Particle Swarm Optimization of Neural Network Architectures and Weights

Marcio Carvalho, Teresa B. Ludermir
Center of Informatics
Federal University of Pernambuco, P.O. Box 7851
Cidade Universitária, Recife - PE, Brazil, 50732-970
{mrc2,tbl}@cin.ufpe.br

Abstract

The optimization of architecture and weights of feed forward neural networks is a complex task of great importance in problems of supervised learning. In this work we analyze the use of the Particle Swarm Optimization algorithm for the optimization of neural network architectures and weights aiming better generalization performances through the creation of a compromise between low architectural complexity and low training errors. For evaluating these algorithms we apply them to benchmark classification problems of the medical field. The results showed that a PSO-PSO based approach represents a valid alternative to optimize weights and architectures of MLP neural networks.

1. Introduction

Artificial Neural Networks (ANNs) exhibit remarkable properties, such as: adaptability, capability of learning by examples, and ability to generalize. The training process of ANNs for pattern classification problems consists of two tasks, the first one is the selection of an appropriate architecture for the problem, and the second is the adjustment of the connection weights of the network.

The definition of the network architecture is made by the choice of how many hidden layers the network will have (at least one for MLPs) and how many hidden units each hidden layer will have. The architecture of the network determines its complexity which is represented by the number of synaptic weights, i.e. the number of degrees of freedom of the model.

An appropriate architecture for neural networks must neither be too simple to avoid the underfitting (high bias) of the training data, nor too complex to avoid the overfitting (high variance) of the same data. This is a famous machine learning problem called bias and variance dilemma [11]. The main goal is to obtain a model with at the same time low bias and low variance.

The choice of the appropriate number of weights for an MLP neural network is a complex task. This process is generally taken by hand, in a trial and error fashion. An alternative approach is the use of some global optimization technique, such as genetic algorithms (GA) [5], to take a wider search through the space of feasible architectures, considering an acceptable training error and a relatively low complexity.

For the second task, the adjustment of the connection weights, the Backpropagation algorithm (generalized delta rule) [3] is more frequently used. It is a gradient descent method which originally showed good performance in some non-linearly separable problems, but has a very slow convergence and can get stuck in local minima, such as other gradient-based local methods. Another option for the task of training neural networks is also concerned with the use of global search techniques in the attempt to avoid the local minima problem related to gradient descent methods.

In this work we apply the Particle Swarm Optimization algorithm to the global optimization of MLP architectures and connection weights. In this approach, the two processes of architecture optimization and connection weights optimization are interleaved for a number of iterations, with the results of one process being used in the other.

This approach was originally introduced by Yao in [13], in which, architecture, weights and other MLP parameters are optimized by genetic algorithms. In [4], other techniques based on GA used to optimize MLPs are compared for pattern classification problems. And in [2], Zhang and Shao introduce a methodology entirely based in the standard PSO for architecture and weight optimization with few details about implementation and results obtained. Our work is inspired by Zhang and Shao's methodology but introduces the weight decay heuristic [9] in the weight adjustment process in an attempt to obtain more generalization control. For evaluating the proposed algorithm we used benchmark classification problems of the medical field (`cancer`, `diabetes` and `heart`) obtained from the repository Proben1 [8].

The remainder of the article is organized as follows: Section 2 presents the standard PSO algorithm and the proposed methodology is described in Section 3. The Section 4 describes the experimental setup of this work, and the Section 5 presents and analyzes the results obtained from the experiments. Finally, in Section 6 we summarize our conclusions and future works.

2. Particle Swarm Optimization

The PSO optimization technique is an stochastic search through an n-dimensional problem space aiming the minimization (or maximization) of the objective function of the problem. The PSO was created by Kennedy and Eberhart [7] through the attempt to graphically simulate the social behavior of a flock of birds flying to resources.

In PSO, a swarm of particles is kept, where each one of them represents a point in the solution space. The particles of the swarm move through that space looking for better solutions in a way similar to the search for food and other resources made by a flock of birds. This movement is solely based on the best position visited by the individual and the best position visited by the entire population. Next, we give a better definition of how this behavior was originally modeled by the PSO technique.

