Journal of Applied Sciences 10 (18): 2115-2120, 2010
ISSN 1812-5654
© 2010 Asian Network for Scientific Information

An Adaptive Time-delay Neural Network Training using Parallel Genetic Algorithms in Time-series Prediction and Classification

A. Ourdighi and A. Benyettou
Laboratory of Signal-Image-Speech, Department of Computer Science, Faculty of Science,
Mohamed Boudiaf University of Science and Technology of Oran, El Menaour, Oran, Algeria

Abstract: In this study, we present an Adaptive Time Delay Neural Network (ATNN) training based in parallel genetic algorithms and local search method in a comparative study which applied in several problems. Usually, the ATNN training used a temporal variant of gradient descent algorithm and universal approximation function and consequently inherits problematic of parameters initialization and traps into local minimum. Besides, the algorithm was based on the peculiarity to adapt not only the synaptic weight but also each delay of interconnection which singularizes the ATNN architecture. This adaptation paradigm offers more flexibility for the network to attain the optimal time-delays and to achieve more accurate pattern mapping and recognition than is the case of using arbitrary fixed delays, as has been done previously by Time Delay Neural Network (TDNN). Also, this principal provoked instability on converging process of gradient descent rules and affected the results. So, our aim is to replace discriminated method algorithm to stochastic approach, the training will be base on parallel genetic algorithms: multiple-deme parallel genetic algorithms. The important characteristics of multiple-deme parallel GAs are the use of a few relatively large subpopulations and migration. The model was tested on Time series prediction of Mackey-Glass a chaotic series and phonetic classification. Index Terms-Adaptive Time-Delay Neural Network (ATNN), Adaptable delay, Synaptic weight, Multiple-deme parallel genetic algorithms, local minimum, immigration.

Key words: Adaptive time-delay neural network, adaptable delay, synaptic weight, multiple-deme parallel genetic algorithms, local minimum, immigration

INTRODUCTION

The ATNN was proposed by Lin *et al.* (1992), Inspired of TDNN architecture, which appeared as an adequate proposition remedying the randomized initialization problem of the fixed delays. The resulting network is trained to distinguish the temporal properties and spatiotemporal correlations of various input patterns. The ATNN was used successfully in many works concerning the automatic targets recognition (Lin, 2004), aircraft target recognition (Huaitie *et al.*, 1997) as well as the phonetic classification (Ourdighi *et al.*, 2005).

In theory, this approach is based on estimation of universal approximation function and derived gradient descent algorithm, protecting the external temporal aspect, which was founded in its basic conception. However, this algorithm as any optimization discriminate method remains limited in the large-sized problems, where the probability to converge on a global minimum at reasonable time depends essentially on the good knowledge of the problem. In these cases, in lack of good knowledge, it is advised to turn to methods of stochastic optimizations.

By referring on the works (Spalanzani, 1999; Seiffert, 2001; Tlemsani *et al.*, 2005; Whitley, 1988; Kamp and Savenije, 2006), a Genetic Algorithms (GA) can be used as a learning device on Multi-Layers Perceptron (MLP). Motivated by what GA seems to be like a good alternative training solution, which allow cover important space research for convex problems in several local minimums, we going to eliminate the original learning algorithm and replace it by an evolutionary process.

In this study, the network training will be in charge of genetic algorithms in multiple populations. Each chromosome represents network structure proposition and their evolution into several population will provide the final optimal structure network according to optimal criteria. We tested a model in the chaotic series Mackey-Glass equation and compared the results with other methods. Also, we extended the experiment to phonetic classification based on TIMIT database.

