Coupling Weight Elimination and Genetic Algorithms

George Bebis, Michael Georgiopoulos, and Takis Kasparis

Department of Electrical & Computer Engineering
University of Central Florida
Orlando, FL 32816

Abstract

Network size plays an important role in the generalization performance of a network. A number of approaches which try to determine an "appropriate" network size for a given problem have been developed during the last few years. Although it is usually demonstrated that such approaches are capable of finding small size networks that solve the problem at hand, it is quite remarkable that the generalization capabilities of these networks have not been thoroughly explored. In this paper, we have considered the weight elimination technique and we propose a scheme where it is coupled with genetic algorithms. Our objective is not only to find smaller size networks that solve the problem at hand, by pruning larger size networks, but also to improve generalization. The innovation of our work relies on a fitness function which uses an adaptive parameter to encourage the reproduction of networks having good generalization performance and a relatively small size.

1. Introduction

A lot of emphasis has been given in the last few years in the development of techniques which try to improve generalization by modifying the network structure during training. However, in most cases, the feasibility of an approach is illustrated showing results on network size reduction and convergence speed, while less or no emphasis has been given to the generalization issue [1]-[5]. In fact, in some studies where generalization was addressed, improvements were observed using artificial data sets only [6]-[8], while no significant improvement, or even worse, generalization has been reported in some other studies where real data sets were used [9]-[12]. Our interest in this paper is to improve the performance of weight pruning techniques. Weight pruning techniques are very sensitive to the selection of certain parameter values which determine when pruning should start and when it should stop. If pruning starts too early, the network might not be able to learn the desired mapping. On the other hand, if pruning stops early, it might not be possible to sufficiently prune the network. Also, if pruning does not stop at the right point, it might be possible to overprune and this will deteriorate generalization. Determining appropriate pruning parameter values to control the beginning and the end of the pruning process is usually done by trial and error.

In this paper, we propose the coupling of genetic algorithms [13] and weight pruning. In particular, the weight elimination technique [14],[15] a representative weight pruning technique and the most general of weight decay approaches has been chosen to be coupled with the genetic algorithm. However, the framework of the approach we propose is more general and other pruning techniques can also be coupled with genetic algorithms. The goal of the proposed approach is to prune oversized networks with the objective that the obtained pruned networks will always have a small size and a better generalization performance than that of their unpruned counterparts. Our primary interest is not only to determine an appropriate network size by pruning oversized networks but also to demonstrate that the pruned networks improve generalization.

Choosing a "good" fitness function is probably the most critical issue in genetic algorithm design since it provides the mechanism for evaluating the members (encoded solutions) of a population. The innovation of the proposed approach relies on the use of a fitness function which takes into consideration both network size and generalization. In particular, during the generation of new genetic populations, an adaptive parameter weights the importance of network size versus generalization, encouraging the reproduction of networks having good generalization and a relatively small size. The motivation for using genetic algorithms is that as new populations are formed, the network structure of each member in a population changes in a different way since each network is associated with different parameter values. Since the characteristics among different members in the population can be exchanged by applying the genetic operators, new, more powerful members may be discovered as evolution proceeds.

Acknowledgment: This research was supported in part by a grant from Harris Corporation.

The organization of the paper is as follows: Section 2 reviews the weight elimination technique and illustrates its sensitivity to the selection of certain parameter values which affect network size and generalization. Section 3 discusses the genetic algorithm approach. The databases used, the experiments performed, and the results obtained are presented in Section 4. Finally, our conclusions are given in Section 5.

