PSO for Reservoir Computing Optimization

Anderson Tenório Sergio, Teresa Bernarda Ludermir

Center of Informatics (CIn), Federal University of Pernambuco (UFPE), P.O. Box 7851,
Cidade Universitaria, Cep: 50.740-530 - Recife - PE - Brazil
{ats3, tbl}@cin.ufpe.be
http://www.cin.ufpe.br

Abstract. Reservoir Computing is a new paradigm of artificial neural networks that has obtained promising results. However there are some disadvantages: the reservoir is created randomly and needs to be large enough to be able to capture all the features of the data. We propose a method to optimize the initial parameters using PSO – Particle Swarm Optimization. Our method took until 10.25% of the time required for exhaustive search and some approaches got no error statistically worse than others method tested for all databases used.

Keywords: reservoir computing, pso, optmization.

1 Introduction

PSO (Particle Swarm Optimization) is an optimization algorithm that has some advantages over other global search techniques. The algorithm is based on the social behavior of flocks of birds: a population of solutions is maintained and each individual seeks to improve its performance based on its best experience and the best experience of the group. In general, their operators are fast, its implementation is simple and its convergence is relatively fast.

In this work, we intend to show how the PSO and two extensions of this algorithm can work on Reservoir Computing optimization.

Reservoir Computing (RC) is a recent paradigm of Artificial Neural Networks. RC uses a similar architecture of Recurrent Neural Networks (RNN) for temporal processing without the need for training. RC was introduced parallel and independently as Liquid State Machine (LSM) [1] and Echo State Network (ESN) [2] a few years ago. In general, the concept of RC is based on building a random RNN (reservoir), without changing the weights. After this phase, a linear regression function is used to train the system output. Schrauwen et al [3] show that the transformation nonlinear dynamics offered by the reservoir is sufficient for the output layer, called readout, be able to extract the output signals using a simple linear mapping. Promising results in using Reservoir Computing is shown in [3] and [4].

Several parameters influence the efficiency of the RC, for example, the number of nodes used and the type of activation function. Setting these parameters without any indication of problem behavior is difficult, and the use of an optimization algorithm to accomplish this task can be of great importance. Ferreira e Ludermir [5] show

interesting results regarding the use of Genetic Algorithms for optimization of certain parameters of Reservoir Computing.

All computations with Reservoir Computing in this paper used a Matlab toolbox available in [6].

2 Reservoir Computing

Reservoir Computing is a recent paradigm of artificial neural networks developed independently as Liquid State Machine [1] and Echo State Network [2]. In common, all the Reservoir Computing approaches have the feature of using the computational power of recurrent neural networks without the need of train this architecture. The weights are set randomly in the reservoir at the beginning of the process and unmodified. In general, a readout function is used to train the network output. The readout is a function that does linear regression of the input signals using, for example, pseudo-inverse. Fig. 1 shows a diagram of a simple Echo State Network.

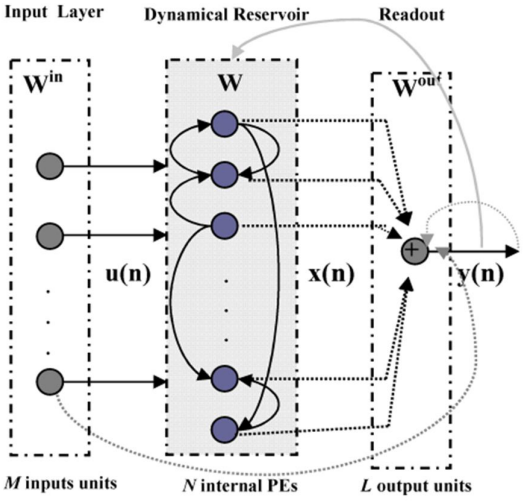

Fig. 1. Basic ESN architecture used is this work.

In the diagram, the ESN has M input units, N internal PEs and L output units. The value of the input unit at time n is u(n), of internal units are x(n), and output units are y(n).

