data set →	Diabetes (T = 30)	Glass (T = 40)	Waveform (T = 50)	Synthetic (T = 50)
Accuracy of Entire Ensemble	77.1±0.8	68.6±0.7	86.1 ± 0.6	83.3± 0.8
Accuracy of Pruned Ensemble	77.7±1.3	68.9±0.8	86.0 ± 0.5	83.8± 0.7
Pruning level (%)	63.3	62.5	68	64
Number of clusters	4	4	3	5

The question that naturally arises from the previous method is what happens when we miss the optimal number of clusters. In order to examine this effect, we performed k-means based clustering approach with the different number of discovered clusters. Figure 5 and 6 show how this effect influences both the classification accuracy and the pruning level.

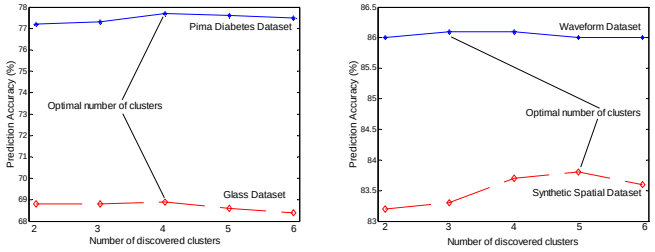

Figure 5. The classification accuracy of pruned neural network ensembles for different numbers of clusters discovered on different datasets.

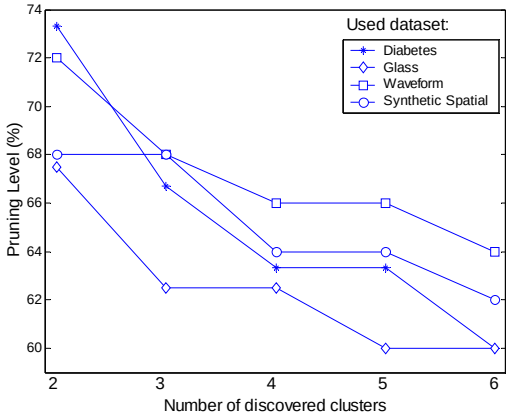

Figure 6. The percentage of eliminated classifiers (pruning level) when different numbers of clusters are discovered on different datasets.

It is evident from Figure 5 that the number of discovered clusters does not significantly influence the classification accuracy achieved by pruned neural network ensembles. However, it is apparent from Figure 6 that there is a trend of decreasing the pruned level when the number of discovered clusters is increasing. By considering both the achieved accuracy and the pruning level, we can identify the optimal number of clusters that gives the best classification for the maximum possible pruning level. Figure 5 shows the optimal number of clusters for different data sets.

When integrating the procedure of distributing the voting weights into the clustering based method of pruning classifiers, we could even achieve slightly better prediction accuracy and pruning level (Table 2).

Table 2. The accuracy and pruning level achieved by pruning initial ensemble with T classifiers using the *clustering approach with distributing the voting weights*

Data set →	Diabetes (T = 30)	Glass (T = 40)	Waveform (T = 50)	Synthetic (T = 50)
Accuracy of Entire Ensemble	77.1±0.8	68.6±0.7	86.1 ± 0.6	83.3± 0.8
Accuracy of Pruned Ensemble	**77.9±0.9**	**69.1±0.8**	**86.3 ± 0.5**	**84.2± 0.9**
Pruning level (%)	66.7	65	74	68
Number of clusters	4	4	3	5

Results from the experiments presented in Table 2 indicate that by using the clustering method with distributing the voting weights the initial ensemble of classifiers can be further pruned with slightly increase in prediction accuracy. Comparing the accuracies and pruning levels in Table 2 and Table 1, it is apparent that consistently better accuracy and larger pruning may be achieved by incorporating the process of distributing the weights.

Finally, the method of building the classifier tree and its pruning is implemented. Experiments on all four considered data sets using this method are presented in Table 3.

Table 3. The accuracy and pruning level achieved by pruning initial ensemble with T classifiers using *the method of building classifier tree*

Data set →	Diabetes (T = 30)	Glass (T = 40)	Waveform (T = 50)	Synthetic (T = 50)
Accuracy of Entire Ensemble	77.1±0.8	68.6±0.7	86.1 ± 0.6	83.3± 0.8
Accuracy of Pruned Ensemble	77.2±1.5	68.6±1.0	86.0 ± 0.7	83.2± 0.9
Pruning level (%)	53.3	55	62	60

Unlike previous clustering based methods, the method of building classifier tree was not successful when pruning classifiers from the initial ensemble. The prediction accuracy achieved by pruning redundant classifiers was comparable to the accuracy of the entire ensemble, while the pruning level was smaller than in clustering based methods.