Corresponding Author: Mohamed Boudiaf, Faculty of Science, Mohamed Boudiaf Sciences and Technology University, USTO, P.O. Box BP 1505, El Menaour, Oran, Algeria

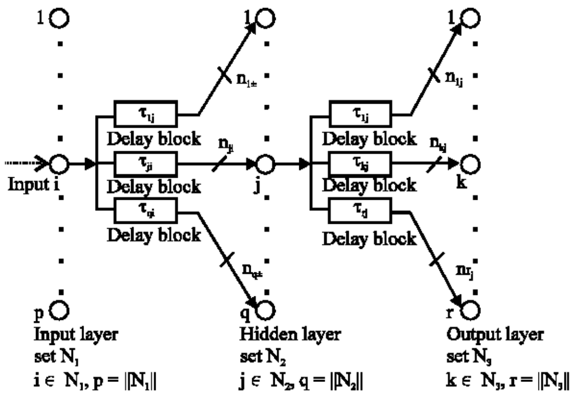

Fig. 1: An example of three layered ATNN

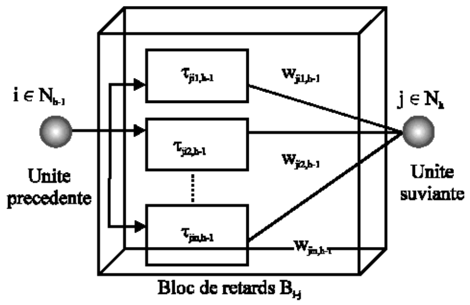

Fig. 2: Representation of bloc delay between two units

ATNN approach: With similar architecture of TDNN, an adaptive time delay neural network employs modifiable time delays along the interconnections between two processing units. For more illustration, Fig. 1 showed an example of three layered ATNN.

Each node j of layer h (e.g., j ∈ Nh), receives nji inputs from each i ∈ ij, h-1 (Fig. 2), where, Ij, h-1 is the subset of nodes on layer h-1 ($I_{j,\,h\text{-}1} \subseteq N_{h\text{-}1}$) that connects to unit j on layer h.

Time frame ji is a set of time delays (τ_{ji1}, ..., τ_{jinji}) employed on the connections to node j from node i. The time frame can have an additional index h ($\tau_{ji,h}$) to signify layer h. Node i of layer h is connected to node j of the next layer, with the connection line having an independent and modifiable time-delay $\tau_{jik,h}$ and weight $w_{jik,h}$. Each node sums the net inputs from the activation values of the previous neurons, through the corresponding time-delays on each connection line. Then the output of node j is governed by a symmetric sigmoid function f of the net input. The adaptation of both time delays and weight parameters is derived based on the gradient descent method to minimize the cost function E during training based on error back-propagation (Lin, 1994). The cost function is defined as the instantaneous mean square error measure of the desired output d_j and network output $a_{j,\,L}$ at time t_n as:

$$E(t_n) = \frac{1}{2}\sum_{j\in N_L}\left(d_j(t_n) - a_{j,L}(t_n)\right)^2$$

where, L denotes the output layer. So, it obvious that we have at least one criteria to optimize in training, which consent to integration of evolutionary algorithm possible.

Integration of genetic algorithm on ATNN training: There are various methods to integrate GA into the neural network. These methods get generally the network topology (e.g., hidden layers number or neurons number) or on training network by adapting a parameters (Branke, 1995; Castillo *et al*., 2003).

The unpredictable network parameters initialization is mainly the fault which limits the performance of gradient descent algorithm. In contrast, a GA assures a research in the complete domain. According to the generations, this research space is refined towards some various sub-spaces (Belew *et al*., 1990).

Genetic algorithms develop a set of candidate solutions, called a population, where a chromosome is other one than a possible solution of the problem to be resolved (Goldberg, 1988). Every chromosome of this population sees attributing a function called adaptation function or fitness which allows measuring its quality or its weight. Then, the best chromosome selected, undergo crossings and mutation and new population of solutions is produced for the following generation. This process continues, until sufficient fitness achieved or time delayed for example.

In parallel genetic algorithms using multiple populations, instead of using a single population, we build sets small sub-populations which evolve separately (Grosso, 1985; Cantu-Paz, 1998; Tinos and Yang, 2007). With this insulation, every subpopulation can evolve with its own parameters, in different directions, that are towards different solutions. Three different models exist: the global model, the diffusion model and the migration model (Yang, 2008).

