A LAMARCKIAN MODEL COMBINING LEVENBERG-MARQUARDT ALGORITHM AND A GENETIC ALGORITHM

Paulo Pires – paulo.pires@upt.pt
Universidade Portucalense – Departamento de Gestão
Rua Dr. António Bernardino de Almeida, 541
4200-072 Porto – Portugal

Pedro Castro – pmc@estg.ipvc.pt
Escola Superior de Tecnologia e Gestão do Instituto
Politécnico de Viana do Castelo – Dep. de Informática
Av. do Atlântico – 4900-348 Viana do Castelo – Portugal

ABSTRACT

We review the integration between the genetic and evolutionary techniques with artificial neural networks. A Lamarckian model is proposed based on genetic algorithms and artificial neural networks. The genetic algorithm evolves the population while the artificial neural network performs the learning process. The direct encoding scheme was used. This model was submitted to several data sets and provided good results, exhibiting superior robustness when compared with the Levenberg-Marquardt and the Scaled Conjugate Gradient algorithms. It also achieved the best solutions in the regression problems.

1. Introduction

The genetic and evolutionary computation comprises a group of techniques of which is part the genetic algorithms, the evolutionary strategies, the genetic programming and the evolutionary programming (Cortez et. al, 2002). As the author states, there isn't a clear division and exists an overlap of those techniques. Of the above referred techniques the genetic algorithms (GA) have been proving that they are robust and efficient methods in the resolution of optimization problems. The GA, such as artificial neural networks (ANN), had been also inspired by biological phenomena, being in this case an analogy to the mechanisms of the evolution and the natural selection. The application of GA as an optimization technique of an ANN has been very promising. The literature review reveals that the optimization of an ANN with GA produces good results comparatively with other algorithms, such as the Backpropagation (BP) algorithm and their variants. In this paper we review the possible interactions between GA and ANN, and further investigate the Lamarckian approach using the Levenberg-Marquardt algorithm with a variation of the GA that uses a best individuals' strategy.

According to Gruau (1994) the interaction between GA and ANN can be reduced to the resolution of the right codification of the ANN's architecture in chromosomes able to be manipulated by the GA. The author groups the alternatives in three schemes: direct encoding; parametric encoding; indirect encoding. In direct encoding the weights matrices are encoded directly in the chromosomes. In the parametric encoding a set of parameters is encoded in the chromosome, such as the number of layers, the number of neurons of each layer and the type of connections among each layer. In the indirect encoding the chromosome holds a grammar with a certain definition, which allows the generation of families of ANN.

This classification differs considerably from (Whitley, 1995), where the GA have been applied to ANN following three strategies. The first strategy is equivalent to direct encoding, but allows the learning rate inclusion in the chromosome. The second strategy includes the parametric and grammatical encode and it is designated by evolutionary strategy of ANN's architectures. The third strategy applies the GA in the selection of the learning vectors and in the interpretation of the output variables.

For Grönroos (1998) and Koehn (1994) GA is applied following direct or indirect encoding strategies. But Yao (1995) divides the strategies in weights adjustment, architecture evolution and evolution of learning rules. The first two strategies are associated respectively to direct and indirect encoding.

Another classification made by Curran and O'Riordan (2002) identifies five possible strategies of integrating GA and ANN: weights adjustment of the connections maintaining a fixed architecture; architecture evolution and weights adjustment with a learning algorithm; identification of transfer functions; evolution of learning rules; combination of several strategies. The author also makes the explicit division between direct and indirect encoding.

From the literature review one can group the codification in direct, indirect or in a combination of the two. In direct encoding a GA chromosome is composed by genes, which represents the ANN weights. Other parameters, such as the learning rate or momentum coefficient may also be included. With the direct encoding scheme the algorithm that minimizes the error function is replaced by the GA. The GA maintains a population of chromosomes, where each chromosome contains all the ANN parameters. Usually the ANN architecture remains fixed during the learning phase and the chromosome can use binary or real coding of the genotype. Therefore it is feasible to use the conventional genetic operators. With

the indirect encoding scheme the ANN architecture is encoded in the chromosome. In fact it is not the architecture itself that is encoded, but the production rules that produce an outcome that originate it. The consecutive application of the genetic operators evolves the ANN's architecture. Generally, in indirect encoding the weights are not encoded in the chromosome. If the problem requires variables that have real domains, then it is necessary to apply a learning algorithm, which can be one of the fastest variants of the BP algorithm. When we combine the direct and indirect encoding schemes the ANN's architecture doesn't remain fixed during the learning phase. The weights are also adjusted using GA, genetic programming or other techniques. The chromosome has to include the information about the architecture and the weights.

