A hybrid method for searching near-optimal artificial neural networks

Leandro M. Almeida, Teresa B. Ludermir
Federal University of Pernambuco – Center of Informatics
P.O. Box 7851, Cidade Universitária, Recife - PE, Brazil, 50732-970
{lma3,tbl}@cin.ufpe.br

Abstract

This paper describes a method for searching near-optimal neural networks using Genetic Algorithms. The method uses an evolutionary search with the simultaneous selection of initial weights, transfer functions, architectures and learning rules. Experimental results have shown that the method is able to produce compact, efficient networks with satisfactory generalization power and shorter training times in comparison to other algorithms.

1. Introduction

Artificial Neural Networks (ANNs) have been successfully applied in fields such as pattern recognition, speech recognition, signal processing and function approximation [1, 10]. Neural network effectiveness is conditioned by the specific choice of parameters for a given problem. An ANN model can obtain good performance when its parameters have been correctly defined. An optimal ANN can be seen as a instance of a neural network tailored to a specific problem, thereby having a smaller architecture with faster convergence and better generalization performance [1, 10].

The specific and correct (near-optimal) configuration of ANN models for a certain problem through trial-and-error is considered a tedious, less productive and error-prone task [1]. The construction of near-optimal ANN models involves difficulties such as the exponential number of parameters that need to be adjusted (number of hidden layers, number of hidden units, training algorithms, transfer functions, learning rate, etc); the need for a priori knowledge of the problem domain and ANN functioning to define these parameters; and the presence of an expert when such knowledge is lacking.

The manual search process of near-optimal ANNs remains a challenge even when rules regarding the use of ANNs are followed (PROBEN1 Report [8]). Therefore, the automatic search of near-optimal models appears to be a good solution and avoids the manual trial-and-error approach. The automatic search process of near-optimal ANN models is widely explored, using evolutionary techniques. In this approach, evolutionary techniques and ANNs are combined to produce models with low error and high generalization control, yielding Evolutionary ANNs (EANNs), as described by Yao [10]. EANNs are seen as a special kind of ANN, as they allow the exploration of many necessary aspects or components in building well-performing ANN models. EANNs are able to search initial weights, transfer functions, topology setups and learning rules (training algorithm parameters).

One kind of evolutionary technique, the Genetic Algorithm (GA), is often used to search near-optimal ANN models with topology optimization, as presented in [3, 4]. Others include transfer functions, initial weights and learning rules, as presented in [1, 2]. Work using non-evolutionary techniques prune connections that are considered less significant [6, 7] or freeze weights when the same inputs are submitted to the network [6].

In this work, we present a method named NNGA-DCOD for near-optimal ANN searching using ANN and GA with direct encoding. It is based on the previous work of Yao and Abraham [1, 10]. In this method, we search for weights, transfer functions, architectures and learning algorithm rules. The methodology consists of an evolutionary search system in layers, where a GA searches for learning rules in the first layer,another GA searches for architectures and transfer functions in the second layer and, yet another GA searches for initial weights in the final layer. Experiments using real data sets were performed and showed interesting results when compared to other methods found in the literature.

This paper is organized as follows: Section 2 presents the NNGA-DCOD in more details. Section 3 describes experimental results using NNGA-DCOD. Section 4 summarizes our conclusions and presents future work.

2. NNGA-DCOD Method

The motivation for Hybrid Systems (HS) is related to the fact that the combination of the best features of two or more techniques is better than the application of these techniques separately. An example of HS is the merging of GA and ANN for searching near-optimal ANN models. In work by Yao [10], we are given diverse definitions and discussions on the hybridization of GA and ANN. One such definition refers to the layered process of an evolutionary search of ANNs, as illustrated in Figure 1.

In this process, the evolution speed is faster with the initial-weight layer than with the others. This is a consequence of the high dimensionality of exploration space for initial weights due to the lack of priori knowledge on excellent sets of initial weights. The higher layers search for architectures and learning algorithm rules, which have more a priori knowledge information and allow the restriction of a more specific search space.

