01 Jan 2003

Combining Evolving Neural Network Classifiers Using Bagging

Sunghwan Sohn

Cihan H. Dagli

Missouri University of Science and Technology, dagli@mst.edu

Recommended Citation

S. Sohn and C. H. Dagli, "Combining Evolving Neural Network Classifiers Using Bagging," *Proceedings of the International Joint Conference on Neural Networks, 2003*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2003.

The definitive version is available at https://doi.org/10.1109/IJCNN.2003.1224088

Combining Evolving Neural Network Classifiers Using Bagging

Sunghwan Sohn and Cihan H. Dagli
Smart Engineering Systems Lab
Department of Engineering Management
University of Missouri-Rolla
Rolla, MO 65409
ssohn@umr.edu, dagli@umr.edu

Abstract-**The performance of the neural network classifier significantly depends on its architecture and generalization. It is usual to find the proper architecture by trial and error. This is time consuming and may not always find the optimal network. For this reason, we apply genetic algorithms to the automatic generation of neural networks. Many researchers have provided that combining multiple classifiers improves generalization. One of the most effective combining methods is bagging. In bagging, training sets are selected by resampling from the original training set and classifiers trained with these sets are combined by voting. We implement the bagging technique into the training of evolving neural network classifiers to improve generalization.**

I. INTRODUCTION

The complex structure of neural networks (NNs) provides the powerful performance to solve difficult problems. However, this complexity causes the difficulty of how to construct the proper architecture. Generally, this architecture is implemented by trial and error. This approach may not yield the optimal network. Genetic algorithms (GAs), which are the biologically inspired optimization algorithm, could be a good option to search the optimal network architecture [1]. Here, the GA can search the space of the network architecture globally for a given requirement such as minimum error or low complexity [2].

Previous researches have shown that combining multiple unstable classifiers, such as neural networks, reduces classification error. Unstable classifiers can have universally low bias but have high variance [3]. Combining multiple classifiers can reduce a variance. This approach is to use diverse information that allows a more robust decision rather than relying on single classifier. Combining classifiers has the potential to offer complementary information because the samples misclassified by different classifiers might not overlap. One effective combining method is bagging [4].

In this paper, the evolving neural network (ENN), which is the combination of NNs and GAs, is proposed. The GA is used to find the proper network architecture as well as the proper feature subset for a given problem [5]. Also, the bagging technique is applied to combining evolving neural networks.

Section II explains the brief literature reviews about how to combine NNs and GAs. Section III introduces several methods of combining classifiers. Then, it explains the bagging technique and discusses the proper size of combining classifiers. Section IV describes the proposed method of constructing evolving multiple neural networks utilizing bagging. Section V provides experimental results of using the proposed network. Finally, section VI concludes with a summary of this paper.

II. COMBINING NNs AND GAs

A genetic algorithm has been adapted to designing a neural network in several approaches. These methods are briefly discussed in the following:

Feature Selection: GAs can be used to select the proper feature sets in pattern recognition problems. It was used to reduce the dimensionality of feature set for a *K*-nearest neighbor classifier in a speech recognition task [6]. It was also used to find scaling factors for each feature to improve the performance of a *K*-nearest neighbor classifier [7].

Neural Network Training: GAs have been used to train the weights of neural networks and to work as a learning algorithm [8], [2], [9]. It performs a global search of the weight space. The GA can be used even though error-gradient information is not available or is computationally expensive. However, the GA is very slow in fine tuning of weights to a solution.

Neural Network Architecture: This is the most interesting topic in designing NNs. The topology of a NN can be optimized by using GAs [10], [11], [12]. During the evaluation an individual is translated into a network architecture. Usually, this network is trained using a separate training method such as backpropagation. After training a network, the fitness measure is evaluated for the network performance. This fitness measure can be the error performance on the training data. However, it often includes other factors such as network size or complexity.