According to Whitley (1995) there are two main factors for the avoidance of direct encoding: first, the BP algorithm variants are very efficient in supervised learning of ANN; second, the learning process of an ANN represents a problem that is not well suited for the standard GA, since it is a multimodal optimization problem, usually named *Permutations Problem* or *Competing Conventions Problem*.

In spite of being controversial, the direct encoding scheme continues to be applied with success. Example of that is the study of Seiffert (2001) and Falco et al. (1997), where the authors state that GA are more effective in the search of the global optimum than the BP algorithm. In fact GA are global search methods, while the BP algorithm and its variants are local search methods.

Another perspective in GA and ANN integration is described by Sasaki and Tokoro (1997, 1999) and Rocha et al. (2003). In this approach, illustrated in figure 1, the concept of learning, which occurs in each individual – ANN – and the concept of evolution, which takes place in consecutive generations – GA – are combined, resulting in what is called the Lamarckian evolution. In Lamarckian evolution a GA and a learning algorithm are iteratively applied. The chromosomes resulting from the GA population evolution are decoded and submitted to a learning algorithm and then the improved results are encoded back into GA individuals. Lamarckian evolution remains as a main research subject in the field of neural networks.

According to Ku and Mak (1997, 1998) and Ku et al. (2000) there are two alternatives to embedding learning in the evolutionary search: Baldwin and Lamarckian. Although the Baldwin effect is more biological plausible, they achieved better and faster results with Lamarckian evolution in weights optimization of recurrent ANN. The Lamarckian approach was compared with the gradient descent and Baldwinian learning achieving a better convergence. Lamarckian evolution can also be implemented in Hopfield networks (Imada and Araki 1996). The method iteratively applies the Hebbian learning and the genetic operators to optimize the weights. The results prove that Lamarckian evolution can improve pattern storage capacity of the network.

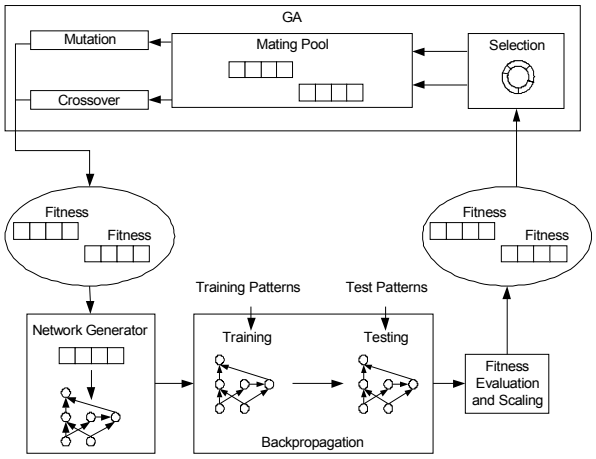

Figure 1 – Integration of GA and ANN (adapted from (Mandischer, 1993)).

In (Hakkarainen et al. 1996) and (Kyngäs and Hakkarainen 1996) is explained a Lamarckian approach that evolves a set of randomly initialized ANN. Those ANN are trained with a BP algorithm during a number of epochs and thereafter the worst ANN are removed. The genetic operators are then applied and training proceeds with the new ANN. This method was used in the sunspot prediction problem and though it got good results, the authors state that the method is computationally intensive.

In (Cortez et al. 2001) is described an evolutionary approach that evolves a set of neural networks. Each individual in the population encodes an ANN topology and training is done with the RPROP algorithm. The genetic operators recombine the individuals producing new topologies, allowing the system to find the best topology for a given problem. In the evaluation forecasting problems, the model gave good results.