Using in this neural integration the migration model, the principle is to divide the population in multiple subpopulations (Fig. 3). These subpopulations

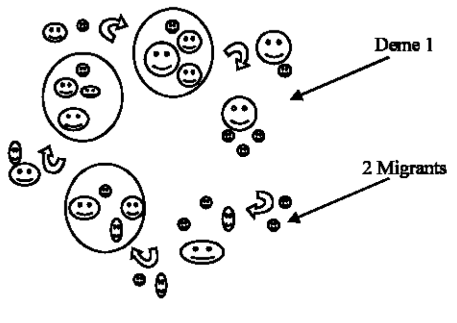

Fig. 3: Migration model

evolve independently from each other for a certain number of generations (isolation time). After the isolation time a number of individuals is distributed between the subpopulations (migration). The number of exchanged individuals (migration rate), the selection method of the individuals for migration and the scheme of migration determines how much genetic diversity can occur in the subpopulations and the exchange of information between subpopulations (Ahn *et al.*, 2004).

In this case of serial-time prediction, using of each subpopulation between 40 and 80 chromosomes, which are representing ATNN network structure, the chromosome is a serial of weight and delay of connections coded in binary code. The adaptation function will be error inverse 1/E.

Experimental results: A very interesting learning problem is prediction of a deterministic, but chaotic, time-series that we want to take as an application example of ATNN with parallel genetic algorithms training. The particular time series we choose is the time-series generated by Mackey-Glass equation. The function is generated by the following differential equation:

$$\frac{\partial x(t)}{\partial t} = a * x(t-\tau)/(1+x^{c}(t-\tau)) - bx(t) \qquad (1)$$

With: a = 0.2, b = 0.1, c = 10, t = 17.

For ATNN architecture, we used one neuron for input layer representing x(t) with 200 sequences to learning. Using receptive fields of four delays size with one shift to memorize signal, the three neurons of hidden layer used it to accumulate the activation of input neuron. Concerning out-put layer, with six receptive fields, we used one neuron to reproducing x(t+1). For prediction, the initial segment was fixed to 9 delays. This structure is used to define number of synaptic weight and delays for coding into chromosome. Finally, for parallel genetic algorithms, we used four subpopulations with different size between 40 and 80 genotypes. The migration frequency was fixed to 1 migration every 10 generations with individual number estimated at 5.

For a better demonstration of each models performance, after several experiments, the graphs shown below represent the worst graphic generations of the used networks.

First, we applied integration into TDNN with same parameters for more comparison and validation. The result was satisfied, like is shown in Fig. 4, comparing to TDNN training by descent gradient algorithm shown in Fig. 4a with error rate estimated at 0.451 after 1500 iterations, parallel genetic algorithms training shown in Fig. 4b given less error rate estimated at 0.04.

Figure 5 showed with more of details that error rate convergence was perfectly assured during training. Also, we could see that training by parallel genetic algorithms converged better than classic method.

After this trial, We experimented the training to ATNN for knowing if this integration would provided more stability for learning and parallel adaptation of parameters, consequently, if it would converged at better solution.

The results was very satisfied, as shown at Fig. 6a and b, the error attained 0.006 after only 507 iterations with the evolutionary algorithms training when it was estimated at 0,18 after 1500 iteration which is an optimization by 10^{-2} in the result. Also, we observed that TDNN which given less good result than ATNN using descent gradient algorithm, become more better with parallel genetic algorithms training.

