Particles Gradient: A New Approach to Perform MLP Neural Network Training based on Particle Swarm Optimization

César Daltoé Berci and Celso Pascoli Bottura

Faculdade de Engenharia Elétrica e Computação, Universidade Estadual de Campinas
Av. Albert Einstein 400, Campinas, Brazil

Abstract. The use of heuristic algorithms in neural networks training is not a new subject. Several works have already accomplished good results, however not competitive with procedural methods for problems where the gradient of the error is well defined. The present document proposes an alternative for neural networks training using PSO(Particle Swarm Optimization) to evolve the training process itself and not to evolve directly the network parameters as usually. This way we get quite superior results and obtain a method clearly faster than others known methods for training neural networks using heuristic algorithms.

1 Introduction

ANNs (Artificial Neural Networks) is a computational paradigm inspired in the operation of the biological brain, and seeks to explore certain properties present in the human neural processing, that are very attractive from the computational view point[10]. In this paper will be treated a specific type of ANNs, called MLP, a widely applied ANNs model, for which is found a vast literature.

MLP ANNs training process are usually a non linear optimization process, frequently based on the gradient of the ANNs error surface, which is calculated through the *backpropagation* algorithm [18],[19]. Among the more efficient methods can be mentioned, the quasi-Newton method: *BFGS* [1] and the conjugated directions method: *Scaled Conjugate Gradient* [14],[15], all of them considered order 2 optimization methods.

The present document introduces a new approach for MLP ANNs training using PSO [11],[17],[12],[6] (or another heuristic algorithm like genetic algorithms, however, differing from others proposals founded in the pertinent literature [20],[16] [21],[7],[9]). The proposed method: PG uses a sub-optimum gradient, and unit steps are taken in the direction of this gradient. This proposal makes use of the full exploratory capacity of the PSO, united with the efficiency of gradient descent methods.

2 The Particle Gradient Method

There are some works that try to optimize the ANNs vector of parameters using meta-heuristic algorithms, in most cases genetic algorithm are implemented. This process is in general more onerous from the computational cost[1] viewpoint, and shows poor results (with respect to the process convergence rate) when compared to conventional optimization methods, based on gradient descent. This scenario created the main motivation to this work, to create a meta-heuristic training method, that rivals procedural training methods with respect to convergence rate, here understood as learning efficiency, even when implemented in a digital machine.

The proposed method, represents an alternative solution inspired in the work of Chalmers *The Evolution of Learning: An Experiment in Genetic Connectionism*[8], that applied evolutionary processes to evolve the learning process itself, and not its solution.

In the PG: Particles Gradient method, the PSO algorithm is not used to optimize the vector of parameters itself, but to optimize the learning process by finding a sub-optimum gradient vector at each learning iteration (season).

2.1 Codification

The PSO algorithm applied in the proposed PG algorithm, uses a population of n_p particles, with a real codification described as follows:

- Particle Position: Vector containing real values belonging to the space R^N, represent the error gradient for an ANN.
- Fitness: $e(x, \theta - p)$ where p is a particle position, x is the ANN input and θ is its vector of parameters.
- Speed Modulation: A method to control the particles speed to provide a faster convergence rate [3].

2.2 The PG Algorithm

This algorithm is based on the gradient descent algorithm where steps are taken in the direction of a gradient vector, however, here unit steps are taken in the direction of a sub-optimum gradient found by the PSO algorithm.

To each iteration of the main algorithm, an initial gradient vector is calculated using the *Backpropagation* method, or simply by setting it equals to origin, which is clearly a good estimate, and than inserted in the population[2]. Later a vector representing a sub-optimum gradient is evolved by the PSO algorithm by n_i generations, and then an unit step is taken in this new descent direction.

The use of the initial gradient seeks to accelerate the convergence of the PSO algorithm, giving to him a reference point. However, the execution of the method without the use of an initial gradient also obtains good results, even similar to ones found using an initial gradient, as can be seen in [4]. This procedure is useful when it is not possible to accomplish the retro-propagation phase of the ANN, impeding the gradient vector construction, and consequently the application of faster procedural optimization methods.