2. Reducing network size using weight elimination

Weight elimination is a general weight decay approach proposed by Weigend et. al. [14],[15]. It minimizes a modified error function which is formed by adding a penalty term to the original error function of the back-propagation algorithm. Specifically, the modified error function has the form:

$$E_{WE} = E_0 + \lambda_{WE} E_1 = \sum_k (t_k - o_k)^2 + \lambda_{WE} \sum_{i,j} \frac{w_{ij}^2/w_0^2}{1 + w_{ij}^2/w_0^2} \quad (1)$$

There are several issues to be addressed during the implementation of the weight elimination technique. The first and probably most important issue is deciding when pruning should start affecting training. This issue is straightforward related to the choice of the weight factor λ_{WE}. In general, this parameter is chosen by trial and error. However, experience has shown that better results can be achieved if this parameter can be determined adaptively during training. Weigend et. al. [14] have proposed a procedure for this. Initially, λ_{WE} is set to zero. Then, it increases, decreases, or stays the same according to a methodology based on a number of parameters such as the error of the network, the average error, and a desired error provided externally by the user. The amount by which λ_{WE} must be increased or decreased is another parameter specified by the user. We have found that this procedure is not very robust. Here, we have chosen an alternative way for determining λ_{WE}, motivated by [6]. Specifically, λ_{WE} is determined as follows:

$$\lambda_{WE} = \lambda_{0\_WE} e^{-\beta_{WE} E_{gen}} \quad (2)$$

where λ_{0_WE} and β_{WE} are constants to be defined. E_{gen} is an estimation of the generalization error of the network. When E_{gen} is large, λ_{WE} will be small and the second term in Eq. (1) will contribute almost nothing to the total error. When E_{gen} starts decreasing, the second term will start being more significant driving small weights to zero. A common way of estimating the generalization performance of a network during training, is by using cross-validation [16]. If G_{val} represents the ratio of the correctly classified patterns from the validation set over the total number of patterns in the validation set, then we define E_{gen} as $E_{gen} = 1 - G_{val}$. Updating G_{val} and λ_{WE} takes place in every epoch.

Another important issue to be addressed is when training, and consequently pruning, should stop. It has been reported that the exact stopping point is not very important [15]. However, our experimental results shows that this is not consistently true since it might result in overpruning the network, which can deteriorate generalization significantly. It is also important to decide when is appropriate to remove connections associated with small weight values. In our implementation, connections are removed at the end of the training process. A weight w_{ij} is removed only if $|w_{ij}| < |w_0|$.

2.1. Sensitivity of weight elimination.

In this section, we consider the sensitivity of weight elimination on the selection of various parameter values. Four different databases have been used in the experiments reported in this subsection: the Numbers, the Ionosphere, the Wine, and the Breast-cancer databases. Details about each of the databases are given in section 4. First, we examined the sensitivity of weight elimination on the selection of the β_{WE} parameter. Thus, we fixed w_0 (w_0=0.25) and the initial weights and we performed 20 different experiments using different β_{WE} values each time (from β_{WE}=10 to β_{WE}=100 in increments of 5). From the results it was obvious that different b_{WE}'s are more appropriate for different databases, making the choice of appropriate β_{WE} values problem dependent. For the Numbers database, the best β_{WE}'s seems to be in the range of [20-30], for the Ionosphere database in the range of [15-50], for the Wine database in the range of [25-35] and for the Breast-cancer database in the range of [30-100]. Next, we tested the dependence of weight elimination on the selection of the w_0 parameter. For this reason, we fixed β_{WE} (β_{WE}=30) and the initial weights and we performed 20 different experiments using different w_0 values each time (from w_0=0.1 to w_0=1.0 in increments of 0.05). The results demonstrated that the choice of w_0's value is not quite problem dependent and that choosing w_0 close to 1.0 gives good results both in terms of network size and generalization. In the last set of the experiments, we tested the sensitivity of weight elimination on the selection of different initial weights. This time, we fixed β_{WE} and w_0 (β_{WE}=30 and w_0=0.25) and we performed 20 different experiments using different random initial weights each time (randomly chosen in the range of [-0.1 - 0.1]). The results showed that the choice of the initial weights is very critical since both generalization and network size are affected in an unpredictable way.

