Neural Network Hybrid Learning: Genetic Algorithms & Levenberg-Marquardt

Ricardo B. C. Prudêncio and Teresa B. Ludermir

Center of Informatics, Federal University of Pernambuco
P.O.Box 7851, Cidade Universitaria, Recife-PE, Brazil, 50.732-970

Abstract. The success of an Artificial Neural Network (ANN) strongly depends on its training process. Gradient-based techniques have been satisfactorily used in the ANN training. However, in many cases, these algorithms are very slow and susceptible to the local minimum problem. In our work, we implemented a hybrid learning algorithm that integrates Genetic Algorithms(GAs) and the Levenberg-Marquardt(LM) algorithm, a second order gradient-based technique. The GA-LM algorithm was used to train a Time-Delay Neural Network for river flow prediction. In our experiments, the GA-LM hybrid algorithm obtained low prediction errors within a short execution time.

1 Introduction

Artificial Neural Networks (ANNs) have been deployed in a variety of real world problems (Haykin (1994)). The success of ANNs for a particular problem depends on the adequacy of the training algorithm regarding the necessities of the problem. The existing gradient-based techniques (Battiti (1992)), in particular the Backpropagation algorithm (Rumelhart et al.(1986)), have been widely used in the training of ANNs. However, as these algorithms perform local searches, they are susceptible to the local minimum problem (Masters 1995). The use of stochastic algorithms, such as Genetic Algorithms (GAs)(Goldberg (1989)), is an interesting alternative for ANN training, since they are less sensitive to local minima. Nevertheless, they are generally slow compared to the fastest versions of gradient-based-algorithms.

In this light, we implemented a new hybrid algorithm that integrates Genetic Algorithms (GAs) and the Levenberg-Marquardt (LM) algorithm (Levenberg (1944))(Marquardt (1963)). Our algorithm aims to combine the capacity of GAs in avoiding local minima and the fast execution of the LM algorithm. In our experiments, we trained a Time Delay Neural Network (TDNN) (Lang and Hinton (1988)) for a river flow prediction problem. Our experiments revealed the following: (1) the implemented algorithm generated networks with good training performance, regarding the error obtained in the validation data, and good generalization performance, regarding the test errors; and (2) the hybrid algorithm was very efficient in terms of the execution time. Based on these results, we opted to integrate the GA-LM algorithm in a ANN design system previously proposed in (Prudêncio and Ludermir (2001a)) for river flow prediction problems.

In what follows, we present issues on training algorithms in section 2, and an overview of the hybrid GA & LM training approach in section 3. Section 4 presents the case-study of river flow prediction and section 5 presents the experiments comparing the hybrid GA-LM approach to other training procedures. Section 6 shows our final remarks.

2 Deterministic and stochastic algorithms

Neural Networks training algorithms can be classified as deterministic or stochastic (Masters (1995)), according to the optimization algorithm used to minimize the cost function. The former type, which includes the various gradient-based algorithms, such as Backpropagation(BP) (Rumelhart et al. (1986)) and Conjugate Gradient (Barnard and Cole (1989)), is characterized by the use of deterministic search operators. In general, these algorithms determine, from a starting point in the search space, a direction which minimizes the error in that given point and steps toward this direction. These algorithms are very sensitive to fall in local minima, since they perform local searches. This problem is crucial in the training of ANNs, since the error surface usually contains multiple local minima and few global minima.

A usual way to minor this effect is by random restarting the initial weights a number of times, and keeping the trained weigths with best results. However, some issues must be addressed regarding the number of restarts: (1) when this number is small, the obtained results may be unsatisfactory. This will depend on how affected is the algorithm by the weight initialization; (2) when the number of restarts is high, the training results tend to be better, however the execution time increases significantly, specially for Backpropagation. This process is probably more plausible if we consider more efficient algorithms, such as those of second-order (Battiti (1992)), including the Levenberg-Marquardt algorithm.