Rocha et al. (2003) have evaluated the performance of Lamarckian and Baldwinian models and have concluded that Lamarckian is superior in static environments while the Baldwinian is superior in dynamic environments. Those finds do confirm the ones obtained by Sasaki and Tokoro (1997) that Lamarckian is efficient in static environments, performing poorly and being unstable in dynamic environments.

More studies with Lamarckian models can be found in (Cortez et al., 2002, Rocha et al., 2000) where Lamarckian achieved good results in machine learning benchmarks.

2. Conceptual model

Based on previous good results shown by Lamarckian evolution we propose a model using a faster version of the BP algorithm, the Levenberg-Maquardt algorithm and a version of GA that uses a best individuals' strategy.

The LM algorithm (Levenberg-Marquardt), described in (Hagan and Menhaj, 1994) is a variant of Newton's method. Although Newton's method requires the calculation of the Hessian matrix, the LM algorithm is based in the Gauss-Newton method, avoiding that calculation. The LM algorithm approximates the Hessian matrix through the Jacobian matrix. The LM algorithm can only be applied with the mean square error function (Bishop, 1995). The values of the connections are adjusted following the equation:

$$W\{k+1\} = W\{k\} - \left(J^t\{k\} J\{k\} + \mu\{k\} I\right)^{-1} J^t\{k\} e\{k\}$$

Where $W\{k\}$ is the weights matrix in epoch k, $J\{k\}$ is the Jacobian matrix in epoch k and $e\{k\}$ is the error in epoch k.

In the proposed model, the GA uses a strategy that guarantees best individuals found so far over generations are never lost. This is achieved by selecting the best solutions from parents and offspring populations to create a new population for the next generation. This scheme guarantees best solutions are never lost and a faster convergence of the GA. Also, repeated individuals are not allowed during the creation of offspring to maintain population diversity.

The conceptual model of the GA and ANN integration is formalized in figure 2. In the presented model, the GA starts with a population of 50 individuals, each representing a feedforward architecture. The population evolves during 100 generations. The GA use stochastic universal sampling selection to select the individuals that participate in the creation of new solutions (Baker, 1987), uniform crossover and mutation is implemented by flipping each chromosome bit with a probability equals to the inverse of the chromosome length. The direct encoding scheme was used, meaning that a chromosome holds a sequence of genes, each representing a weight value. Each connection was encoded using 29 bits. The Gray code was employed.

After the evolution phase, the chromosomes are decoded into feedforward architectures with the respective weights. Then the model enters in a cycle. The learning phase is accomplished with the LM algorithm and is divided in two stages. The first stage comprises the training of all the 50 networks during 2 epochs and thereafter the best network is chosen. The best network is then trained until it reaches convergence or it reaches a predefined number of epochs. If the network does not converge, then all the trained networks and their weights are encoded back and the GA evolves the population once more. In this phase, the GA evolves the population during 50 generations. This cycle is repeated until a solution is obtained or the number of cycles is exhausted.

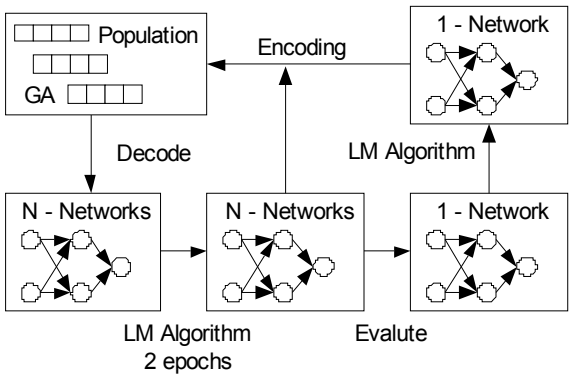

Figure 2 – Lamarckian evolution proposed model.

3. Benchmarking setup

To evaluate the proposed model we selected four well known data sets. These data sets can further be divided in two classification and two regression problems. All the algorithms were tested in the MathWorks Matlab version 6.5 release 13 with Neural Network Toolbox 4.0.1 and the Genetic Algorithm toolbox was implemented in Delphi 5.0.