3. The genetic algorithm approach

As we show in the previous section, weight elimination is quite sensitive to the selection of certain parameters and substantial differences in terms of generalization and network size can be observed by changing these parameter values. Finding good parameter values requires an extensive experimentation. Here, we propose not to perform each experiment independently, but to form a population of networks having the same parameter settings with these chosen for the individual experiments and apply genetic algorithms. Initially, we start with two-layer networks (i.e., one hidden and one output layer), having enough nodes in the hidden layer to ensure convergence. After the network has been chosen, we encode it into a structure that can be handled by the genetic algorithm and we create P copies of it, where P represents the population size. Each of these copies is assigned a different set of parameter values. The parameters were chosen to be the initial weights and β_{WE}. This choice was based on the experimental results presented in section 2.1 which indicated that these parameters affect generalization and network size the most. New populations are generated by applying the genetic operators of reproduction, crossover and mutation.

The fitness of each member is measured by first decoding it into a network. Then, we train it for a number of epochs using weight elimination in order to record the network's performance in terms of generalization and size. The evaluation function consists of two terms: the first term returns an evaluation with respect to the generalization performance of the network while the second term returns an evaluation with respect to it's size. A factor λ_{WE} weights the importance of generalization versus network size. Each network in the population is associated with it's own λ_{WE} parameter. After the first few generations, members in the same population will have totally different characteristics. This variety in network sizes and generalization performances will allow the genetic algorithm to search and probably discover better solutions. Next, we describe the network representation scheme, the genetic operators, and the fitness evaluation function used.

3.1. Network representation scheme

Since our approach considers a large predefined, two-layer network with the objective to reduce the number of connections, we do not really need to encode the number of layers and the number of nodes per layer. The quantity which is important to encode is the number of connections between successive layers. Here, we have adopted the approach proposed by Montana and Davis [17]. According to this approach, the weights and biases of a network are encoded in a straightforward way as a string of real numbers. Decoding is also straightforward.

3.2. Genetic operators

The genetic operators used in this work are the most commonly used operators: the reproduction, the crossover, and the mutation operators. The purpose of the reproduction operator is to create a new population based on the evaluation (fitness) of the members of the old population. Our implementation uses the roulette wheel selection scheme described in [13]. Fitness scaling has also been implemented [13]. In addition to fitness scaling, two more heuristics have been incorporated in our implementation: the generation gap and the elitism strategy [18].

Crossover is applied after reproduction. Here, we are using a modified crossover operator which we call the *crossover_nodes* operator. The idea is to swap groups of weights feeding into the same node. The reason is quite plausible; each node in the network contributes to the solution that the network tries to find. Thus, weights feeding into a node serve a role in finding a solution for the problem at hand. Swapping weights arbitrarily may not make a lot of sense while swapping groups of weights feeding into nodes is more sensible. This operator has also been used in [17]. We have also considered a modified *crossover_nodes* operator. Swapping weights feeding nodes located at hidden layers higher than the first, may be disruptive since the internal knowledge representations between two different networks are probably quite different. However, swapping weights feeding nodes located at the first hidden layer only, may be less disruptive for the networks, since it can be considered as an exchange of feature detectors. We call this operator as the *crossover_first_layer_nodes* operator.

The last genetic operator used is the mutation operator. This operator picks randomly a member from the population and changes it slightly. In its simplest form, mutation changes the value of a weight by adding a small random value. Following our discussion regarding the *crossover_nodes* operator, the mutation operator used in this study does not change single weights but groups of weights feeding into a node. This modified operator which we call the *mutate_nodes* operator, has also been used in other studies [17].