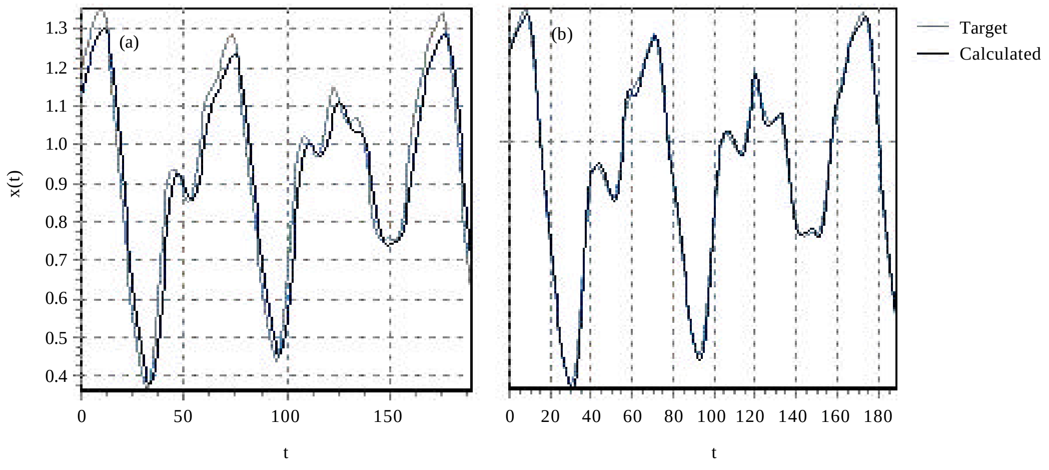

Fig. 4: Comparison of evolution results of x(t) using (a) TDNN with descent of gradient algorithm (b) and TDNN training with parallel GA after 1500 iterations

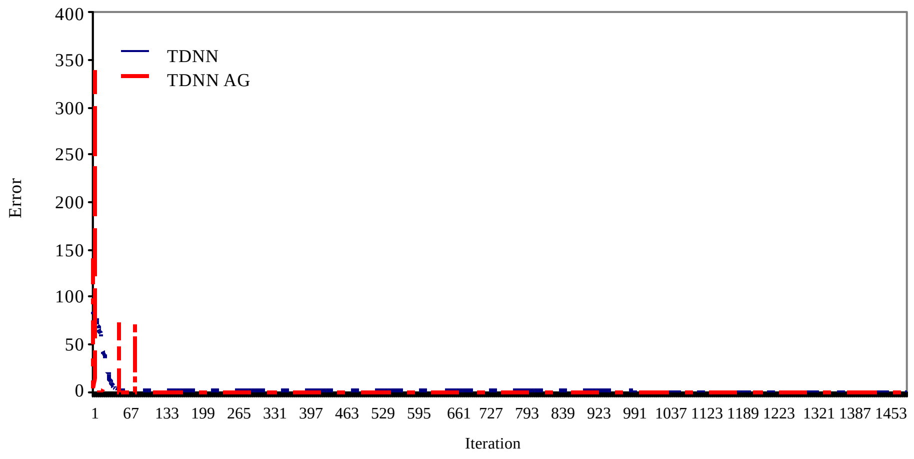

Fig. 5: Comparison of evolution error rate between simple TDNN et TDNN training with GA on 1500 iterations

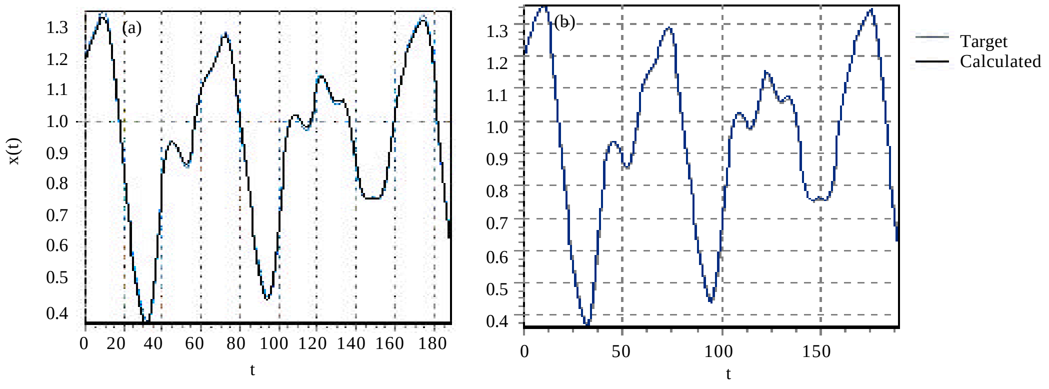

Fig. 6: Comparison of evolution results of x(t) using ATNN with descent of gradient algorithm (at left) and ATNN training with parallel GA (at right) after 1500 iterations