Neural Network Parameters: The GA is sometimes used to determine initial weights for backpropagation [13]. Also, it was used to evolve centers and widths for a radial basis function network [14].

In this paper, the GA is used in designing a NN for both adaptable feature selection and evolving architecture. The details will be described in section IV.

0-7803-7898-9/03/$17.00 ©2003 IEEE

III. Combining Classifiers

Previous studies have shown that the combined classifiers could produce more reliable decisions than the single classifier alone. This has been shown in various methods such as a majority vote, average, linear combination methods, etc. Kittler et al. [15] provided a theoretical framework of above combining methods. Some researchers focus on training a portion of the training set to produce classifiers having different prediction. Schapire [16] introduced boosting that is rooted in a distribution-free or probably approximately correct (PAC) model of learning. Freund and Schapire [17], [18] proposed an algorithm, which is to adaptively resample so that the weights in the resampling are increased for hard samples and combine by weighted voting. Breiman [19] proposed bagging that selects training sets by resampling and each classifier trained with these sets are combined by voting. In this paper, we utilize bagging to have diverse evolving neural network classifiers and so improve generalization. The next section briefly explains the bagging classifier.

A. *Bagging Classifiers*

Bagging [19] is a combining method that produces members for its ensemble by training each classifier on a random redistribution of training data. Each classifier has the training set generated by different random sampling with replacement, where the size of the training set is same as the original. Some of the original samples may be repeated in the training set while others may not be selected. Therefore, an individual classifier could have a higher test error. However, combining these classifiers can produce a lower test error than that of the single classifier because the diversity among these classifiers generally compensates for errors of any individual classifier [20].

Breiman [19] pointed out that bagging is effective on unstable learning algorithms, such as neural networks, in that small changes in the training set could cause large changes in the resulting predictors.

B. *Size of Combining Classifiers*

Hansen and Salamon [21] suggested that combining with as few as ten individuals was adequate to reduce error. Recent work in boosting and bagging suggested that further improvement is possible even after ten individuals have been added to the ensemble [22]. Opitz and Maclin [20] empirically demonstrated that much of the error reduction occurred after ten to fifteen classifiers when using bagging and boosting in neural networks. Also, they showed that the error reduction nearly did not appear after 25 classifiers. For this reason, we also perform the experiment to select proper individual classifiers to reduce the size of the ensemble.

IV. Combining Evolving Neural Networks

The proposed evolving neural networks have the ability to select proper feature subsets and to evolve neuron-connection links. This ability enables the network to properly adjust a given problem.

A. *Designing Evolving Neural Networks*

Fig. 1 shows the basic steps of constructing evolving neural networks. The individual of the population is translated into a network structure and then trained by a separate training module such as backpropagation. The feature selection and neuron connectivity between a hidden layer and an output layer are implemented by the binary genetic algorithm. After training a network, the fitness measure is evaluated for the network performance.

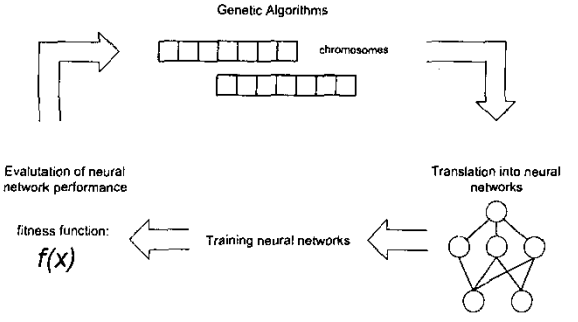

Fig. 1. Optimizing a NN architecture using GA

The issues for implementing GAs are explained in the following:

Encoding

The NN structure has to be properly translated into a chromosome for the effective evolution. Here, a binary encoding is used because it is simple and proper to express feature selection and neuron connectivity. The chromosome is composed of feature selection and neuron connectivity parts. In the feature selection part, "1" means a selected feature and "0" means an unselected feature. In the neuron connectivity part, "1" means an existing connection link and "0" means no connection link between neurons. Fig. 2 is an example of encoding neural networks having one hidden layer.