3.3. Fitness evaluation

The choice of a fitness function is problem dependent and is probably the most critical issue in genetic algorithm design. When genetic algorithms are combined with neural networks, the most commonly used approach to evaluate the performance of a member in the population is to train the network represented by this member and record its mean squared error. This is quite inappropriate for our purpose, since it does not account for the network's generalization performance and size. To perform an evaluation based on network size and generalization, we have considered a fitness function having the following form:

$$E_{fit} = G_{net\_val} + \lambda_{GA}(1 - E_{net\_size}) \tag{3}$$

The first term (G_{net_val}) accounts for generalization, while the second term ($1 - E_{net\_size}$) accounts for the network size. The parameter λ_{GA} is a weighing factor which controls the importance of the two terms. If λ_{GA} is very small, the fitness of a member is mostly determined by its generalization performance only. However, when λ_{GA} assumes large values, both generalization, and size influence the fitness of a member. The value of the weighing factor λ_{GA} is determined adaptively, in a similar manner that λ_{WE} is determined in weight elimination. Specifically, λ_{GA} is determined as follows:

$$\lambda_{GA} = \lambda_{0\_GA} e^{-\beta_{GA} E_{net\_gen}} \tag{4}$$

where λ_{0_GA} and β_{GA} are constants specified by the user and E_{net_gen} denotes the generalization error of the network. The goal of the genetic algorithm is to find solutions which maximize the above fitness function. It is clear from the definition of the fitness function that reproduction favors members with good generalization performance and a relatively small network size. In early generations, network size does not play an important role in reproduction and the fittest members are the members which generalize best. However, in future generations both network size and generalization affect reproduction. For an estimation of E_{net_gen}, cross-validation is used again. Recalling our discussion in section 2, E_{net_gen} is defined to be $1 - G_{net\_val}$ where G_{net_val} is the generalization performance of a network over the validation set. The network size E_{net_size} is defined to be the number of effective connections of the network (connections whose associated absolute weight values are greater than $|w_0|$) over the total number of connections. Both E_{net_gen} and E_{net_size} take values between 0 and 1.

TABLE 1. Data sets used and network architectures chosen.

Data and Network Architectures					
Data set	Training set	Validation set	Test set	Classes	Architecture
Numbers	150	50	150	10	63-40-10
Ionosphere	200	31	120	2	34-30-2
Soybean	254	53	255	15	35-60-15
Breast-cancer	200	99	400	2	10-40-2
Wine	75	28	75	3	13-35-3
Iris	90	15	45	3	4-35-3
Balance	250	125	250	3	4-60-3
Cars	400	100	346	7	18-80-4

4. Simulations and results

In order to evaluate our approach, an extensive experimental study has been performed using one artificial and seven real databases. The real databases were selected from the collection of the databases distributed by the machine learning group at the University of California at Irvine [19]. For each problem, data was first normalized in [0,1] if that was necessary. Then data was divided into a training, a validation, and a test set. Details are provided in Table 1. Four approaches have been compared: the original back-propagation (BP), the back-propagation with weight elimination (BP_WE), the genetic algorithm approach using the *crossover_nodes* and *mutate_nodes* operators (GA_BP_WE), and the genetic algorithm approach using the *crossover_first_layer_nodes* and *mutate_nodes* operators (GA1_BP_WE).

For each problem considered, a two-layer network was chosen (see Table 1). The size of the networks chosen for each problem was considered to be big enough since we were able to successfully train smaller size networks for the same problems, without any particular difficulty. In the case of the BP and BP_WE techniques, experimental results were obtained by running 20 experiments with each method, for each database. For each experiment, a different set of initial weights was used. For comparison purposes, both the BP and BP_WE techniques used the same 20 initial weight configurations, for each database. In the case of the BP_WE technique, we had also to choose values for the parameters β_{WE} and w_0 (see Eq. (2)). The parameter w_0 was set to 1.0 as was mentioned in section 2. We have chosen a different β_{WE} value (in the range of [10 - 100]) for each of the 20 experiments performed per database. To challenge the genetic algorithm approach (which uses the same β_{WE} values as we will explain later), the same 20 β_{WE} values have been used for all the databases.

