ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Automatically searching near-optimal artificial neural networks

Leandro M. Almeida and 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. The idea of automatically searching neural networks that learn faster and generalize better is becoming increasingly widespread. In this paper, we present a new method for searching near-optimal artificial neural networks that include initial weights, transfer functions, architectures and learning rules that are specially tailored to a given problem. Experimental results have shown that the method is able to produce compact, efficient networks with satisfactory generalization power and shorter training times.

1 Introduction

Manual searching of Artificial Neural Networks (ANNs) for a specific problem currently relies heavily on human experts with sufficient knowledge on the different aspects of the network as well as the problem domain. When the complexity of the problem domain increases and when near-optimal networks are desired, manual searching becomes more difficult and unmanageable [1]. An optimal ANN can be seen as an instance of a neural network tailored to a specific problem, thereby having a smaller architecture with faster convergence and better generalization performance [1, 2, 3]. 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, 3]. The construction of near-optimal ANN models involves difficulties such as the exponential number of parameters that need to be adjusted; 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 [3].

The automatic searching 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. One kind of evolutionary technique, the Genetic Algorithm (GA), is often used to search near-optimal ANN models with topology optimization, as presented in [4, 5]. Other works include transfer functions, initial weights and learning rules, as presented in [1, 3, 6]. There are also works that employ non-evolutionary techniques, which prune connections that are considered less significant [7, 8] or freeze weights when the same inputs are submitted to the network [7].

In this work, we present an ameliorated version of a method developed for searching near-optimal networks using ANN and GA with direct encoding,

named NNGA-DCOD (aNN + GA - Direct enCODe) presented in [3]. The NNGA-DCOD differ of other works because search ANNs having high performance, low complexity, trained with five epochs only and because the evolutionary searches uses individuals with direct encoding [3]. NNGA-DCOD achieved good results in the primary experiments, motivating the improvement of the method and, consequently, increased experimentation in order to verify the real power of this method in automatically searching near-optimal ANNs for a given problem. This paper is organized as follows: Section 2 presents the NNGA-DCOD; Section 3 describes experimental results; and Section 4 summarizes our conclusions and presents future work.

2 NNGA-DCOD Method

NNGA-DCOD adopts a framework for searching ANNs, as defined by Yao [2]. The framework used in the present work has a layered process of an evolutionary search of ANNs, as illustrated in Figure 1. In this process, the evolution speed is faster with the initial-weights layer than with the other layers. This is a consequence of the high dimensionality of exploration space for initial weights due to the lack of a priori knowledge on excellent sets of initial weights. The higher layers search for architectures and learning algorithm, which have more a priori knowledge and allow the restriction of a more specific search space.

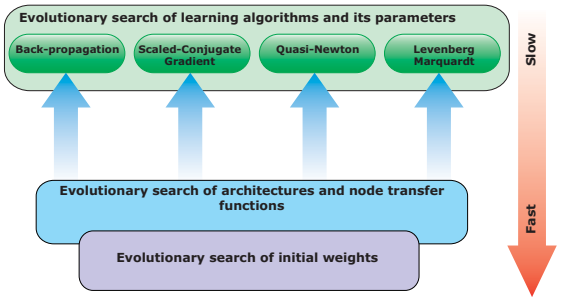

Fig. 1: Evolutionary search in layers [1].

The variation of ANN parameters (training algorithm parameters, initial weights, architecture, etc.) produce different results for the same problem [9], contributing to the use of an evolutionary search of the layers. In the NNGA-DCOD GAS is used 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, which exchange information for the search of near-optimal ANNs. Table 1 describes the parameters used in the NNGA-DCOD method. GA and ANN parameters were defined after the observation of executions of the algorithms

	Parameters for:	Values
GA	- Encoding	Direct
	- Elitism / Mutation	10% / 40%
	- Selection	Tournament
	- Population/Generation (Algorithms|Architec.|Weights)	7/30 | 10/5 | 10/5
ANN	- Type	MLP feedforward
	- Transfer functions	Pure-linear (P), Tang-sigmoid (T) and Log-Sigmoid (L)
	- Number of: hidden layers / nodes / training epochs	up to: 3 / 16 / 5
	- Range of initial weights	[-0.5, 0.5]
	- Output neuron	linear
Training alg.	BP - Learning rate and momentum	[0.05, 0.25]
	LM - Learning rate	[0.001, 0.02]
	SCG - Step lengths	[1.0E-06, 100]
	- Limits on step sizes	[0.1, 0.6]
	QNA - Scale factor to determine performance	[0.001, 0.003]
	- Scale factor to determine step size	[0.001, 0.02]
	- 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.

developed. The training algorithm parameters are those used in [1].