In all the experiments the data was pre-processed, with the inputs and targets normalized so that they have zero mean and unit standard deviation. Each data set was divided in two parts: the training set (75% of training pairs) and the test set (25% of training pairs). All the algorithms performed 30 runs and the number of epochs was restricted to a maximum of 1000. The parameters of each algorithm were properly tuned. The weights initialization in the BP algorithms and its variants followed the Nguyen-Widrow initialization algorithm (Nguyen and Widrow, 1989).

The choice for the number of neurons is a cumbersome task. There are some heuristics that provide an approximate value, but in fact they don't work in many problems. It should be noted that an ANN with fewer neurons than necessary will lead to underfitting and an ANN with more neurons than necessary will lead to overfitting. To avoid overfitting, one can use statistics methods and heuristics such as early stopping, regularization, weight decay, cross validation and Bayesian regularization and model selection (Precheltz, 1997, Bishop, 1995, Sarle, 1995, Sigurdsson et al., 2000, Larsen and Hansen, 1995, Goutte and Larsen, 1998,

Larsen et al., 1998). In this study, we use cross-validation, regularization and model selection.

To evaluate the efficiency of the algorithms in regression problems the following measure was established: R_{Test} – R regression value of the test data set. The R-values are obtained from a linear regression between the network outputs and the targets, corresponding to the correlation coefficient between two vectors. The R_{Test} gives the ANN ability to predict values when facing new data. From the R_{Test} different values, the most interesting is the R_{Test} max, which is the maximum R value obtained from a linear regression between the network outputs and the corresponding targets in the 30 runs. A perfect fit would have a value of 100%.

In the classification problems the following measure was established: P_{ct} – percentage of test pairs classified correctly. The classification matrix was also useful to evaluate the performance of the algorithms. We also used the R_{Test} to assess the classification performance of the algorithms.

The test data set is never used during training. This part of the data set is only used when the learning process has finished.

In order to assess the Lamarckian Approach– LA – performance the same problems were applied with Scaled Conjugate Gradient backpropagation – SCG, (Möller, 1990) and the LM backpropagation.

The first data set is the two spiral problem – classification problem – with 768 training pairs and was applied in a 2-15-10-5-1 architecture (figure 3). The second data set, provided by Marko Bohanec, is known as the Car Evaluation Database and it is also a classification problem. Is has 1728 training pairs and was applied in a network with a 6-15-1 architecture. The third and fourth data sets were provided by Frank Smieja from the Adaptive Systems Research Group of the GMD and according to the author these approximation tasks are very difficult to solve for neural networks with no-local activation functions. The third data set is a regression problem with 400 training pairs which was tested in a 2-15-1 architecture. The fourth data set is also a regression problem and was applied in a 2-15-1 architecture. This data set has also 400 training pairs.

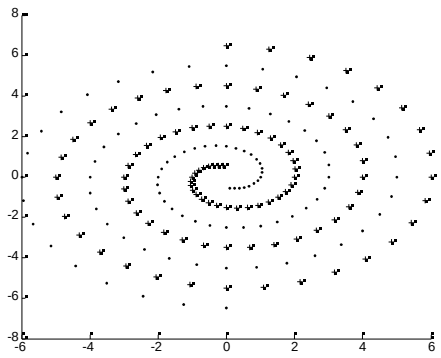

Figure 3 – The two spiral representation.

5. Benchmarking results

The following tables contain the results of the evaluations performed in the several problems.

Table 1 – Evaluating the test set in the two spiral problems

	P_{ct}			R_{Test}		
Algorithm	aver	max	min	aver	max	min
SCG	98.4%	100%	97.3%	97.4%	97.8%	97.1%
LM	98.8%	100%	96.8%	97.9%	98.6%	97.1%
LA	99.8%	100%	99.7%	98.0%	98.9%	97.2%

Table 2 – Evaluating the test set in the car evaluation database

	P_{ct}			R_{Test}		
Algorithm	aver	max	min	aver	max	min
SCG	92.0%	95.4%	88.0%	93.0%	94.0%	91.2%
LM	92.1%	96.5%	83.8%	93.0%	95.5%	88.2%
LA	92.1%	93.5%	90.3%	93.0%	93.9%	91.4%

The best classification matrix achieved by the algorithms is presented in table 3.

Table 3 – Best classification matrix.