Let s be the swarm size, n be the dimension of the optimization problem (which in this work is the number of weights of the MLP neural network or the number of hidden layers of the MLP architecture) and t the current instant, each particle $1 \leq i \leq s$ has a position $x_i(t) \in \mathbb{R}^n$ in the solution space and a velocity $v_i(t) \in \mathbb{R}^n$ which controls the direction and magnitude of its movement. Also, each particle keeps in memory the best individual position $y_i(t) \in \mathbb{R}^n$ visited until the instant t, and the whole swarm keeps in memory the best position $\hat{y}(t) \in \mathbb{R}^n$ visited by all the particles. The equations eq. (1) and eq. (2) describe, respectively, how the new velocity and the new position of a particle are determined. In the equations below the term j represents the index of the component of the vectors (dimension n) presented above.

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 (y_{ij}(t) - x_{ij}(t)) + \\ + c_2 r_2 (\hat{y}_j(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (2)$$

The scalar w is the inertia weight (momentum term) which multiplies the prior velocity of the particle (instant t) and controls the degree of exploration of the search. The values r_1 and r_2 are uniform random variables taken from $U_{ij1}(0,1)$ and $U_{ij2}(0,1)$, respectively. The individual and global acceleration coefficients, $0 < c_1, c_2 \leq 2$, respectively, have fixed and equal values, and are responsible for taking control of how far a particle can move in a single iteration.

The standard PSO algorithm is presented in Alg. 1. Rapid convergence in unimodal functions, with good success rate, and premature convergence in multimodal functions are properties frequently attributed to the standard PSO algorithm [6].

Algorithm 1 Standard PSO algorithm

```
1: randomly initialize population of particles
2: repeat
3:   for each particle i of the population do
4:     if $f(x_i(t)) < f(y_i(t))$ then
5:       $y_i(t) = x_i(t)$
6:     end if
7:     if $f(y_i(t)) < f(\hat{y}(t))$ then
8:       $\hat{y}(t) = y_i(t)$
9:     end if
10:  end for
11:  update velocity and position of each particle according to eqs. (1) and (2)
12: until stop criteria being satisfied
```

3. The PSO-PSO Methodology

The methodology used to optimize weights and architectures of MLP neural networks is based on the interleaved execution of two PSO algorithms, one for weight optimization (inner PSO) and the other for architecture optimization (outer PSO). This approach was presented by Zhang and Shao in [2], in which few details were given on the performance of the optimized neural networks.

In this methodology, the outer PSO simply searches for the number of hidden units for each of the considered hidden layers in the MLP network. In this work, we considered only network architectures of one hidden layer for all the classification problems, but the extension for a more general case is straightforward. The inner PSO is responsible for the optimization of weights for each of the architectures (particles) present in the outer PSO. At the end of the inner PSO execution for an architecture of the outer PSO, the best set of weights found is recorded in the particle representing that architecture. The two processes are interleaved for a specific number of times.

The PSO-PSO methodology developed in this work used a PSO algorithm to search for architectures and a PSO with weight decay (PSO:WD) to search for weights. The PSO:WD algorithm was created in a previous work [9] and has more generalization control than the standard PSO. For the evaluation of performance of the particles of the two PSOs, we used different partitions of the example patterns set. The training set partition (50%) was used within the inner PSO to optimize weights while the validation set partition (25%) was used within the outer PSO to search for architectures. The remained data (25%) was used to test the final MLP network found by the process.

It should be noted that all the three partitions used in the methodology are disjoint. That restriction is related to the aim of improving the generalization control of the optimized networks. This can be done through the adjustment of the complexity (number of hidden units) of the networks guided by data examples different from the ones used to guide the search for synaptic weights.

Algorithm 2 PSO-PSO:WD algorithm for simultaneous optimization of weights and architectures

```
1:  randomly initialize population of architectures $A$
2:  repeat
3:      for each particle $A_i$ of the population $A$ do
4:          initiate PSO:WD $P_i$
5:          insert $A_i.net$ into $P_i$
6:          execute $P_i$ by $t$ iterations through training set
7:          $A_i.net = P_i.\hat{y}$
8:          evaluate $f(A_i.net)$ through validation set
9:      end for
10:     for each particle $A_i$ of population $A$ do
11:         update velocity and position of $A_i$ according to eqs. (1) and (2)
12:         update $A_i.net$ to the new architecture represented by $A_i$
13:     end for
14: until stop criteria being satisfied
```

The complete algorithm for the methodology created in this work is presented in Alg. 2, in which the term $A_i.net$ represents the vector used to record the best network found so far for the architecture A_i. Note that this term is updated with the best particle at the end of an execution of the inner PSO (line 7), and is used to assist a new execution of the inner PSO with previous good results (line 5).