The results from all experiments performed on all four data set implied that 50 – 70% of constructed classifiers could be effectively pruned without any significant decrease in the prediction accuracy. Using the clustering based methods the accuracy of pruned ensembles for some data sets (diabetes, glass, synthetic) can be even improved when comparing to the accuracy of the entire ensemble. The best performance was achieved by the clustering method with distributing the voting weights.

4 Conclusions

Several new methods for pruning neural network classifier ensembles using unsupervised clustering algorithm and breadth-first building the classifier tree are proposed. Experimental results on several data sets indicate that the proposed techniques for pruning classifiers can effectively achieve similar or even better prediction accuracy while requiring for 50 – 70 % less number of classifiers.

The proposed framework for pruning classifier is applicable to both centralized and distributed environment where large number of classifiers may be present. Our future work will include the methods of post-pruning the classifiers in distributed environments, where this step may be necessary in order to increase system throughput.

In addition, alternate clustering algorithms will be considered where the optimal number of clusters may be identified by the clustering algorithm itself. Finally, we are working to extend the proposed methods to regression-based problems.

Acknowledgments. Work in part supported by INEEL LDRD Program under DOE Idaho Operations Office Contract DE-AC07-99ID13727.

5 References

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

[2] Breiman, L., "Bias, Variance and Arcing Classifiers", Technical Report TR-460, Department of Statistics, University of California, Berkley, 1996.

[3] Freund, Y., and Schapire, R. E., "Experiments with a new boosting algorithm", *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 325-332, 1996.

[4] Carney, J. G., Cunningham, P., "The NeuralBAG Algorithm: Optimizing Generalization Performance in Bagged Neural Networks", in Verleysen, M. (Ed.), *Proceedings of the 7th European Symposium on Artificial Neural Networks*, pp. 35-40, 1999.

[5] Drucker, H., Schapire, R. and Simard, P., "Improving Performance in Neural Networks Using a Boosting Algorithm", in Hanson, S. J., Cowen, J. D., Giles, C. L. (Eds.), *Advances in Neural Information Processing Systems*, Morgan Kaufman, 5, pp. 42-49, 1993.

[6] Breiman, L., "Heuristic of instability in model selection", Technical Report, Statistics Department, University of California at Berkley, 1996.

[7] Hansen, L., and Salamon, P., "Neural Network Ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, pp. 993-1001, 1990.

[8] Krogh, A., Vedelsby, J., "Neural Network Ensembles, Cross Validation and Active Learning", in Tesauro, G., Touretzky, D., and Leen, T., (Eds.), *Advances in Neural Information Processing Systems*, vol. 7, MIT Press, 1995.

[9] Hashem, S., Schmeiser, B. and Yih, Y., "Optimal Linear Combination of Neural Networks: An Overview", *Proceedings of the IEEE International Conference on Neural Networks*, 1994.

[10] Maclin, R. and Shavlik, J., "Combining the Predictions of Multiple Classifiers: Using Competitive Learning to Initialize Neural Networks", *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995.

[11] Tumer, K., and Ghosh, J., "Error Correlation and Error Reduction in Ensemble Classifiers", *Connection Science* 8, pp. 385-404, 1996.

[12] Tamon, C. and Xiang J., "On the Boosting Pruning problem", *Proceeding of the 11th European Conference on Machine Learning*, pp. 404-412, 2000.

[13] Margineantu, D. D., and Dietterich, T. G., "Pruning Adaptive Boosting", *Proceedings of the 14th International Conference on Machine Learning*, pp. 211-218, 1997.

[14] Quinlan, J. R., *C4.5 Programs for Empirical Learning*, Morgan Kaufman, San Francisco, 1993.

[15] Agresti, A., *Categorical Data Analysis*, John Wiley and Sons, Inc., 1990.

[16] Cohen, J., "A coefficient of Agreement for Nominal Scales", *Educational and Psychological Measures*, 20, pp. 37-46, 1960.

[17] Efron, B., Tibshirani, R., *An Introduction to the Bootstrap*, Chapman and Hall, London, 1993.

[18] Ali, K., and Pazzani, M., "Error reduction trough learning multiple descriptions", *Machine Learning*, 24, pp. 173-202, 1996.

[19] Kaufman, L., Rousseeuw, P., *Finding groups in data: an introduction to cluster analysis*, Willey, New York, 1990.

[20] Blake, C.L. & Merz, C.J., UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[21] Pokrajac D, Fiez T, Obradovic Z., "A Spatial Data Simulator for Agriculture Knowledge Discovery Applications", in review.

[22] Riedmiller, M., Braun, H., "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm", *Proceedings of the IEEE International Conference on Neural Networks*, pp. 586–591, 1993.

[23] Hagan, M., Menhaj, M.B., "Training feedforward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks* 5, pp. 989-993, 1994.