LA				LM				SCG			
292	12	0	0	294	6	0	0	298	7	0	0
8	80	4	0	6	81	3	0	2	76	3	0
0	7	13	5	0	12	14	3	0	16	12	4
0	0	1	10	0	0	1	12	0	0	3	11

Table 4 – Evaluating the test set in the Smieja functions

	R_{Test} – Smieja 1			R_{Test} – Smieja 2		
Algorithm	aver	max	min	aver	max	min
SCG	93.9%	94.0%	93.7%	94.9%	96.0%	93.5%
LM	94.4%	96.8%	90.0%	94.7%	96.9%	89.9%
LA	96.6%	97.7%	94.9%	97.7%	98.8%	97.0%

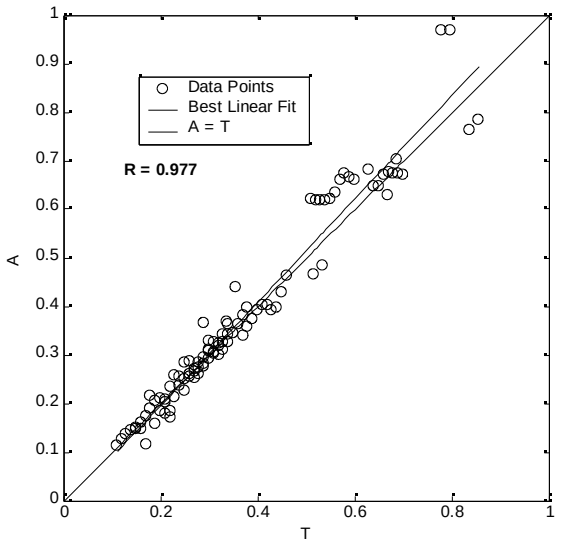

Figure 4 – R-test for Smieja 1 best LA solution.

In all the tests performed Lamarckian evolution was always much slower than the BP variants. This confirms the findings in literature review. The LM was always the fastest algorithm followed by the SCG. Nevertheless, the LM requires more computation resources, specially in terms of memory capacity.

In the two spiral problem all the algorithms behave similarly. They were able to classify correctly the input vectors of the test set at least once. The calculated average was almost 100%, proving that all the algorithms performed well. In spite of that Lamarckian evolution results were slightly better, with a higher average and a narrower interval, exhibiting a more robust behaviour.

Before analysing the results in the second data set we have to remark that this classification data set is not balanced. This means that there are more training pairs belonging to some classes than others. As we know this imposes some problems to the algorithms, since they will try to better fit the classes with more training pairs. In the car evaluation database the average classification was almost identical and the R average was equal to all the algorithms. In this test we have to stand out that both the LM and SCG reached higher results. They were able to classify more training pairs correctly than the LA. But the LA achieved a narrower interval, followed by the SCG and then by the LM. In fact, the LM gave the best and the worst results. In this data set it is also important to analyse the classification matrix. It is clear that all the algorithms favoured the classes with more training pairs and gave poorer results in the classes with lesser training pairs. It deserves also a remark that the run with the best P_{ct} isn't always the run which gives the best classification matrix. It may classify correctly more training pairs but it can introduce bias in some classes or as in this data set the classes with more training pairs are classified more accurately.

Results from the regression problems are presented in table 4. As an example, the linear regression between the network outputs and the targets and the correlation coefficient are presented in figure 4 for the best LA solution of Smieja 1 function.

In both regression problems, the LA was superior to LM and SCG. It achieved a higher average and a maximum R_{Test} and simultaneously a narrower interval.

An aggregate analysis of the four data sets results allows us to conclude the LA is a viable alternative to the backpropagation variants, namely the LM and SCG, although in the second data set it lacked the ability to obtain the maximum correct classification and the best classification matrix as was the LM. Nevertheless, the LA was more robust, producing a narrower interval. In the regression problems LA was always superior, providing a greater average compared with the others and, at least once, a run achieved a better maximum R. The LA gave once more a narrower interval.

Therefore, the results from the four data sets consistently demonstrated that LA is more robust than the LM and SCG. In practice, this means one can apply the LA once with a greater probability to obtain a good solution. Unfortunately, we cannot extrapolate the same findings to the achievement of the best solution, because it fails to do so in the second data set.