The method starts the search by randomly generating the populations for all kinds of individuals, Next, the fitness is calculated based on the ANN Mean Squared Error (MSE) achieved in the training set. The genetic operators maintain the diversity of individuals for the search of all layers with a tournament and a small range of aleatory selection, where the new individuals generated for the next offspring must be distinct from individuals of the actual offspring. Apparently, the amount of individuals used on the NNGA-DCOD is small, but as this method is iterative with nested loops, many new individuals are created at every generation of each kind of search. Thus, the search space explored is large and satisfactory results are achieved. Therefore, 3 nested data structures are used for individuals. The first is composed of a set of parameters for 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.

The improvements executed on the NNGA-DCOD refer to the fitness calculation and solution evaluation found in the search process. Currently, the error information from the training set is used for the calculation of the fitness of all individuals. The error from the validation set is used only to present the gradual evolution of the search. Lastly, the near-optimal solutions found by the method are tested with unseen data along with a process search (test set). These changes the search developed process make reliable and honest and were been adopted based on benefits previously pointed out and discussed by [10].

3 NNGA-DCOD: Experimentation setup

We have selected five well-known benchmark problems to evaluate the NNGA-DCOD method: Cancer with 9 attributes(at), 699 examples(ex) and 2 classes(cl);

Glass (9at, 214ex, 6cl); Heart-Cleveland (35at, 303ex, 2cl); Horse (58at, 364ex, 3cl); and Prima-diabetes (8at, 768ex, 2cl). To perform the experiments, we used five iterations of two-fold cross-validation (5 x 2 cv). At each iteration, the data were randomly divided into halves. One half was the input for the algorithms (70% for training and 30% for the validation set), and the other half was used to test the final solution (test set). To determine whether the differences among the algorithms are statistically significant, we used a combined F-test described by [10]. Let $p_i^{(j)}$ denote the difference in the accuracy of two classifiers in fold j of the i-ht iteration of 5 x 2 cv, $\bar{p} = (p_i^{(1)} + p_i^{(2)})/2$ denote the mean, and $s_i^2 = (p_i^{(1)} - \bar{p})^2 + (p_i^{(2)} - \bar{p})^2$ the variance, then

$$f = \frac{\sum_{i=1}^{5} \sum_{j=1}^{2} (p_i^{(j)})^2}{2 \sum_{i=1}^{5} s_i^2}$$

is approximately F distributed with ten and five degrees of freedom. We rejected the null hypothesis that the two algorithms have the same error rate with a 0.05 significance level if $f > 4.74$. The accuracy results presented in the next section are based on error information from the ten test sets. This methodology was used in experiments due to the fact that usual method generates an increase of type-I errors: The results are incorrectly deemed significantly different more often than expected, given the level of confidence used in the test [10].

The ANN search through trial-and-error was performed following the previously described methodology, using the same database split scheme and number of training epochs. We performed 30 runs in each fold for the following network setup: 2, 4, 6, 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.

3.1 NNGA-DCOD: Experimentation results

LM achieved the best mean performance among the tested algorithms for the cancer, prima and glass problems, proving significantly different ($\alpha = 0.05$) and better than the other algorithms. No tested algorithms was significantly different or better for heart and horse problems, but LM was among those that achieved better results for these problems. 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 a better adjustment of the weights of the network, such as BP.

Table 2 shows the near-optimal networks found with the NNGA-DCOD method. These networks are better than those found with the manual method, having simple architectures and in some cases having the smallest possible configuration for a MLP. Even with a very simple architecture, the networks found automatically achieved satisfactory error performances and can be considered as

near-optimal solutions having a simple structure and satisfactory performance. The majority of near-optimal networks were achieved with the LM algorithm, but in some cases, such as the heart problem, the best solution was obtained with SCG algorithm. Other solutions that use simple training algorithms can be adopted on place of the LM algorithm. Examples of this are the heart and horse problem, where the solution found with QNA and SCG, respectively, had a similar performance, but with a simpler structure and training algorithm.