The stochastic algorithms, on the other hand, are characterized by performing global searches and by implementing probabilistic search operators. Among them, we highlight the use of GAs (Montana (1989)) and the Simulated Annealing algorithm (Masters (1995)) in the ANN training. These algorithms determine a new point in the search space by random operations applied to the current weights, most commonly deploying the Gaussian or the Cauchy perturbation (Yao (1995)). The following points in the search space to be examined are selected also on the basis of probabilistic criteria. Due to these characteristics, the stochastic algorithms are less vulnerable to the local minima problem. Another advantage of these algorithms is that the cost function does not need to be differentiable, since it does not use gradient information.

Despite the above-cited advantages, the stochastic algorithms are slower than the faster versions of gradient based techniques (Yao (1995)). Besides, although they are efficient to run global searches, they are less efficient to undergo more refined local searches. A very promising approach emerges in this scenario: the hybrid training, which combines the advantages of both deterministic and stochastic approaches. In the hybrid training, the ANN's weights are alternately modified by a deterministic and by a stochastic algorithm. The stochastic algorithm defines the initial weights used by a local deterministic algorithm to proceed a more fine-tuned local search. This is, in fact, an alternative approach to the random restart of the local algorithm. The ideal behavior of the hybrid algorithm should present the computational efficiency of the gradient-based techniques with the capacity of the stochastic techniques in avoiding local minima.

In (Belew et al. (1990)), the authors combined GAs to the BP algorithm obtaining better results than the use of each algorithm in isolation. In (Masters (1995)), the authors combined the Simulated Annealing and Conjugate Gradient. Other applications of hybrid training algorithms can be found in (Kinnebrock (1994)) and (Castillo et al. (2000)).

3 Hybrid GA-LM algorithm

In this work, we adopted the hybrid learning approach described above using Genetic Algorithms to define the initial weights used as input by the Levenberg-Marquardt algorithm. Hence, we use Genetic Algorithms as the global search procedure and the LM algorithm as the local search procedure. The choice for the LM algorithm is due to its better efficiency (in comparison to other gradient-based algorithms) when dealing with small ANNs, which is the case of the deployed TDNN. The choice of GA as the global procedure is due to its capability of evaluating multiple points in the search space at same time. Hence, they are less sensitive to fall in local minima (Goldberg (1989)).

In our hybrid algorithm, each GA chromosome stores a weight configuration that serves as a starting point for the LM algorithm. In each LM training, the algorithm runs until one of these criteria is verified: maximum number of 500 iterations, generalization loss of 10% or stopping in the training progress of 5%. Details of these stopping criteria can be found in (Prechelt (1994)). The GA's fitness function evaluates a chromosome through the MSE (Mean Squared Error) on the validation data after the LM training. This measure was chosen because it estimates the generalization performance of the trained network (Bishop (1995)).

Regarding the weights' representation, two main approaches must be considered: binary and real schemes. In the former, the value of each weight is stored in a string of bits with a fixed length, and the chromosome is formed by the concatenation of all strings. This scheme has problems of scalability and the precision of the results may not be satisfactory since it is limited by the number of bits in the initial string. The real scheme does not suffer from this drawback since the weights are represented by real numbers. For this reason, we opted to use the real representation in our algorithm.

Considering the genetic operators, it is possible to identify a problem when defining the crossover operator. Two ANNs can be functionally equivalent but bear considerably different genotypes, which makes it difficult the generation of children with the same fitness. This is the case, for instance, of two ANNs with the same weights however in different order. This problem is known as the permutation problem (Hancock (1992)). Although some authors suggest that this is not a difficult problem, there is still a lot to be investigated (Yao (1995)). As we are not convinced that the permutation problem effects are not severe, we opted to discard the crossover operator. Therefore, the only operator adopted in our work was the Gaussian (or Normal) mutation, where a small number obtained from a $N(0, \sigma^2)$ distribution is added to the current value of the weight. The standard deviation σ is a parameter of the operator.

We observe here that, in each execution of the hybrid algorithm, if the GA is set up to perform g generations with i chromosomes per generation, the number of weight configurations generated by the GA, and consequently the number of LM executions, is equal to i multiplied by g. The hybrid algorithm returns the trained weights with the lowest validation error, among the i x g configurations of trained weights.