The population size P of the genetic algorithm approach was set equal to 20, that is, equal to the number of individual experiments performed for each database. The initial population was formed by first encoding the initial network and then copying it P times. The parameter values of each network (β_{WE}, w_0, and initial weights) were chosen exactly the same with those used in the individual experiments using the BP and BP_WE approaches. In other words, the setting of the initial population was exactly the same with the initial setting of the networks used in the 20 individual experiments performed for each database using the BP and BP_WE approaches. The objective was to demonstrate that the genetic algorithm approach utilizes exactly the same amount of information which was available to the networks trained with the BP and BP_WE approaches, however, it is capable of discovering better solutions, taking advantage of the interchange of information which takes place during evolution.

The evaluation of each network from the population was performed by first training each network for a number of epochs using the weight elimination technique. The number of epochs used to train each network was about 10% [20] of the average number of epochs required by the BP approach to converge for the same problem (average over 20 different experiments performed for each database using the BP approach). After a network in the population has been trained for a number of epochs, we compute its classification performance (G_{net_val} - classification performance over the validation set) and its size (E_{net_size} - effective number of connections over the total number of connections). Connections with small weights are not removed before the genetic algorithm has converged. This was also the case with the BP_WE approach.

The convergence of the genetic algorithm was determined by considering the improvement I_n at each generation. The improvement I_n at the n-th generation is defined as the average fitness at the n-th generation over the average fitness at the $(n-1)$-th generation. To avoid early convergence, we have used the following criterion based on the average improvement A_n defined as follows:

$$A_n = \gamma A_{n-1} + (1-\gamma) I_n \tag{5}$$

A_n is the average improvement at step n and γ is a constant usually chosen very close to 1.0. Here, γ was set equal to 0.9. A_0 can be computed by evaluating each member of the initial population before evolution begins. New populations are allowed to form as far as the average improvement keeps increasing, that is, while $A_n \geq A_{n-1}$.

TABLE 2. Comparison of the best solutions obtained

Generalization and network size of best solutions								
Data set	BP		BP_WE		GA_BP_WE		GA1_BP_WE	
	test	size_reduction	test	size_reduction	test	size_reduction	test	size_reduction
Numbers	0.8333	0%	0.833	69.3%	0.84	68%	0.853	68.96%
Ionosphere	0.942	0%	0.96	96.67%	0.975	92.54%	0.983	93.7%
Soybean	0.83	0%	0.827	28%	0.839	63.7%	0.835	57.43%
Breast-cancer	0.97	0%	0.975	96.36%	0.98	96.93%	0.98	97.51%
Wine	0.92	0%	0.933	90.6%	0.96	90.6%	0.947	86.45%
Iris	1.0	0%	1.0	87.28%	1.0	87.98%	1.0	88.34%
Balance	0.908	0%	0.908	47.83%	0.908	85.1%	0.912	85.3%
Cars	0.795	0%	0.792	2.1%	0.81	44.36%	0.795	38.6%

In all the simulations performed, the learning rate and momentum were both set equal to 0.1. The generation gap was set equal to 0.9 while the parameter which determines the number of best fit copies in future populations (called *Cmult* in [13]) was set equal to 1.5. The crossover and mutation probabilities were chosen 0.6 and 0.001 correspondingly. λ_{0_WE} and λ_{0_GA} were both set to 1 (Eq. (2) and (4) correspondingly). Different β_{GA} values (Eq. (4)) were used during our experimentation. The best solutions obtained correspond to β_{GA} values in the range of [1.0 - 10.0]. For β_{GA} values much greater than 10.0, we did not observe a great reduction in terms of network size.

A detailed description of the experimental results for each one of the databases can be found in [21]. Here, we present only a summary of the results we obtained (Table 2). The first column presents the database used while the next columns present the best network solutions found for each of the four methods we considered in our experiments. For each method we report the best generalization performance achieved and the reduction in network size associated with this solution. It is obvious that the pruned networks obtained by the combination of genetic algorithms and weight elimination have much better generalization capabilities. In terms of network size, the results are also much better in many cases (for example, in the case of the Soybean database). The use of the *crossover_first_layer_nodes* operator seems to be very beneficial in some cases.