The near-optimal networks demonstrate 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 the problem. A good example can be seen in the performance of solutions obtained with BP in the automatic and manual methods. The reduction of error is very significant comparing the two methods for searching near-optimal ANNs in the all problems.

Data sets / Algorithms		NNGA-DCOD				Trial-and-error			
		Near-optimal ANN		Mean and StD		Near-optimal ANN		Mean and StD	
		Test error	Setup			Test error	Setup		
Cancer	BP	0.0371	3P	0.0464	0.0064	0.1821	4T	0.2333	0.0461
	LM	0.0177^b	1L	**0.0249**	0.0047	0.0200	12T	0.0264	0.0054
	QNA	0.0221	1L	0.0301	0.0043	0.0274	4T	0.0321	0.0044
	SCG	0.0235	1T	0.0306	0.0038	0.0248	2T	0.0336	0.0093
Prima	BP	0.1854	3P	0.1912	0.0037	0.2335	12T	0.2706	0.0275
	LM	0.1477^b	4T	**0.1548**	0.0044	0.1559	6T	0.1575	0.0052
	QNA	0.1605	9P	0.1672	0.0041	0.1761	8T	0.1805	0.0116
	SCG	0.1595	1P	0.1669	0.0040	0.1748	10T	0.1820	0.0092
Heart	BP	0.1090	1T	0.1395	0.0167	0.2279	6T	0.3010	0.0431
	LM	0.0960	4L	0.1212	0.0131	0.1109	2T	0.1351	0.0125
	QNA	0.1001	1T	0.1234	0.0135	0.1208	2T	0.1404	0.0183
	SCG	0.0889^b	3T	0.1252	0.0189	0.1164	4T	0.1405	0.0146
Horse	BP	0.1563	1P	0.1660	0.0057	0.2310	10T	0.3334	0.0783
	LM	0.1427^b	3T	0.1521	0.0043	0.1606	4T	0.1688	0.0053
	QNA	0.1444	9T	0.1554	0.0073	0.1615	6T	0.1680	0.0038
	SCG	0.1497	1L	0.1551	0.0041	0.1623	8T	0.1699	0.0042
Glass	BP	0.1129	1P	0.1176	0.0039	0.2147	8T	0.2858	0.0766
	LM	0.0864^b	4L	**0.0940**	0.0044	0.0928	6T	0.0959	0.0082
	QNA	0.1003	6T	0.1029	0.0021	0.1048	12T	0.1092	0.0031
	SCG	0.1001	7P	0.1026	0.0019	0.1105	16T	0.1116	0.0042

Table 2: Near-optimal ANNs found through NNGA-DCOD and trial-and-error for all data sets. Results significantly different and better than others are highlighted in **bold**. Best solutions between the near-optimal are marked with "^{b}".

Table 3 displays the performance comparison from 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, horse and glass problems, the NNGA-DCOD produced the best results among the methods tested, whereas for the other problems, he is among those that achieved the best results, producing interesting performances.

Method	Error				
	Cancer	Prima	Heart	Horse	Glass
NNGA-DCOD^e	0.0177	0.1477	0.0889	0.1427	0.0864
GEPNET^e [5]	–	0.1927	0.1368	–	0.3516
COVNET^e [5]	–	0.1990	0.1426	–	–
MOBNET^e [5]	–	0.1984	0.1363	–	0.3516
CNNDA^{ne} [7]	0.0116	0.1875	–	–	–
COOPNN-ENSEMBLE^e [4]	0.0057	0.1615	0.0735	0.2088	0.1321

Table 3: Comparison between Evolutionary^e and non-Evolutionary^{ne} methods.

4 Conclusions

In this paper, we presented an ameliorated version of a method for searching near-optimal ANNs. The results show that this method is able to achieve compact networks with satisfactory performances, even when compared with other methods. The improvements helped to demonstrate the power of this method, but due to the rigorous criterion of selection, the near-optimal networks found in this work had only one hidden layer. Future work will be the revision of the selection criterion that currently discard complex networks even when having a 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, 2004.

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

[3] L. M. Almeida and T. B. Ludermir. A hybrid method for searching near-optimal artificial neural networks. *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, 2006.

[4] 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.

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

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

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

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

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

[10] E. Cantú-Paz and C. Kamath. An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(5):915–927, 2005.