4 Case study: river flow prediction

As case study, we trained a TDNN which was used to forecast the monthly river flow of a hydrographic reservoir. The relevance of working with this problem comes from the fact that the operation planning of a hydroelectric power station can be improved based on its flow forecasting system, since the latter reveals the reservoir's energetic potential. Among other works that applied ANNs to the problem of river flow prediction, we highlight (Kadowaki et al. (1997)), (Valenca (1999)) and (Prudêncio and Ludermir (2001a)).

The input data used in our work was obtained from the Guarapiranga reservoir, which is part of the Brazilian Electrical Sector. We have available 300 flow values from January 1963 to December 1986 (Valenca (1999)). This series was equally divided into training, validation and test data sets. The TDNN used in our experiments has a time-window of length 12, receiving the last 12 months flow, and 5 units in the hidden layer. This network employs the one-step prediction of the river flow based on the last 12 months.

The work presented here was developed in the context of a main work which proposes the use of Case-Initialized Genetic Algorithms (Grefensttete and Ramsey (1993)) to implement a system to design neural networks for time series prediction (Prudêncio and Ludermir (2001b)). The case-initialization of GAs consists of generating the first GA's population from well-succeeded solutions to problems which are similar to the one being tackled. The inspiration of this technique comes from the fact that similar problems have

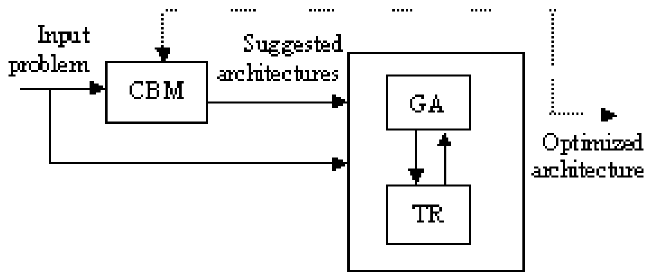

Fig. 1. NN optimization model using Case-Initialized GAs.

similar search spaces and, therefore, good solutions to a particular problem can provide information about the search space for similar problems.

Figure 1 shows the architecture of an implemented prototype. The CBM module maintains a case base in which each case associates a time series to a well-succeeded network architecture used to predict it. Given a new problem (new time series), the Case-Base Manager (CBM) module retrieves a predefined number of cases, selected on the basis of their similarity to the input problem. The Genetic Algorithm (GA) module performs an optimization of the ANN architecture, using as the initial population the architectures retrieved by the CBM module. The Training module (TR) is responsible for training the weights of each architecture. The output network will be the best architecture generated by the GA and trained by TR. Following, a new case is created and inserted in the base, in order to suggest more adequate solutions in the future. The preliminary results of this prototype for the river flow prediction problem were very promising (Prudêncio and Ludermir (2001a)).

The TR module actually implements the randomly initialized Levenberg-Marquardt algorithm. However, as the proposal of the optimization model is to develop a general system which works well in situations with different needs and constraints, we should experiment different training algorithms and choose the more robust one. The proposal of GA-LM hybrid algorithm is a tentative of developing such robust algorithm.

5 Experiments and results

In our experiments, four procedures were compared in the learning of the TDNN described above: RAND-BP(n), RAND-LM(n), GA-BP(i,g) and GA-LM(i,g). The former two procedures correspond to the usual training process where the gradient-based algorithm is executed n times with random initial weights and the configuration of trained weights with the lowest validation error is returned. RAND-BP(n) and RAND-LM(n) used the BP and the LM algorithms respectively. The procedure GA-BP(i,g) corresponds to a hybrid algorithm in which the GAs are used as the global stochastic algorithm and the BP is the local gradient-based algorithm. And finally, the procedure GA-

LM(i,g) corresponds to the proposed algorithm where the LM is combined with GAs.

The random procedures were run for two different values of n (number of weight restarts): 100 and 200. To proceed with a fair comparison, we setup the parameters i and g in such a way that the procedures with GAs implement the same number of restarts as the random procedures. Hence, for $n = 100$ restarts, we used $i = 10$ and g =10, and for $n = 200$, we used $i = 10$ and $g = 20$. The values for mutation rate and the standard deviation of the Gaussian perturbation were set to 0.1 and 0.3 respectively. These values were chosen in preliminary tests.

In order to compare the performance of these procedures, we ran each of them 10 times, and calculated the average values of the MSE in the validation and test sets, as well as, the average execution time.