This finding raises an interesting question. LA is considerably slower than LM and SCG, but it probably requires a single run. On the contrary, it is well known from the literature that BP algorithms and its variants are dependable on the initial weight values. Therefore it would be interesting to evaluate only one execution of the slower LA vs. several executions of faster variations of BP algorithm.

4. Conclusions and future work

We proposed a Lamarckian model combining a GA based on a best individuals' strategy for the evolution process and a two phase Levenberg-Marquardt algorithm that performs the learning process. The GA addresses the global search and keeps the best individuals, but still maintains diversity. The proposed model is based on a direct encoding scheme.

This Lamarckian model was compared with the Levenberg-Marquardt backpropagation algorithm and with the Scaled Conjugate Gradient backpropagation algorithm. The Lamarckian model proved to be the most robust in all the tests and provided the best solutions in the regression problems.

As a result of this study some further research should be made. First, the LA effectiveness in one run should be measured in CPU time against the effectiveness of several runs of BP variants. This research will give new insights regarding the robustness of the algorithms and the required corresponding computer resources. Second, the Lamarckian model should be evaluated and extended to other data sets to verify the results obtained. Finally, the proposed Lamarckian model should be tested with other BP variants, such as QuickProp or even the Cascade Correlation, and with other variants of GA.

Acknowledgments

The authors would like to thank Sérgio Tinoco for its help in the text revision and the two referees for their helpful comments.

References

Baker, J. - Reducing bias and inefficiency in the selection algorithm. Genetic Algorithms and their Applications: Proceedings of the 2nd Int. Conf. on Genetic Algorithms, (1987), p. 14-21.

Bishop, Christopher M. - Neural networks for pattern recognition. Oxford New York: Clarendon Press; Oxford University Press, 1995. xvii, 482 p. ISBN 0198538642.

Cortez, P., Rocha, M. e Neves, J. - Evolving Time Series Forecasting Neural Network Models. Proceedings of International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models (ISAS 2001). Havana, Cuba, (2001), p. 84-91.

Cortez, P, Rocha, M. e Neves, J. - A Lamarckian Approach for Neural Network Training. Neural Processing Letters. 15:2, (2002), p. 105-116.

Curran, D. e O'Riordan, C - Applying Evolutionary Computation to Designing Neural Networks: A Study of the State of the Art. Departement of Information Technology, National University of Ireland, 2002. Technical Report NUIG-IT-111002.

Falco, I. De, Della, A., Natale, P. e Tarantino, E. - Artificial Neural Networks Optimization by means of Evolutionary Algorithms. Research Institute on Parallel Information Systems, Department of Mathematics and Applications, University of Napoles "Frederico II", 1997.

Goutte, Cyril e Larsen, Jan - Adaptive Regularization of Neural Networks using Conjugate Gradient. Proceedings of ICASSP'98. Seattle. 2, (1998), p. 1201-1204.

Grönroos, Markos A. - Evolutionary design of neural networks: University of Turku, Finland, 1998. Master Thesis.

Gruau, Frédéric - Neural networks synthesis using cellular encoding and the genetic algorithm. Lyon, France: Universite Claude Bernard-Lyon I, 1994. PhD Thesis.

Hagan, Martin T. e Menhaj, Mohammad B. - Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks. 5:6, (1994), p. 989-993.

Hakkarainen, J., Jumppanen, A., Kyngäs, J. e Kyyrö, J. - An evolutionary approach to neural network design applied to sunspot prediction. Department of Computer Science, University of Joensuu, 1996. 6 p. Technical Report.

Imada, Akira e Araki, Keijiro - Lamarckian Evolution of Associative Memory. Proceedings of IEEE International Conference on Evolutionary Computation, (1996), p. 676-680.

Koehn, Philipp - Combining genetic algorithms and neural networks: The encoding problem. Knoxville: University of Tennessee, 1994. Master Thesis.

Ku, K. W. e Mak, M. W. - Exploring the Effects of Lamarckian and Baldwinian Learning in Evolving Neural Networks. Int. Conf. on Evolutionary Computation (ICEC'97), (1997), p. 617-20.