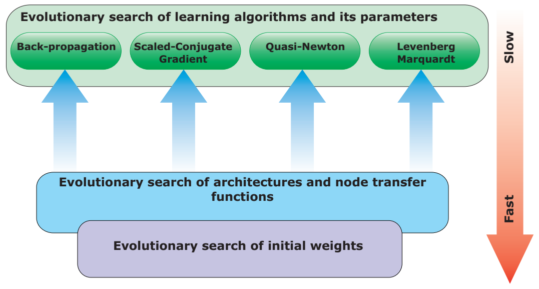
Evolutionary search of learning algorithms and its parameters
Back-propagation
Scaled-Conjugate Gradient
Quasi-Newton
Levenberg Marquardt
Slow
Evolutionary search of architectures and node transfer functions
Evolutionary search of initial weights
Fast

Figure 1. Evolutionary search in layers [1].

The idea of an evolutionary search in layers for near-optimal ANNs is motivated by the fact that its parameters vary, producing different results for the same problem. This applies mainly to the number of hidden layers and neurons, the type of transfer functions, the values of initial weights and the options of training algorithm parameters [5, 9].

GAs are used in the NNGA-DCOD to locate basins of attraction, where near-optimal solutions are more likely to be found. Four different ANN training algorithms are then applied to refine the search in these basins: Back-propagation(BP), Levenberg-Marquardt(LM), quasi-Newton Algorithm(QNA) and Scaled Conjugate Gradient (SCG). In NNGA-DCOD, there is a GA for each layer described in Figure 1. These GAs exchange information for the search of near-optimal ANNs that have all the necessary configurations for their correct functioning. Table 2 describes the parameters used in the NNGA-DCOD method. GA and ANN parameters were defined after the observation of executions of the algorithms developed. The training algorithm parameters are the same used by Abraham in [1].

The three GAs use 3 nested data structures for their individuals. The first is composed of a set of parameters for

Parameters for:		Values
GA	- Coding mode	Direct
	- Elitism	10%
	- Mutation	40%
	- Selection	Tournament
	- Population / Generation	Algorithms: 7 / 30
		Architectures: 10 / 7
		Initial Weights: 10 / 5
Neural Networks	- Type	MLP, feed-forward
	- Learn Algorithms	BP, LM, QNA and SCG
	- Transfer Functions	Purelin (P), Tang-sigmoid (T) and Log-Sigmoid (L)
	- Number of hidden layers / nodes	up to 3 / 16
	- Number of training epochs	up to 5
	- Range of initial weights	-/+ 0.5
	- Function error	MSE
Learning Algorithms	BP - Learning Rate and Momentum	0.05 – 0.25
	LM - Learning Rate	0.001 – 0.02
	QNA - Step lengths	1.0E-06 – 100
	- Limits on step sizes	0.1 – 0.6
	- Scale factor to determine performance	0.001 – 0.003
	- Scale factor to determine step size	0.001 – 0.02
	SCG - Change in weight for second derivative approximation	0 – 0.0001
	- Regulating the indefiniteness of the Hessian	0 - 1.0E-06

Table 1. NNGA-DCOD parameters.

the training algorithms and a population of the second data structure. The second is composed of a set of architectures and a population of the third data structure. Finally, the third data structure is composed of a set of sets of initial weights.

Figure 2 shows the general idea of the algorithm. The search execution for near-optimal ANNs is a time-consuming process, in which the upper layers depend on the lower layers of evolutionary search. Therefore, the execution of one generation of parameter search only occurs after the execution of the generations for the architecture search. Similarly, each generation of search for architectures occurs only after the execution of the generations for the weight search. For each generation of evolutionary search, many ANN models are trained by different algorithms, thereby considerably increasing execution times.

```
1 - Randomly generate the populations for: initial weights, architectures and
parameters of the learning algorithms;
2- For each learning algorithm, do:
        2.1 - Evaluate fitness of all the individuals.
        2.2 - For each generation of the search for parameters:
          2.2.1 - For each generation of the search for architectures:
            2.2.1.1 - For each generation of the search for initial weights:
              i - Select parents for reproduction based on fitness value;
              ii - Apply genetic operators and produce offspring for next
              generation. Refill the population back to defined size.
            2.2.1.2 - Perform steps i and ii for the architectures;
          2.2.2 - Perform steps i and ii for the parameters.
```

Figure 2. NNGA-DCOD algorithm.

The fitness measure for initial-weight individuals is mainly composed of the ANN Mean Squared Error (MSE) in the test set. In the case of individuals with the same test error, the training error is used as the settling criterion. The fitness for individuals of the population of architectures is the weighted combination of the test error of the best initial set of weights and the architecture complexity given as the percentage of hidden neurons from the total number. Finally, the fitness function for the parameters of the learning algorithms is the mean of the architecture fitness of the its three best architectures. Experiments performed with NNGA-DCOD are described in more details in the next section.