As explained in the figure, the layer named as reservoir receives signal values coming from the input layer. The reservoir, with a certain number of processing units, is designed with recurrent connections. The weights of these connections are random and do not change. The readout layer then does a simple linear mapping of the reservoir output. The dotted lines represent connections that can be trained and shaded lines indicate optional connections.

3 PSO – Particle Swarm Optimization

PSO is a global optimization technique based on a population of solutions. In general, the algorithm is based on the social behavior of flocks of birds, where an individual mimics the actions of the group's best (or most suitable). The process starts with defining the population of solutions. Each individual, named particle, is a possible solution. Each particle has a position and speed, and the update process is based on its best experience and the best experience of the group. PSO was created by Kennedy and Eberhart in 1995 [7].

Let s be the size of the swarm, n the dimension of the problem and t the present time. Each particle $1 \leq i \leq s$ has a position $x_i(t) \in \mathbb{R}^n$ in the solution space and a speed $v_i(t) \in \mathbb{R}^n$, that controls the magnitude and direction of movement. Each particle keeps the best individual position $y_i(t) \in \mathbb{R}^n$ visited up to time t. On the other hand, the whole swarm keeps in memory the best position $\hat{y}(t) \in \mathbb{R}^n$ visited so far by each particle.

Throughout the algorithm, the speed of each particle is guided by two variables, or search points: the best individual position visited so far (the optimization cognitive term, $y_i(t)$) and the global best position visited so far (the optimization social term, $\hat{y}(t)$). Mathematically, the new velocity of each particle is given by equation 1, while equation 2 determines its new position.

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 \left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_2 \left(\hat{y}_j(t) - x_{ij}(t)\right) \quad 1 \leq i \leq s, 1 \leq j \leq n \tag{1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad 1 \leq i \leq s, 1 \leq j \leq n \tag{2}$$

The momentum term (or inertia), w, causes a more exploratory search in the first iterations and higher level of exploitation in recent iterations. This variable is a scalar that usually decreases linearly from 0.9 to 0.4. The variables r_1 and r_2 are uniform random variable ranging from 0 to 1 and are related to two terms in equation (cognitive and social). The values c_1 and c_2 are the coefficients of individual and local acceleration, respectively.

Although the standard PSO and several variations of the original algorithm have been developed and applied in a wide range of optimization problems, solving problems of high complexity with efficient search and fast convergence has been one of the challenges in this research area. It is based on this background that the EPUS-PSO (Efficient Population Utilization Strategy for PSO) [12] was introduced, seeking to improve these aspects in the original algorithm. Thus, the EPUS-PSO is based on three concepts: Population Management (the population size is variable), Solution Sharing (the particles can acquire knowledge of other individuals) and Search Range Sharing (disturbs the solutions and opens the range of possibilities in the search).

The APSO [13] was also developed with the aim of increasing the search efficiency and speed of convergence of the original PSO algorithm. Basically, the APSO consists of two main steps. First, the algorithm identifies the course of its

execution, by evaluating the distribution of population and fitness function of each particle, in which four states of the evolutionary generation of solutions is: exploration, exploitation, convergence or jump out. Through this knowledge, there is automatic control of some variables of the algorithm, as the term of inertia and acceleration coefficients. The second step concerns an elitist learning strategy (ELS) that is activated when the state is classified as evolutionary convergence.

To classify the evolutionary state of generation, the APSO uses the approach known as ESE (Evolutionary State Estimation), based on search behavior and population distribution of solutions of the PSO.

4 PSO for Reservoir Computing Optimization

Reservoir Computing, as well as other architectures of neural networks, suffer from certain disadvantages that come with the ability of connectionist systems to learn from examples and generalize the information absorbed. Among these, it is the heuristic way how techniques such as intelligent computing tend to be seen. In general, there are no rules for defining the network configuration which best applies to a particular problem in order to achieve better results. Included in this configuration, also related to the volume of input data for training, is the definition of the initial parameters of the network.