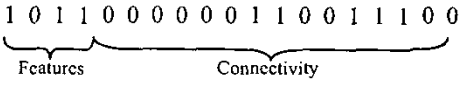

Fig. 2. The NN encoding into a chromosome

The selected features (input neurons) and the hidden neurons are fully connected. The neuron connectivity defines weight connections between a hidden layer and an output layer. Fig. 3 shows the corresponding architecture of evolving neural networks generated by Fig. 2.

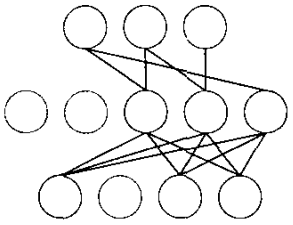

Fig. 3. The ENN generated by Fig. 2

Genetic Operators

The idea of GAs is essentially from Darwinian natural selection. Selection provides the driving pressure in GAs. The tournament selection is used in this experiment. This selection randomly chooses a set of chromosomes and picks out the best chromosome based on the fitness value for reproduction.

Crossover is used to generate a new population of NNs. The architecture of two NNs is exchanged by crossover to search the optimum architecture of NNs. Here, two-point crossover is used.

After generating new offspring, mutation is performed on the selected chromosomes. Each gene in every chromosome has a chance to mutate by a given mutation rate. Mutation changes the element value to a new one. Thus, mutation serves to toggle feature selection or neuron connectivity in the architecture of NNs.

Fitness Function

The fitness measure is used to select proper networks for a given problem. The used fitness function is based on the classification performance and network complexity (1).

$$FV = \alpha(CR_v) + \beta\left(1 - \frac{C}{C_{\max}}\right) \quad \alpha, \beta > 0, \ \alpha+\beta=1. \tag{1}$$

where α and β are weight constants for the performance and complexity respectively

CR_v is the correct classification ratio

C is the complexity defined by the number of connections used between a hidden layer and an output layer

C_{max} is the maximum complexity defined by the number of full connections between a hidden layer and an output layer

This fitness value has range [0,1] and the larger value denotes the better fitness. The GA operates to maximize the fitness value so that it maximizes the correct ratio and minimizes the complexity.

B. Combining Evolving Neural Networks By Bagging

In combining classifiers, accuracy and diversity of the individual classifier should be considered to ensure the good performance. In other words, each individual classifier should have a certain level of accuracy and also it should produce errors on different parts of the input set to compensate for another's errors. In this paper, the GA is used to find the proper network architecture so that the individual classifier can produce high accuracy. Bagging is applied to provide diversity and so improve generalization in combining classifier. Fig. 4 shows the overview of this process.

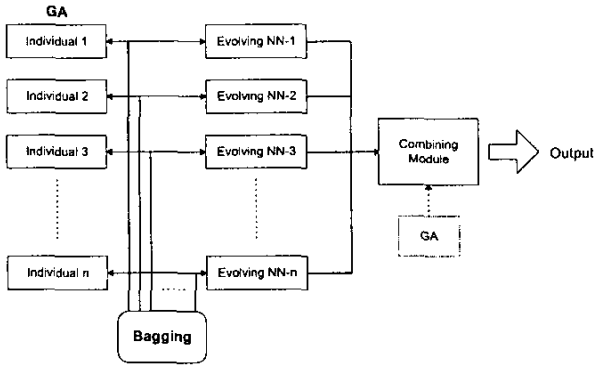

Fig. 4. Combining ENNs by bagging

Each individual classifier is implemented by designing functionally independent networks using the GA – i.e., each individual in the population of the GA is translated into the sub-classifier in the ensemble. This network architecture found by the GA can provide high accuracy for a given data. Here, the individual classifiers are trained by the different training sets that are selected by bagging. Then, they are combined by voting for the final decision. However, some individual classifiers in the population might not significantly contribute to the final decision. Therefore, the properly selected individual classifiers might have enough information for the robust final decision. For this reason, we also examine the performance of combining selected classifiers. Another GA is used to choose these proper classifiers. Here, the mean squared error is used as the fitness function.