4. Experimental Setup

The experiments of this work included the PSO-PSO algorithm based on Zhang and Shao's work [2] and the PSO-PSO:WD algorithm created in this work. The PSO:WD algorithm is better described in the previous work [9] in which we combined the standard PSO algorithm with the weight decay heuristic as an attempt to improve the generalization performance of the trained MLP networks. The same parameters used in [9] for the PSO:WD algorithm was applied in this work for the subprocess of weight optimization.

For evaluating all of these algorithms we used three benchmark classification problems of the medical field obtained from the Proben1 repository [8] (`cancer`, `diabetes` and `heart`).

The representation of MLP networks adopted for the inner PSO (optimization of weights) is based on vectors of real values corresponding to each of the network weights. For the outer PSO algorithm (optimization of architectures) we used integer scalar variables to represent the number of hidden units of the considered hidden layer and a vector of real values (net) to record the best network found so far for the underlying architecture.

$$NMSE = \frac{100}{N \times C} \sum_{n=1}^{N} \sum_{k=1}^{C} (t_k^n - o_k^n)^2 \qquad (3)$$

The quality measure for the particles during the optimization process was the Normalized Mean Squared Error (NMSE), described in eq. (3), in which C is the number of classes of the problem and N is the number of training patterns. The parameters of the algorithms are described below.

- PSO for architectures optimization
  - swarm size ($s = 20$)
  - stop criteria (15 iterations)
  - NMSE of validation data for quality measure of the architectures
  - search space limit $[1, 12]$
  - acceleration factors ($c_1 = c_2 = 1.4960$)
  - fixed inertia factor ($w = 0.7298$)
- PSO for weights optimization
  - swarm size ($s = 30$)
  - stop criteria (100 iterations)
  - NMSE of training data for quality measure of the network weights
  - search space limit $[-2.0, 2.0]$
  - acceleration factors ($c_1 = c_2 = 1.4960$)
  - fixed inertia factor ($w = 0.7298$)

5. Results

In order to compare the Classification Error Percentage (CEP) performance of all the algorithms, each one was executed 50 independent times for each benchmark data set. In every execution, the corresponding data set was randomly partitioned into three stratified subsets: training data set (50%), validation data set (25%) and test data set (25%). The results of the experiments are shown in Table 1 in which we report the mean CEP, the standard deviation and mean time in seconds for the two tested algorithms with the 3 data sets of the medical field.

Table 1. Mean CEP($\bar{x}$), standard deviation (σ) and mean time ($\bar{t}$) in seconds for each algorithm and each of the 3 data sets. The best mean CEP for each data set is indicated in bold.

Algorithm	Data Set			
		Cancer	Diabetes	Heart
PSO-PSO	$\bar{x}$	4.8342	24.3646	19.8870
	σ	3.2510	3.5710	2.1473
	$\bar{t}$	606.9000	900.8400	1238.0000
PSO-PSO:WD	$\bar{x}$	**4.1371**	**23.5417**	**18.0957**
	σ	1.5056	3.1595	3.0570
	$\bar{t}$	807.1400	916.5800	1332.9000

From the results showed in the Table 1 we note that for all the tested data sets, the PSO-PSO algorithm with the weight decay technique applied to the weights adjustment phase obtained better mean generalization performance than the algorithm based on Zhang and Shao's work [2]. We note also from Table 1 the large execution times presented by the two algorithms. It confirms the need to carefully choose the number of particles for the swarm of architectures, the number of iterations for the swarm of architectures and the number of particles for the swarm used in weights adjustment. These variables can be found by trial and error for a given data set, taking into consideration the overall execution time of the whole algorithm and avoiding the possibility of overtraining in the weights adjustment phase.

At the Figures 1, 2 and 3, we present boxplots of the resulting CEPs for the two algorithms applied to the `Cancer`, `Diabetes` and `Heart` data sets, respectively. In Figure 1, we see a little difference in the median of the performance obtained for the `Cancer` data set with PSO-PSO:WD comparing to the PSO-PSO algorithm. We note also small variances in both algorithms, and a small trend for better results presented by the PSO-PSO:WD algorithm.

In Figure 2, we also note similar performances, but with

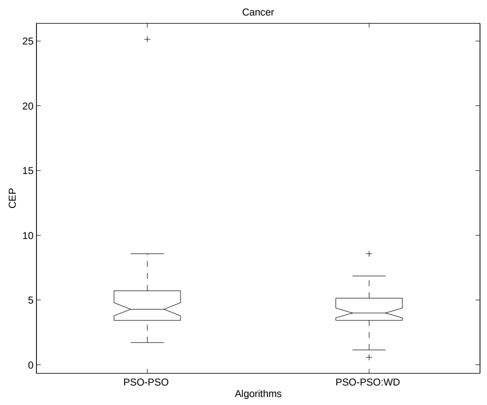

Figure 1. Boxplot of the CEP for the `Cancer` data set

the PSO-PSO:WD algorithm presenting a better median and the best CEP result from the the two algorithms compared. In Figure 3, we clearly note a better performance presented by the PSO-PSO:WD algorithm compared to the PSO-PSO one, related to the median of the performance, the best CEP result, and a trend to present better results showed by the lower halves of the boxes.