Some studies have built Hybrid Intelligent Systems that combine neural networks with optimization algorithms. Yamazaki used Tabu Search, Simulated Anealing and Backpropagation to optimize the weights of a neural network Multi-layer Perceptron [15]. More recently, Zanchetin proposed a global and local optimization method combining Tabu Search, Simulated Anealing, Backpropagation and Genetic Algorithms [16]. Other algorithms have been used: Carvalho did an analysis of neural network MLP optimization by particles swarms [17].

Regarding the use of Intelligent Hybrid Systems with Reservoir Computing as one of the techniques involved, some recent work can be cited. In [18] and [19], Ferreira and Ludermir used an evolutionary approach to optimize RC. Using Genetic Algorithms to find the best configuration of the initial parameters of the network, the results with the genetic search proved to be 20% faster than exhaustive search.

Given this research interest in the field of Intelligent Computing and Intelligent Hybrid Systems, this work investigates the impact of using the particle swarm optimization algorithm to find the best configuration of Reservoir Computing initial parameters. The use of PSO in order to optimize the RC results is unprecedented.

5 Results

Based on [5], the following parameters were used in the experiments: number of processing units in the reservoir; activation function in the reservoir; spectral radius of the reservoir weight matrix; interconnection between the input and output layers and feedback connection in the output layer.

To investigate the most convenient way to find the best set of global parameters of the RC, the experiments are also composed of an exhaustive search phase of these variables best settings. Thus, the work will seek to compare the results obtained through these two ways to search for the best global parameters: exhaustive search algorithm and use of particle swarm optimization.

In PSO algorithm, there are the concepts of particle (a single solution) and a population of particles (a set of solutions). By using this optimization algorithm, it is necessary to define the representation of the solution. In the simulations, also based on [5], a particle p is encoded by a vector composed by: N – number of node in the reservoir (100 to 280); nf – node activation function in the reservoir (Logistic or Hyperbolic Tangent); sr – spectral radius (0.75 to 0.95); io – interconnection among the input and output layers (yes or no) and oo – feedback connection in the output layer (yes or no).

A particle is then defined by p = (N, nf, sr, io, oo). The fitness function for each particle is the test error of cross validation using Reservoir Computing.

To compare the efficiency of the methods, we used the MSE (Mean Squared Error) of the prediction of time series in a cross-validation with 10 partitions. Using the method of cross-validation, the number of partitions is an important parameter to be defined. Cross-validation with 10 partitions has been shown to be an appropriate value for most problems [21].

In the exhaustive search for optimal parameters of Reservoir Computing for particular databases, the MSE is calculated on each iteration. Thus, the number of training cycles will be given by the number of parameter settings desired for the purposes of search. The use of PSO and its extensions in order to optimize the search aims to reduce the number of training cycles. With less training cycles, the search is performed more quickly. Thus, the number of training cycles of each method to achieve the optimum initial parameters of Reservoir Computing is also an important approach for comparison.

In exhaustive search, a combination of these parameters was used so that the number of training cycles was set at 400. The number of cycles of training in the use of PSO and its extensions is variable according to the algorithm is running.

For the experiments, we used as databases the following time series: memory test series (MEMTEST), Narma order 10 series (NARMA-10), Narma order 30 (NARMA-30), Mackey-Glass mean chaos (MG-15) and Mackey-Glass moderate chaos (MG-30). Table 1 shows the results of each database in the approaches previously presented.

Table 2 shows the comparison between models according to the Student's t test at 5% significance (95% confidence). In this table, the "=" sign indicates that the null hypothesis was not rejected (the difference between the mean errors is not statistically significant) and the models have the same performance. The sign "<" indicates that the null hypothesis was rejected and that the models for comparison has underperformed selection and, finally, the ">" indicates that the null hypothesis was rejected and that the models can outperform selection.