Procedure (100 restarts)	AVG MSE Validation	Procedure (200 restarts)	AVG MSE Validation
GA-BP(10x10)	34.45	GA-BP(10x20)	32.96
GA-LM(10x10)	34.68	GA-LM(10x20)	34.04
RAND-BP(100)	35.01	RAND-BP(200)	34.82
RAND-LM(100)	35.41	RAND-LM(200)	35.51

Table 1. Ranking based on the Average MSE of the validation set.

In Table 1, we presented the ranking of algorithms for 100 and 200 restarts in terms of the MSE in the validation set, which measured the ANN training performance. As we can see, the performance of the algorithms were very similar for both numbers of restarts. We could say that the procedures are statistically equivalent with high confidence. Despite this similarity, the proposed hybrid algorithm obtained good results in this measure for both number of restarts (2nd place in the ranking of algorithms). Another observation is that, for both BP and LM algorithms, the use of GA initialization improved the results, compared to the random procedures.

In Table 2, we presented the results in terms of the MSE in the test set. As we can see, the proposed hybrid algorithm presented an intermediate ranking in this measure (2nd place for $n = 100$ and 3rd place for $n = 200$). We observed that the ranking of the GA procedures for the test set was, in general, worse than the ranking obtained for the validation set, particularly for $n = 200$. This effect is more drastic in the GA-BP procedure, which obtained the best validation error, however, the worst test error. This observation may indicate that the use of GA initialization led to an overfitting of the validation data, however this conclusion must be better investigated. Other papers about hybrid learning did not mention this effect.

Procedure (100 restarts)	AVG MSE Test	Procedure (200 restarts)	AVG MSE Test
RAND-LM(100)	34.41	RAND-LM(200)	35.63
GA-LM(10x10)	36.17	RAND-BP(100)	36.07
RAND-BP(100)	36.24	GA-LM(10x20)	36.80
GA-BP(10x10)	36.73	GA-BP(10x20)	37.53

Table 2. Ranking based on the Average MSE of the test set.

Procedure (100 restarts)	AVG Execution Time	Procedure (200 restarts)	AVG Execution Time
GA-LM(10x10)	2.4 min	GA-LM(10x20)	4.7 min
GA-BP(10x10)	6.7 min	RAND-LM(200)	12.0 min
RAND-LM(100)	6.8 min	GA-BP(10x20)	13.5 min
RAND-BP(100)	9.7 min	RAND-BP(200)	19.2 min

Table 3. Ranking based on the Average Execution Time (minutes).

The most significant result of these experiments appears when we examined the execution time obtained for each of the learning procedures (see Table 3). As we can see, our proposed algorithm was the fastest algorithm, reducing significantly the execution time compared to the other procedures. This improvement was at least 64% for 100 restarts, and 60% for 200 restarts. This characteristic is of main importance for applications with strong time constraints. The good performance of our hybrid algorithm in this point can be explained by the fact that, at each GA generation, the LM algorithm takes advantage of the optimization performed in the previous generation. Hence, it only refines the search that has already been hardly performed in the previous generations. Furthermore, the LM algorithm is a second-order algorithm, which is, in general, faster than the Backpropagation (Masters (1995)).

6 Conclusion

This paper presents the implementation of a hybrid learning algorithm for ANNs where the GAs were responsible for selecting the initial weights for the Levenberg-Marquardt algorithm. This algorithm was compared to different combinations of initialization strategies and gradient-based techniques on the training of a TDNN for a river flow prediction problem.

Based on the undergone experiments, we concluded that the GA-LM algorithm obtained good results in terms of quality of prediction and showed excellent results in terms of time execution. Despite these good results, the use of GA's initialization may sometimes led to overfitting, although the GA-LM algorithm was less sensitive to this problem. These results lead us to perform more accurate experiments with GA-LM algorithm in order to use it in our general system to design neural networks (as presented in section 4). Eventually, we will test other stochastic techniques.

Another point to investigate in the future is how to avoid the problem of overfitting in hybrid algorithms, evaluating in which situations a hybrid algorithm can be applied with a lower risk of affecting the generalization performance. It is also necessary, to undergo a more detailed study about the influence of the GA's parameters on the performance of the NN training.