Considering the t-tests made we can statistically state that only for `Heart` data set the PSO-PSO:WD algorithm performed better than the PSO-PSO algorithm with a t value of -3.3565.

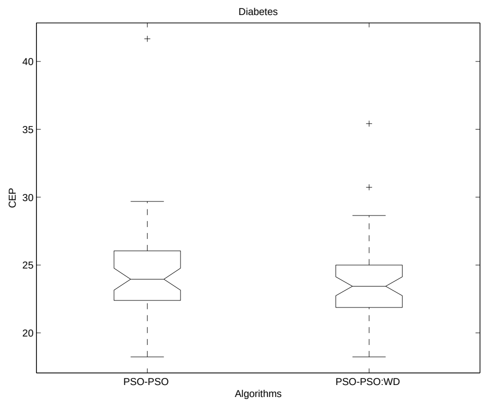

Figure 2. Boxplot of the CEP for the `Diabetes` data set

This result from the t-student statistical test does not invalidate the other results presented in Table 1 and in Figures 1, 2 and 3. It states that the PSO-PSO with weight decay technique performed similar to the PSO-PSO algorithm in two of the three problems tested in this work, but better in `Heart` data set, one of the two hardest data sets

Table 2. Mean CEP ($\bar{x}$) and standard deviation (σ) for each algorithm developed in this work compared to results obtained from previous works with the same objectives and applied to the same data sets. The best mean CEP for each data set is indicated in bold.

Algorithm	Data Set			
		Cancer	Diabetes	Heart
PSO-PSO	$\bar{x}$	4.8342	24.3646	19.887
	σ	3.2510	3.5710	2.1473
PSO-PSO:WD	$\bar{x}$	4.1371	23.5417	18.0957
	σ	1.5056	3.1595	3.0570
Evolutionary Programming [14]	$\bar{x}$	**1.3760**	22.3790	16.7650
	σ	0.9380	0.0140	2.0290
Genetic Algorithm (Conn. Matrix)[4]	$\bar{x}$	3.2300	24.5600	23.2200
	σ	1.1000	1.6500	7.8700
Genetic Algorithm (Neural Cross.)[10]	$\bar{x}$	9.3100	**21.4200**	**14.9000**
	σ	4.4000	2.1900	2.7800
Tabu Search[12]	$\bar{x}$	(-)	27.4045	(-)
	σ	(-)	(-)	(-)
Simulated Annealing[12]	$\bar{x}$	(-)	27.1562	(-)
	σ	(-)	(-)	(-)
Yamazaki (TS+SA)[12]	$\bar{x}$	(-)	25.8767	(-)
	σ	(-)	(-)	(-)

from the chosen ones (the hardest problem of this work was the `Diabetes`).

In Table 2, we present the mean CEP results obtained by other works involving optimization of connection weights and architectures, or just the optimization of connection weights and connectivity pattern.

In [14] the evolutionary programming technique is used to optimize the initial values of connection weights, architectures, choice of activation functions, as well as other important parameters of MLP networks. In [4] and [12], Genetic Algorithm, Tabu Search, Simulated Annealing and Yamazaki Methodology are applied and compared in the optimization of connection weights and connection configuration (connectivity pattern). The Yamazaki Methodology combines the advantages of SA and TS. In [10], a modified version of GA is used to optimize weights and architectures of MLP neural networks. In this version, the crossover genetic operation was made by the exchange of neurons with their underling connection weights.

Through the results exposed, we can state that the presented methodology along with the Zhang and Shao's work are effective alternatives to the MLP weights and architectures optimization problem. The obtained results are not the best ones known for the chosen benchmark data sets, but lie between the generalization performance presented by well known optimization techniques in these pattern classification problems.