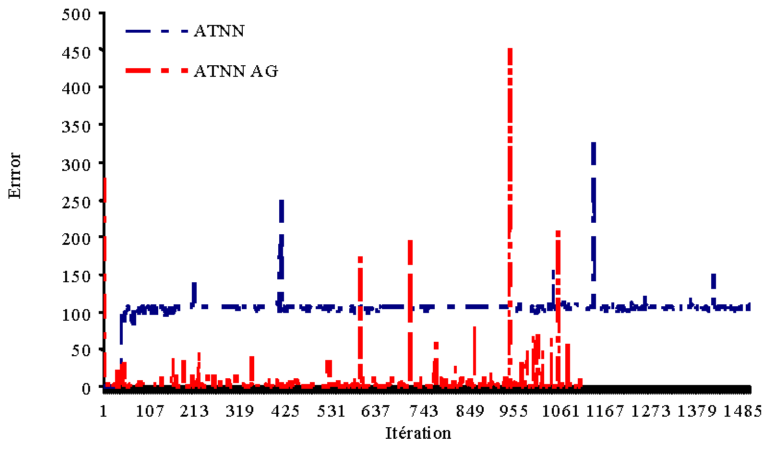

Fig. 7: Comparison of evolution error rate between simple ATNN et ATNN training with parallel GA on 1500 iterations

The error rate was less stable for parallel genetic algorithm but convergence was faster than descent of gradient algorithm, like is shown in Fig. 7.

Trying to extend this experimentation, we moved towards a phonetic classification. Of the TIMIT database, we have selected three phonemes of plosives categories.

Table 1: Results obtained by applying ATNN and ATNN with AG

	ATNN (%)		ATNN-AG (fitness 1) (%)		ATNN-AG (fitness 2) (%)	
Results	Train	Test	Train	Test	Train	Test
/b/	41.11	38.17	6.25	6.00	55.72	55.04
/d/	76.45	72.03	90.11	89.32	88.25	67.46
/g/	56.88	54.14	74.01	72.98	11.77	48.12
Total (%)	63.10	58.90	72.00	69.22	64.43	59.63

Table 2: Results obtained by applying ATNN training with AG with different subpopulations

	ATNN-AG (The same fitness) (%)		ATNN-AG (different fitness) (%)	
Results	Train	Test	Train	Test
/b/	0.00	0.00	50.12	23.12
/d/	91.75	14.95	69.29	32.96
/g/	73.97	89.60	66.41	65.20
Total (%)	72.32	45.15	65.59	42.90

However, the /b/, /d/ and /g/ are known for difficulty of separability (Weibel, 1988; Weibel *et al.*, 1989).

Unlike the first case, different fitness choices are offered. So, we could attribute for each subpopulation a fitness for varying the evolution. We could use the error function or the total train rate or the rate of some classes.

The result showed comparison of AG integration with different forms of fitness in several experimentations. We opted for using the rate or error.

For using AG with subpopulations, the results weren't so good. We noticed the speed of the convergence, but there was a decline in return on a generalization. The results obtained by simple GA integration with two different forms of fitness shown by Table 1 confirmed the efficient of the method to optimize the result where the rate augmented to 10% compare to the results of temporal gradient descent rules. In Table 2 with GA using subpopulations, the results moved back and were less encouraging in spite of the 2% of augmentation in training rate, the result shown issues.

CONCLUSION

The integration of multiple-deme parallel genetic algorithms in ATNN training finally strengthened its performance in a tangible way. In fact, we can really notice it on the results of the applications approached on our work, especially on the prediction.

We consider that the results obtained, in the classification case, are in the preliminary state and that our work amounts to a trial phase. We plan on one hand to continuous the experiment to the entire TIMIT database and on the other hand, to take into account the effect of the noise through NTIMIT database.