References

BARNARD, E. and COLE R.A. (1989): A Neural-Net Training Program based on Conjugate-Gradient Optimization. *Technical Report CSE-89-014, Oregon Graduate Institute, Beaverton, OR.*

BATTITI, R. (1992): First and Second-Order Methods for Learning Between Steepest Descentand Newton's Method. *Neural Computation, 4, 141–166.*

BELEW, R.K., MCINERNEY, J. and SCHRAUDOLPH, N.N. (1990): Evolving Networks: Using the Genetic Algorithm with Connectionist Learning. *Technical Report CSE-CS-90-174, University of California, San Diego, CA.*

BISHOP, C. (1995): *Neural Networks for Pattern Recognition.* Oxford University Press, UK.

CASTILLO, P.A., CARPIO, J., MERELO, J. J., RIVAS, V., ROBER, G. and PRIETO, A. (2000): Evolving Multilayer Perceptrons. *Neural Processing Letters, 12, 115–127.*

GOLDBERG, D.E. (1989): *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA.

GREFENSTTETE, J. and RAMSEY, C. (1993): Case-based initialization of genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, CA, 84–91.*

HANCOCK, P.J.B. (1992): Genetic Algorithms and Permutation Problems: a Comparison of Recombination Operators for Neural Net Structure Specification. *Proceedings of the International Workshop on Combinations of Genetic Algorithms, Baltimore, 108–122.* IEEE Computer Society Press, CA.

HAYKIN, S. (1994): *Neural Networks: A Comprehensive Foundation.* IEEE Press/Macmillan College Publishing Company, New York, NY.

KADOWAKI, M., SOARES, S. and ANDRADE, M. (1997): Montly River Flow Prediction Using Multilayer Neural Networks with Backpropagation. *Anais do IV Simpsio Brasileiro de Redes Neurais (in portuguese), 32–35, Goiana, Brazil.*

KINNEBROCK, W. (1994) Accelerating the Standard Backpropagation Method Using a Genetic Approach. *Neurocomputing, 6, 583–588.*

LANG, K and HINTON, G. (1988): Time-Delay Neural Network Architecture for Speech Recognition. *Technical Report CMU-CS-88-152, Carnegie-Mellon University, Pittsburgh, PA.*

LEVENBERG, K. (1944): A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly Journal of Applied Mathematics, 2, 164–168.*

MARQUARDT, D. (1963): An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM J. Applied Mathematics, 11, 431–441.*

MASTERS, T. (1995): *Advanced Algorithms for Neural Networks: a C++ Sourcebook.* John Wiley & Sons, New York, NY.

MONTANA, D. J. and DAVIS, L. (1989): Training Feedforward Neural Networks Using Genetic Algorithms. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, 762–767.*

PRECHELT, L. (1994): Proben1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. *Technical Report 21/94, Fakultat fur Informatik, Universitat Karlsruhe, Germany, September, 1994.*

PRUDÊNCIO, R.B.C. and LUDERMIR, T.B. (2001a): Evolutionary Design of Neural Networks: Aplication to River Flow Prediction. *Proceedings of the International Conference on Artificial Inteligence and Aplications, AIA 2001, Marbella, Spain, 56–61*

PRUDÊNCIO, R.B.C. and LUDERMIR, T.B. (2001b): Design of Neural Networks for Time Series Prediction Using Case-Initialized Genetic Algorithms. *Proceedings of the 8th International Conference on Neural Information Processing, ICONIP' 01, Shanghai, China, 990–995.*

RUMELHART, D.E., HINTON, G.E. and WILLIAMS, R.J. (1986): Learning internal representations by error propagation. In: D.E. RUMELHART and J.L. MCCLELLAND (Eds.): *Parallel Distributed Processing Vol.1.* MIT Press, Cambridge, 318–362.

VALENCA, M.J.S. (1999): Analysis and Design of the Constructive Neural Networks for Complex Systems Modeling. *Ph.D Thesis (in portuguese), Federal University of Pernambuco, Recife, Brazil.*

YAO, X. (1995): Evolutionary Artificial Neural Networks. *Encyclopedia of Computer Science and Technology, 33, 137–170.* Marcel Dekker, New York, NY.