V. EXPERIMENTS

Seven data sets from UCI machine learning repository were used to demonstrate the performance of the proposed

networks. The results of classical neural networks and best neural networks were presented for comparison. The best neural network is a single best individual network in the GA population without using bagging technique. Also, the performance of simple combining methods was compared with the proposed method. TABLE 1 shows the summary of data sets used in this experiment.

TABLE 1
DATA SETS

Data	Features	Samples	Classes
Hepatitis	19	155	2
Liver Disorders	6	345	2
Diabetes	8	768	2
Austrian Credit	15	653*	2
German Credit	20	1000	2
Vehicle	18	846	4
Glass	10	214	6

* Samples having missing values are removed

In the GA process of implementing evolving NNs, the mutation rate was 0.01, the double crossover rate was 0.8, and the elitist strategy was used. Backpropagation with one hidden layer was used. In evolving NNs using bagging, each data set was divided into two parts – training data and test data (80% and 20%, respectively). The training data was resampled and fed into the individual network. Thus, each individual network has a different training set. This resampled training data was used to calculate the fitness value (1) (α=0.99 and β=0.01) of individual networks and the original training data was used to calculate the fitness value in selective combining. The test data was used to provide the classification performance. For each experiment 5-fold cross validation was used to have more robust results. The following TABLE 2 presents the classification results.

TABLE 2
CLASSIFICATION ERROR (%)
(SINGLE vs. COMBINING)

Data	Simple NN	Best ENN	Bag ENN	SelBag ENN
Hepatitis	13.55	12.26	10.32	10.32
Liver	33.91	30.14	29.57	26.67
Diabetes	23.94	23.15	22.76	22.12
Aus credit	14.56	14.41	13.63	12.87
Ger credit	26.00	24.60	23.60	23.30
Vehicle	21.87	21.04	19.51	18.68
Glass	38.07	32.40	30.62	29.19

In TABLE 2, Simple NN is the classical neural network with fixed architecture, Best ENN is the best individual network in the population without using bagging, Bag ENN is the combining all evolving neural networks using bagging, and SelBag ENN is the selectively combining evolving neural networks using bagging.

For all data sets, Best ENN had lower error than that of Simple NN. In fact, the network topology of evolving neural networks was simpler than that of classical networks – i.e., less number of features, less number of hidden nodes, and less complexity. This partial connected network seemed to improve generalization and produce better performance.

When using the combining networks, we could further improve performance. As can be seen in TABLE 2, both Bag ENN and SelBag ENN produced consistently better classification performance than those of Simple NN and Best ENN. In most cases, SelBag ENN had lower error than that of Bag ENN. The number of individual networks used in Bag ENN was from 30 to 80 depending on the data set. However, the number of networks used in SelBag ENN was less than a half of Bag ENN – approximately from 10 to 25 depending on the data set. In section III, we already discussed the size of combining classifiers in bagging. This experiment also supports that the part of individual networks is adequate to sufficiently reduce error.

In TABLE 3, we compare combining evolving neural networks with simple combining methods such as averaging and bagging, which do not employ evolving process. In simple combining, we used the same size of ensemble as in Bag ENN. Simple bagging had a slightly better performance than simple averaging. Combining evolving neural networks produced lower error than those of both simple averaging and bagging.