Ku, K. W. e Mak, M. W. - Empirical Analysis of the Factors that Affect the Baldwin Effect. Fifth Internation Conference on Parallel Problem Solving from Nature (PPSN'98). Amsterdam, (1998), p. 481-490.

Ku, K. W., Mak, M. W. e Siu, W. C. - A Study of the Lamarckian Evolution of Recurrent Neural Networks. IEEE Transactions on Evolutionary Computation. 4:1, (2000), p. 31-42.

Kyngäs, J. e Hakkarainen, J. - Predicting sunspot numbers with evolutionary optimized neural networks. Proceedings of the Second Nordic Workshop on Genetic Algorithms and their Applications: Jarmo Alander, (1996), p. 173-180.

Larsen, Jan e Hansen, Lars Kai - Empirical Generalization Assessment of Neural Network Models. Proceedings of the IEEE Workshop on Neural Networks for Signal Processing V. Piscataway, New Jersey: IEEE, (1995), p. 30-39.

Larsen, Jan, Svarer, Claus, Andersen, Lars Nonboe e Hansen, Lars Kai - Adaptive Regularization in Neural Network Modeling. In G.B. Orr, K. Müller - Neural Networks: Tricks of the Trade, 1998. p. 113-132.

Mandischer, Martin - Representation and evolution of Neural Networks. Artificial Neural Nets and Genetic Algorithms Proceedings of the International Conference. Innsbruck, Austria: Springer, Wien and New York, (1993), p. 643-649.

Möller, Martin - A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Aarhus: Computer Science Department, University of Aarhus, Denmark, 1990. 21 p.

Nguyen, D e Widrow, B - Improving the Learning Speed of Two-Layer Neural Networks by Choosing Initial Values of Adaptive Weights. International Joint Conference on Neural Networks. San Diego, CA. III, (1989), p. 21-26.

Precheltz, Lutz - Early stopping - but when. Fakultät für Informatik; Universität Karlsruhe, 1997. Technical Report.

Rocha, M., Cortez, P. e Neves, J. - The Relationship between Learning and Evolution in Static and Dynamic Environments. Proceedings of Second International Symposium on Engineering of Intelligent Systems (EIS2000). Paisley, Scotland: ICSC Academic Press, (2000), p. 377-383.

Rocha, M., Cortez, P. e Neves, J. - Adaptive Learning in Changing Environments. 11th European Symposium on Artificial Neural Networks (ESANN'2003). Bruges, Belgium, (2003), p. 487-492.

Sarle, Warren S. - Stopped training and other remedies for overfitting. Proceedings of the 27th Symposium on Interface, (1995).

Sasaki, Takahiro e Tokoro, Mario - Adaptation toward Changing Environments: Why Darwinian in Nature? Fourth European Conference on Artificial Life: MIT Press, (1997).

Sasaki, Takahiro e Tokoro, Mario - Evolving Learnable Neural Networks under Changing Environments with Various Rates of Inheritance of Acquired Characters: Comparison between Darwinian and Lamarckian Evolution. Department of Computer Science, Faculty of Science and Technology, Keio University, 1999. 1-22 p. Technical Report.

Seiffert, Udo - Multiple Layer Perceptron Training Using Genetic Algorithms. Proceedings of the 9. European Symposium on Artificial Neural Networks ESANN 2001. Bruges, Belgium. ISBN 2-930307-01-3, (2001), p. 159-164.

Sigurdsson, Sigurdur, Larsen, Jan e Hansen, Lars Kai - On Comparison of Adaptive Regularization Methods. Proceedings of the IEEE Workshop on Neural Networks for Signal Processing X. Sydney: IEEE, (2000), p. 221-230.

Whitley, Darrell - Genetic Algorithms and Neural Networks. In Cuesta, G. Winter and J. Periaux and M. Galan and P. - Genetic Algorithms in Engineering and Computer Science. Chichester: John Wiley and Sons, Ltd., 1995. p. 203-216.

Yao, Xin - Evolutionary Artificial Neural Networks. In Williams, A. Kent and J. G. - Encyclopedia of Computer Science and Technology. New York: Marcel Dekker, 1995. p. 137-170.