The PSO approach was the only one that did not get any results statistically inferior to another method in any of the databases. This approach achieved better results than the other in at least one of the databases: better performance than the EPUS-PSO in

NARMA-10 and NARMA-30 and better performance than the APSO and exhaustive search in the NARMA-30 database.

Table 1. MSE and training cycles number for databases.

Approach	MSE	Training cycles number
MEMTEST		
Exhaustive Search	1.0864e-004 (7.6551e-05)	400
PSO	1.0551e-004 (7.3068e-005)	111
EPUS-PSO	1.3474e-004 (1.0202e-004)	41
APSO	1.4486e-004 (1.4082e-004)	120
NARMA-10		
Exhaustive Search	2.3951e-004 (7.9916e-005)	400
PSO	1.6205e-004 (7.0374e-005)	111
EPUS-PSO	3.8978e-004 (3.5512e-005)	43
APSO	2.3504e-004 (3.7616e-005)	120
NARMA-30		
Exhaustive Search	0.0011 (8.5697e-005)	400
PSO	1.3143e-004 (8.2053e-005)	111
EPUS-PSO	0.0011 (7.9712e-005)	41
APSO	0.0010 (6.4345e-005)	120
MGS-15		
Exhaustive Search	3.9498e-005 (4.6705e-005)	400
PSO	3.9438e-005 (4.6859e-005)	111
EPUS-PSO	3.9035e-005 (4.6296e-005)	41
APSO	3.8633e-005 (4.4524e-005)	120
MGS-30		
Exhaustive Search	2.5548e-005 (4.4150e-005)	400
PSO	3.9491e-005 (4.4763e-005)	111
EPUS-PSO	5.8780e-005 (4.2976e-005)	43
APSO	2.6114e-005 (4.4859e-005)	120

One explanation for the similarity of the results over the test set error is the search space of the variables that sought to optimize. This search space was previously defined. Also, the optimization algorithms had routines to prevent it from exceeding its range initially set. Additionally, the PSO obtain better results than its extensions (which were originally developed to improve the original algorithm) can be explained by the nature of the problem in question. The experiments sought to optimize the initial parameters of Reservoir Computing, which depend on each problem and the experience of the examiner. The EPUS-PSO and APSO also have this characteristic: there is a need to define more than the initial parameters of the original PSO algorithm and their values are also defined in a heuristic way. In the experiments presented in this section, the initial parameters of the algorithm EPUS-PSO and APSO were those suggested by the work that gave rise to such extensions of PSO [12] [13].

PSO, APSO and PSO-EPUS obtained the best results according to the criterion of training cycles required to achieve the optimum values. The number of training cycles is closely linked with the runtime of the algorithms of the proposed methods. Thus,

we can conclude that the use of particle swarm optimization to obtain the optimum initial parameters of a Reservoir Computing has come to require only 10.25% of the time required by exhaustive search of these parameters.

Table 2. Comparison between models according to the Student's T test at 5% significance.

Approach/Database	MT	N-10	N-30	MGS-15	MGS-30
PSO					
EPUS-PSO	=	>	>	=	=
APSO	=	=	>	=	=
ES	=	=	>	=	=
EPUS-PSO					
APSO	=	<	=	=	=
ES	=	<	=	=	=
APSO					
ES	=	=	=	=	=

7 Conclusions and Future Works

Considering the two ways to compare methods (test set error and number of training cycles needed), the method performed better in the experiments was the PSO. This approach took 27.75% of the time required for exhaustive search and got no error test statistically worse than any other method tested for all databases.

In the other hand, when in a certain application the execution time is more critical than the performance of the algorithm itself, the best result was achieved by using the EPUS-PSO. This method took up 10.25% of the time required for exhaustive search and obtained statistically similar test error in three of the five databases used.

As future works, we propose: to use a greater number of databases and compare the results with similar approaches for Reservoir Computing optimization; to expand the search space and obtain results statistically more reliable and to study the possibility of using other parameters for optimization.