TABLE 3
CLASSIFICATION ERROR (%)
(SIMPLE vs. EVOLVING COMBINING)

Data	Simple Avg	Simple Bag	Bag ENN	SelBag ENN
Hepatitis	11.61	12.26	10.32	10.32
Liver	32.17	30.43	29.57	26.67
Diabetes	23.41	23.15	22.76	22.12
Aus credit	13.95	13.64	13.63	12.87
Ger credit	24.50	23.80	23.60	23.30
Vehicle	19.86	19.62	19.51	18.68
Glass	35.34	32.05	30.62	29.19

VI. CONCLUSIONS

This paper proposes combining evolving neural networks that uses bagging to improve generalization. Here, the final decision is made by combining individual networks – i.e., voting the results of individual networks formed by the GA. The proposed evolving network properly selects features and adjusts its architecture so as to effectively fit into a given problem. This problem might be minimum classification error, low complexity, or some special task. It can be defined by the fitness function. The bagging technique in training networks increases the diversity of the ensemble and so helps to improve generalization for the test set. Also, the experimental result shows that the GA can effectively select

proper individual networks in combining without loss of classification performance.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, Michigan, 1975.

[2] X. Yao and Y. Liu, "A New evolutionary system for evolving artificial neural networks," *IEEE Trans. on Neural Networks*, vol. 8, no. 3, pp. 694-713, 1997.

[3] L. Breiman, "Bias, variance, and arching Classifiers," *Technical Report 460, University of California – Berkeley*, Berkeley, CA, 1996.

[4] L. Breiman, "Bagging predictors," *Machine Learning*, 24 (2), pp. 123-140, 1996.

[5] S. Sohn and C. H. Dagli, "Adaptable multiple neural networks using evolutionary computation," *Proc. of SPIE –Applications and Science of Computational Intelligence V*, vol. 4739, pp. 141-149, 2002.

[6] E. J. Chang and R. P. Lipmann, "Using genetic algorithms to improve pattern classification performance," *Proc. of Neural Information Processing Systems*, pp. 797-803, 1991.

[7] J. D. Kelly and L. Davis, "Hybridizing the genetic algorithm and the K-nearest neighbors classification algorithm," *Proc. of Fourth International Conference on Genetic Algorithms*, pp. 377-383, 1991.

[8] M. N. H. Siddique and M. O. Tokhi, "Training Neural Networks: Backpropagation vs Genetic Algorithms," *Proc. IEEE International Joint Conference on Neural Networks*, vol. 4, pp.2673-2678, 2001.

[9] P. W. Munro, "Genetic Search for Optimal Representations in Neural Networks," *International Conference on Artificial Neural Nets and Genetic Algorithms (ANNGA93)*, Innsbruck, Austria, pp. 628-634, 1993.

[10] K. Chellapilla and D.B. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Trans. on Evolutionary Computation*, vol. 5, Issue 4, pp. 422-428, 2001.

[11] G. G. Yen and H. Lu, "Hierarchical genetic algorithm based neural network design," *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pp. 168-175, 2000.

[12] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of Art," *International Workshop on Combinations of Genetic Algorithms and Neural Networks*, Baltimore, MD, pp. 1-37, 1992.

[13] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: using genetic algorithms with connectionist learning," *University of California at San Diego CSE Technical Report CS90-174*, 1990.

[14] A. W. Bruce and D. C. Timothy, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Trans. on Neural Networks*, vol. 7(4), pp. 869-880, 1996.

[15] J. Kittler, M. Hatef, R. P.W. Duin, and J. Matas, "On Combining Classifiers," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 3, 1998.

[16] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197-227, 1990. .

[17] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 148-156, Bari, Italy. 1996.

[18] Y. Freund and R. E. Schapire, "Game theory, On-line prediction and boosting," *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pp. 325-332, Desenzano del Garda, Italy, 1996.

[19] L. Breiman, "Stacked regressions," *Machine Learning*, 24 (1), pp. 49-64, 1996.

[20] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research*, 11, pp. 169-198, 1999.

[21] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12, pp. 993-1001, 1990.

[22] R. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Proceeding of the Fourteenth International Conference of Machine Learning*, pp. 322-330, Nashville, TN, 1997.