5. Conclusions

An extensive experimental study involving one artificial and seven real databases has demonstrated that the coupling of genetic algorithms with weight elimination is a very promising approach. Actually, the framework of our approach is more general since other pruning techniques can be also coupled with genetic algorithms. Weight elimination depends on a number of parameter values whose choice can significantly affect the results. In fact, we show that small size networks can be obtained, however, the generalization performance of these networks is not always satisfactory. On the other hand, the networks obtained by the proposed approach not only are small in size but they also have better generalization capabilities.

References

[1] E. Karnin, "A simple procedure for pruning back-propagation trained neural networks", *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239-242, 1990.

[2] M. Mozer and P. Smolensky, "Skeletonization: a technique for trimming the fat from a network via relevance assessment", in *Advances in Neural Information Processing Systems 1*, pp. 105-115, 1989.

[3] Y. Chauvin, "A back-propagation algorithm with optimal use of hidden units", in *Advances in Neural Information Processing Systems 1*, pp. 519-526, 1989.

[4] A. Krogh and J. Hertz, A simple weight decay can improve generalization", in *Advances in Neural Information Processing Systems 4*, pp. 950-957, 1992.

[5] S. Hanson and L. Pratt, "Comparing biases for minimal network construction with back-propagation", in *Advances in Neural Information Processing Systems 1*, pp. 177-185, 1989.

[6] C. Ji, R. Snapp, and D. Psaltis, "Generalizing smoothing constraints from discrete samples", *Neural Computation*, vol. 2, pp. 188-197, 1990.

[7] J. Depenau and M. Moller, "Aspects of generalization and pruning", in *Proceedings of World Congress on Neural Networks*, vol. III, pp. 504-509, 1994.

[8] H. Thodberg, "Improving generalization of neural networks through pruning", *International Journal of Neural Systems*, vol. 1, no. 4, pp. 317-326,

[9] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-propagation algorithm which varies the number of hidden units" *Neural Networks*, vol. 4, pp. 61-66, 1991.

[10] J. Sietsma and R. Dow, "Creating artificial neural networks that generalize", *Neural Networks*, vol. 4, pp. 67-79, 1991.

[11] B. Hassibi and D. Stork, "Second order derivatives for network pruning: optimal brain surgeon", in *Advances in Neural Information Processing Systems 5*, 1993.

[12] R. Kamimura and S. Nakanishi, "Weight-decay as a process of redundancy reduction", in *Proceedings of World Congress on Neural Networks*, vol. III, pp. 486-489, 1994.

[13] D. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, Reading, MA., 1989.

[14] A. Weigend, D. Rumelhart and B. Huberman, "Generalization by weight elimination with application to forecasting", in *Advances in Neural Information Processing Systems 3*, pp. 875-882, 1991.

[15] A. Weigend, D. Rumelhart and B. Huberman, "Back-propagation, weight-elimination and time series prediction", *Proceedings of the 1990 Connectionist Models Summer School*, pp. 105-116, 1990.

[16] D. Hush and B. Horne, "Progress in supervised neural networks", *IEEE Signal Processing Magazine*, pp. 8-39, Jan. 1993.

[17] D. Montana and L. Davis, "Training feed-forward neural networks using genetic algorithms", in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 762-767, 1989.

[18] J. Grefenstette, "Optimization of control parameters for genetic algorithms", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 122-128, 1986.

[19] P. Murphy and D. Aha, *UCI Repository of machine learning databases*, [Machine-readable data repository]. Irvine, CA: University of California, Department of Information and Computer Science, 1994.

[20] M. Caudill, "Evolutionary neural network", *AI Expert*, pp. 29-33, March 1991.

[21] G. Bebis, M. Georgiopoulos, and T. Kasparis, "Coupling weight elimination and genetic algorithms to reduce network size and improve generalization", Dept. of Electrical and Computer Engineering, University of Central Florida, Orlando, June 1995.