REFERENCES

Ahn, C.W., D.E. Goldberg and R.S. Ramakrishna, 2004. Multiple-deme parallel estimation of distribution algorithms: Basic framework and application. Proceedings of the Parallel Processing and Applied Mathematics, (PPAM'03), Springer-Verlag, pp: 544-551.

Belew, B., J. McInerney and N. Schraudolph, 1990. Evolving Networks: Using the Genetic Algorithms with Connectionist Learning. CSE Technical Report CS90-174, Computer Science, UCSD, California.

Branke, J., 1995. Evolutionary algorithms in neural network design and training: A review. Proceedings of the 1st Nordic Workshop on Genetic Algorithms and their Applications, (NWGAA'95), Finland, pp: 145-163.

Cantu-Paz, E., 1998. A survey of parallel genetic. Calculateurs Paralleles Reseaux Syst. Repartis, 10: 141-171.

Castillo, P.A., M.G. Arenas, J.J. Castillo-Valdivieso, J.J. Merelo, A. Prieto and G. Romero, 2003. Arti?cial Neural Networks Design Using Evolutionary Algorithms, Advances in Sof-Computing. Springer-Verlag New York.

Goldberg, D., 1988. Genetic Algorithms in Machine Learning. Optimization and Search, Addison-Wesley, USA.

Grosso, P.B., 1985. Computer Simulations of Genetic Adaptation: Parallel Subcomponent Interaction in a Multilocus Model. Doctoral Dissertation, The University of Michigan, University Microfilms No. 8520908, USA.

Huaitie, X., Z. Zhaowen and C. Zhenpin, 1997. Aircraft target recognition using adaptive time-delay neural network. Aerospace Electronics Conf. Proc. IEEE, 2: 764-768.

Kamp, R.G. and H.H.G. Savenije, 2006. Optimising training data for ANNs with genetic algorithms. Hydrol. Earth Syst. Sci., 10: 603-608.

Lin, D.T., J.E. Dayhoff and P.A. Ligomenides 1992. The adaptable time-delay neural network for temporal correlation and prediction. Proc. SPIE Intelligent Robots Comput. Vision XI, 1826: 170-170.

Lin, D.T., 1994. The adaptable time delay neural network characterisation and application to pattern recognition, prediction and signal processing, Ph.D. Thesis, Report, ISR.

Lin, D.T., 2004. Target components discrimination using adaptive time-delay neural network. J. Inform. Sci. Eng., 20: 959-980.

Ourdighi, A., Z. Ouksili, A. Benyettou, 2005. La reconnaissance des phonemes par les reseaux de neurones a delais temporels adaptatifs. Proceedings of the 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, March 27-31, TUNISIA, UK., pp: 1-6.

Seiffert, U., 2001. Multiple-layer perceptron training using genetic algorithms. Proceedings of the 9th European Symposium on Artificial Neural Networks, April 25-27, Australia, pp: 159-164.

Spalanzani, A., 1999. Algorithmes evolutionnaires pour l'etude de la robustesse des systemes de reconnaissance automatique de la parole. Ph.D. Thesis, University Grenoble.

Tinos, R. and S. Yang, 2007. A self-organizing random immigrants algorithm for dynamic optimization problems. Programming Evoluable Machines, 8: 255-286.

Tlemsani, R., N. Neggaz and A. Benyettou, 2005. Amelioration de l'apprentissage des reseaux de neurones par les algorithmes evolutionnaires. Proceedings of the 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, March 27-31, TUNISIA, UK., pp : 1-5.

Weibel A., T. Hanazawa, G. Hinton and K. Shinkano, 1989. Phoneme recognition using time-delay neural networks. IEEE Trans. ASSP, 37: 328-339.

Weibel, A., 1988. Consonant Recognition by Modular Construction of Large Phonetic Time Delay Neural Network. ATR Interpreting Telephony Research Laboratories, Osaka, Japan.

Whitley, D., 1988. Applying Genetic Algorithms to Neural Network Problems. International Neural Network Society, Churchill Livingstone, pp: 230.

Yang, S., 2008. Genetic algorithms with memory and estimed-based immigrants in dynamic environments. Evolutionary Computation, 16: 385-416.