References

1. Maass, W., Natschlager, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation, v. 14, n. 11, p. 2531-2560 (2002)
2. Jaeger, H.: The echo state approach to analyzing and training recurrent neural networks. Tech. Rep. GMD 148 - German National Resource Center for Information Technology (2001)
3. Schrauwen, B., Defour, J., Verstraeten, D., Van Campenhout, J.: The introduction of time-scales in reservoir computing, applied to isolated digits recognition. LNCS, v. 4668, Part I, p. 471-479 (2007)
4. Antonelo, E. A., Schrauwen, B., Dutoit, X., Stroobandt, D., Nuttin, M.: Event detection and location in mobile robot navigation using reservoir computing. Proc. International Conference on Artificial Neural Networks, v. 4668, Part II, p. 660–669 (2007)

5. Ferreira, A. A., Ludermir T. B.: Genetic algorithm for reservoir computing optimization. International Joint Conference on Neural Networks, p. 811-815 (2009)
6. Schrauwen, B., D'Llaene, M.: Reservoir Computing Toolbox Manual ". Available: http://reslab.elis.ugent.be/
7. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kauf-mann Publishers, Inc, San Francisco, CA (2001)
8. Van Den Bergh, F.: An Analysis of Particle Swarm Optimizers. PhD thesis, University of Pretoria, Faculty of Natural and Agricultural Science (2001)
9. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation v. 6, n. 1, p. 58–73 (2002)
10. Trelea, I. C.: The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection. Information Processing Letters v. 85, p. 317–325 (2003)
11. Pedersen, M. E. H., Chipperfield, A. J.: Simplifying particle swarm optimization. Applied Soft Computing, v. 10, p. 618–628 (2010)
12. Hsieh, S., Sun, T., Liu, C., Tsai, S. J.: Efficient Population Utilization Strategy for Particle Swarm Optimizer. IEEE Transactions, Man and Cybernetics, v. 30, p. 444-456 (2009)
13. Zhan, Z-H., Zhang, J., Li, Y., Chung, H. S-H.: Adaptive Particle Swarm Optimization. IEEE Transactions, Man and Cybernetics, v. 39, p. 1362-1381 (2009)
14. Haykin, S.: Neural networks: a comprehensive foundation. Prentice Hall, Second Edition (1999)
15. Ludermir, T. B., Yamazaki, A., Zanchetinn, C.: An Optimization Methodology for Neural Network Weights and Architectures. IEEE Transactions on Neural Networks, v. 17, n. 5, p. 1452-1459 (2006)
16. Zanchetinn, C., Ludermir, T. B., Almeida, L. M.: Hybrid Training Method for MLP: Optimization of Architecture and Training. IEEE Transactions on Systems, Man and Cybernetics. Part B. Cybernetics, v. 41, p. 1097-1109 (2011)
17. Carvalho, M., Ludermir, T. B.: Particle Swarm Optimization of Neural Network Architectures and Weights. In: International Conference on Hybrid Intelligent Systems, 2007, Kaiserslautern. Proceddings International Conference on Hybrid Intelligent Systems. Los Alamitos : IEEE Computer Society, p. 336-339 (2007)
18. Ferreira, A. A., Ludermir, T. B.:. Evolutionary strategy for simultaneous optimization of parameters, topology and reservoir weights in Echo State Networks. In: IEEE International Joint Conference on Neural Networks, 2010, Barcelona. Proceedings of International Joint Conference on Neural Networks. Los Alamitos : IEEE, p. 1870-1877 (2010)
19. Ferreira, A. A., Ludermir, T. B.:. Using reservoir computing for forecasting time series: Brazilian case study. International Conference on Hybrid Intelligent Systems, p.602–607 (2008)
20. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the Echo State Network Approach. Tech. Rep. No. 159, Bremen: German National Research Center for Information Technology (2002)
21. Witten, I. H., Frank, E.: Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers (2000)