3. Experimental Studies

We have selected three well-known benchmark problems to evaluate the NNGA-DCOD method. These benchmarks are the cancer data set that consists of 699 examples (EX), 9 attributes (AT) and 2 classes(CL); the diabetes data set with 768 EX, 8 AT and 2 CL; and the heart data set with 297 EX, 13 AT and 5 CL. Data were obtained from real world problems in the medical field available from the UCI machine learning benchmark repository.

For each experiment, the data sets were randomly partitioned with stratification, i.e., maintaining the same proportions of the number of classes for the training (50%), validation (25%) and test sets (25%). The performance of the search for parameters of the learning algorithms was the mean MSE of the best ANNs found after every 30 generations.

For the ANN search by trial-and-error, we performed 30 runs for the following network setup: 8, 10, 12, 14, 16, 18, 20, 22 and 24 hidden neurons for one hidden layer with the (T) transfer function. The purpose of these experiments was to compare the performance between NNGA-DCOD and the manual process using the same database split scheme and number of training epochs.

Figure 3 shows the experimental results of the NNGA-DCOD for the cancer, diabetes and heart problems with the mean of the best ANNs of each generation of the search for parameters of the learning algorithms. These best means are used for building the charts with a confidence interval for every training algorithm. The intention of the charts is to identify significantly different performances. In some cases, the difference between training algorithms not is visually clear. Thus, statistical tests, such as the t-test, are performed in these cases. The results are described in the next subsection.

3.1. Experimental results

LM achieved the best mean performance among the tested algorithms for the cancer and diabetes problems, with a 95% confidence interval obtained by t-test. LM performance had a small standard deviation of results when compared with BP, which undergoes greater interference from the weight initialization process than all the other algorithms. The QNA algorithm had the best mean performance for the heart problem, but with a 93% confidence interval due to the fact that SCG and LM presented similar results.

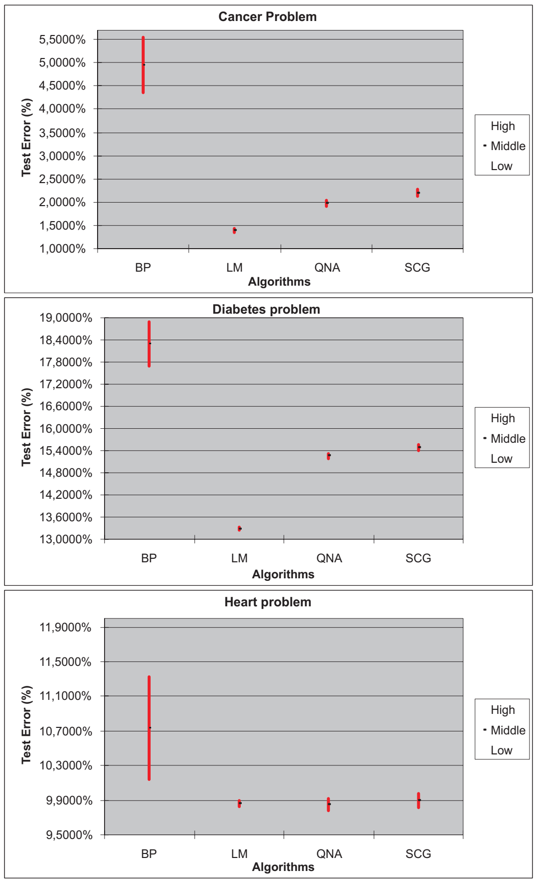

Figure 3. Performances with interval confidence for all training algorithms and data sets.

The better results obtained with LM are a consequence of it being a method that works with second derivative information and converges faster than first-order methods. LM is different from other algorithms that need more training time to achieve better adjustment of the weights of the network, as the example of BP.

The best networks showed that a small number of training epochs did not prevent the method from achieving satisfactory performances, since the parameters of the network were well-chosen. A good set of parameters for ANNs that are tailored to a given problem contributes toward a better performance of the network with regard to problem.

Table 2 shows the best networks found with the NNGA-DCOD method and the manual process. The cancer and diabetes problems achieved the best results with the LM algorithm, which obtained simple architecture and excellent performance for the cancer problem in comparison to the other algorithms. For the diabetes problem, LM achieved the best test error, but its architecture is more complex than that obtained with QNA. Thus, QNA can be considered the best solution for having a simpler architecture as well as satisfactory performance. For the heart problem, QNA again achieved the best network performance with the smallest possible configuration for an MLP.