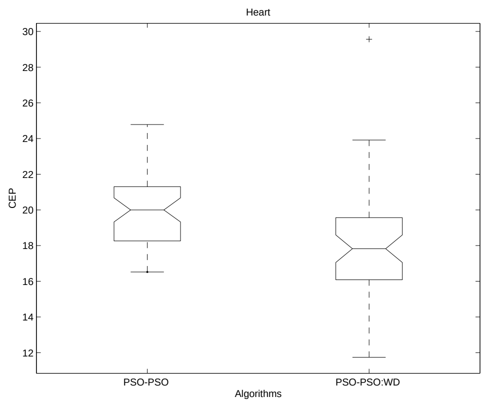

Figure 3. Boxplot of the CEP for the `Heart` data set

6. Conclusions

In this work, we have analyzed the feed-forward neural networks weights and architecture optimization problem with the use of a methodology entirely based on the Particle Swarm Optimization algorithm. The results obtained by this methodology are situated between the results presented by other well studied techniques such as Genetic Algorithms or Simulated Annealing.

This methodology was inspired by a similar process described in [2], where two PSO algorithms are interleaved in the optimization of architectures (outer PSO) and connection weights (inner PSO) of MLP neural networks. A small modification made in this work concerning generalization control improvements was the use of the weight decay heuristic in the inner PSO, i.e. the process of weights adjustment.

The performance of the tested algorithms was evaluated with well known benchmark classification problems of the medical field (`Cancer`, `Diabetes` and `Heart`) obtained from the Proben1 [8] repository of machine learning problems. The results from the t-student tests showed that PSO-PSO:WD obtained generalization performance similar to Zhang and Shao's PSO-PSO (the latter was outperformed only with the `Heart` data set). From Table 1 we saw that the PSO-PSO implementation with weight decay heuristic presented better mean generalization performance for all data sets.

As future works, we consider the fusion of the two interleaved tasks (inner PSO and outer PSO) in a single PSO searching for weights and architectures as a truly simultaneous optimization process. This approach must not violate the dimensionality constraint of all the particles having the same number of components, and should minimize the effects of the context switch (use of training data to optimize weights and validation data for architectures). Another interesting possibility is the addition of a connectivity pattern optimization process to the PSO-PSO algorithm, since this approach have presented good generalization performance in other works [4, 12].

Acknowledgments

The authors would like to thank CNPq (Brazilian Research Council) for their financial support.

References

[1] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting", in *Advances in Neural Information Processing* (3), R. Lippmann, J. Moody, and D. Touretzky, Eds., 1991, pp. 875-882.

[2] C. Zhang, H. Shao. "An ANN's Evolved by a New Evolutionary System and Its Application" in *In Proceedings of the 39th IEEE Conference on Decision and Control*, vol. 4, pp. 3562-3563, 2000.

[3] D. Rumelhart, G.E. Hilton and R.J. Williams, "Learning representations of back-propagation errors", *Nature* (London), vol.323, pp. 523-536, 1986.

[4] E. Cantu-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems", *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, vol. 35, pp. 915-927, 2005.

[5] E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing.* Natural Computing Series. MIT Press. Springer. Berlin. (2003).

[6] F. van den Bergh, "An Analysis of Particle Swarm Optimizers", PhD dissertation, Faculty of Natural and Agricultural Sciences, Univ. Pretoria, Pretoria, South Africa, 2002.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization", in: *Proc. IEEE Intl. Conf. on Neural Networks (Perth, Australia)*, IEEE Service Center,Piscataway, NJ, IV:1942-1948, 1995.

[8] L. Prechelt "Proben1 - A set of neural network benchmark problems and benchmark rules", Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Germany, September, 1994.

[9] M. Carvalho, T.B. Ludermir, "Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay," his, p. 5, in *Sixth International Conference on Hybrid Intelligent Systems* (HIS'06), 2006.

[10] N. Garcia-Pedrajas, D. Ortiz-Boyer and C. Hervas-Martinez, "An alternative approach for neural network evolution with a genetic algorithm: crossover by combinatorial optimization", *Neural Networks*, vol. 19, pp. 514-528, 2006.

[11] S. Geman, E. Bienenstock and R. Doursat, "Neural networks and the bias/variance dilemma", *Neural Computation*, vol. 4, pp. 1-58, 1992.

[12] T.B. Ludermir, A. Yamazaki and C. Zanchetin, "An Optimization Methodology for Neural Network Weights and Architectures" *IEEE Transactions on Neural Networks*, vol. 17, pp. 1452-1459, 2006.

[13] X. Yao, "Evolving artificil neural networks", *Proceedings of the IEEE*, vol. 87, pp. 1423-1447, 1999.

[14] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks", *IEEE Transactions on Neural Networks*, vol. 8, pp. 694-713, 1997.