Data sets / Algorithms		NNGA-DCOD			Trial-and-error	
		MSE		Architecture	MSE	Architecture
		Training	Test		Test	
Cancer	BP	0.0444	0.0428	3 P	0.1464	24 T
	LM	0.0293	0.0119	1 T , 1 T, 1 T	0.0195	8 T
	QNA	0.0328	0.0135	4 T, 1 T	0.0439	20 T
	SCG	0.0321	0.0129	3 T, 1 T	0.0459	8 T
Diabetes	BP	0.1823	0.1750	3 P	0.2518	22 T
	LM	0.1571	0.1315	2 T, 5 T	0.1468	14 T
	QNA	0.1606	0.1441	1 P, 1 T, 1 T	0.1637	18 T
	SCG	0.1628	0.1521	1 P	0.1861	14 T
Heart	BP	0.1212	0.1021	1 P	0.4210	14 T
	LM	0.1165	0.0978	2 P, 5 T, 2 P	0.1012	10 T
	QNA	0.1212	0.0959	1 P	0.1667	8 T
	SCG	0.1214	0.0971	1 P	0.1678	20 T

Table 2. The best ANNs found through NNGA-DCOD and trial-and-error for all data sets.

The algorithm developed achieved the best results for all the data sets when compared with the results achieved through trial-and-error. Table 3 displays the performance comparison form the results obtained with NNGA-DCOD and other work found in the literature. Comparisons between these methods must be made with caution, as the results are obtained with different experimental model setups and the errors are estimated with different methods. Nonetheless, the table shows that NNGA-DCOD is able to achieve very interesting results with very compact networks and few training epochs. For the diabetes problem, the NNGA-DCOD produced the best results among the methods tested, whereas for the other problems, the algorithm developed is among those that achieved the best results and produced interesting performances.

Method	Error		
	Cancer	Diabetes	Heart
NNGA-DCOD^e	0.0119	0.1315	0.0959
GEPNET^e [4]	–	0.1927	0.1368
COVNET^e [4]	–	0.1990	0.1426
MOBNET^e [4]	–	0.1984	0.1363
CNNDA^{ne} [6]	0.0116	0.1875	–
COOPNN-ENSEMBLE^e [3]	0.0057	0.1615	0.0735

Table 3. Comparison between Evolutionary(e) and non-Evolutionary(ne) methods.

4. Conclusions

We have proposed an efficient method for searching near-optimal ANNs using GA and ANN with direct encoding and few training epochs. The results show that this method is able to achieve compact networks with satisfactory performances. Moreover, the NNGA-DCOD comparison with other methods shows very interesting and encouraging results that call for the continuation of this research and the improvement of the work.

There are a number possible improvements for future work. For example, the conduction of more experiments with other types of problems, such as time-series for the verification of the behavior of our method with prediction problems; and the revision of the selection criterion for less complex networks performed by the genetic operators that currently discard complex nets that have satisfactory performance.

Acknowledgments

The authors would like to thank CNPq (Brazilian Research Council) for their financial support.

References

[1] A. Abraham. Meta learning evolutionary artificial neural networks. *Neurocomputing*, (56):1–38, March 2004.

[2] K. P. Ferentinos. Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms. *Neural Networks*, 18(7):934–950, 2005.

[3] N. García-Pedrajas, C. Hervás-Martínez, and D. Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Trans. Evolutionary Computation*, 9(3):271–302, 2005.

[4] N. García-Pedrajas, D. Ortiz-Boyer, and C. Hervás-Martínez. Cooperative coevolution of generalized multilayer perceptrons. *Neurocomputing*, 56:257–283, 2004.

[5] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[6] M. M. Islam and K. Murase. A new algorithm to design compact two-hidden-layer artificial neural networks. *Neural Networks*, 14(9):1265–1278, 2001.

[7] L. Ma and K. Khorasani. New training strategies for constructive neural networks with application to regression problems. *Neural Networks*, 17(4):589–609, 2004.

[8] L. Prechelt. PROBEN1 — A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, september 1994.

[9] J. J. Torres, M. A. Muñoz, J. Marro, and P. L. Garrido. Influence of topology on the performance of a neural network. *Neurocomputing*, 58-60:229–234, 2004.

[10] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, September 1999.