[1] Here the computational cost of a procedure is understood as the number of sum and multiplication operations necessary to accomplish this

[2] Inserting a simple point in a particle's population consists of creating a new particle in the point's position

The GP algorithm for MLP ANNs training is described in detail as follow.

Algorithm 1: Particles Gradient.

```
Determine: $n_p$, $n_i$, $n_{max}$;
Initialize: $\theta$;
for $i$=1 to $n_{max}$ do
    Calculate $g_0$ by the retro-propagation of the error, or set $g_0 = 0$;
    Find the gradient: $g = PSO(n_p, n_i, g_0)$ ;
    Unit step: $\theta_{i+1} = \theta_i - g$
end
```

where PSO represents a Particle Swarm Optimization algorithm.

The algorithm 1, is the result a exhaustive study of the training process, and the functional analysis of the relations between the quantities of interest, taking into account the dimensionality of the involved spaces and the characteristics a priori known about the problem.

These studies converged to a method where the search blind is applied in a space of same dimension of the vector of parameters. However, this choice brings a great advantage with respect to the cost function to be optimized, or the surface where the particles will be moved. This choice provides a condition very particular to the PSO method, which is widely explored in this work.

In the application of a blind search method, as in case, the efficiency of the optimization process may be considerably increased if the method is initialized near the neighborhood of a local optimum point, which represents a good solution to the problem. However, this points are not known a priori, nor their neighborhoods.

The same happens in the ANNs training, the error surface is not complete known, so nothing can be done to increase the training algorithm efficiency. Moreover, it is known that a sufficient small step in a descent direction is ways a minimizing step. Therefore, we conclude that a R^N vector, representing a descent direction, will be certainly found in a neighborhood of the origin.

This information is the main goal of the proposed method, given to blind search method what it needs, a good initial condition. This approach gives the PSO method a considerable efficiency increase, so the algorithm PG, presents a significantly higher convergence rate when compared with others meta-heuristic methods in the same context.

Another important feature of the PSO method, is the intrinsic parallelism of the algorithm. Its implementation in a sequential machine, as a digital computer, will generate a process computationally very onerous, however, in a completely parallel machine, still hypothetical, is obtained a very faster and efficient process.

3 Examples and Comparisons

With the intention of determining the relative efficiency of the proposed method compared with others founded in the literature, some ANN application examples are used, and the efficiency of the learning process is evaluated when the network is trained by different methods.

In this document will be considered, as comparison bases, two quasi-Newton methods, two conjugated directions methods and one simple gradient descent method, all described as follows:

- GRAD: Optimium Gradient [13]: Gradient descent method with super-linear convergence, the fastest among methods with linear convergence rate.
- DFP: Davidson Fletcher Powell [13]: quasi-Newton Method with quadratic convergence.
- BFGS:Broyden Fletcher Goldfarb Shanno [1]: quasi-Newton Method with quadratic convergence, and with smaller sensibility to the bad numerical conditioning than the DFP method.
- FR: Fletcher Reeves [1]: Conjugated directions method, with N-steps quadratic convergence.
- SCG: Scaled Conjugated Gradient[15]: Conjugated directions method that do not use unidimensional searches. It possesses N-steps quadratic convergence, and it is the fastest among these methods from the computational cost viewpoint.

The process of unidimensional search used in the algorithms GRAD, DFP, BFGS and FR is the golden section method, applied by 30 iterations on the initial interval.

3.1 Motor Currents

In theory, the current of a three-phase induction motor can be easily calculated on the basis of motor voltage and power, as shown in equation (1).

$$I = \frac{P}{\sqrt{3}V\eta} \tag{1}$$

where P and V represent the motor power and tension respectively. The variable η takes into account the power factor and efficiency of the motor, that are based on construction factors, the mechanical load and the rotation of the motor. Thus, it is clear difficult to specify the variable η and so, the motor current.

The problem in question uses a neural estimator for the current calculation, based on motor power, voltage and rotation, through a MLP network containing 3 neurons in its sensorial layer, and with 1 neuron in its output layer.

The set used in the training consist on 300 samples obtained from manufacturers catalogs, including motors that meet the following values ranges:

- Power: 0.1 a 330 KW.
- Rotation: 600,900,1200,1800 e 3600 rpm.
- Tension: 220, 380 e 440 V.
- Current: 0.3 a 580 A.

In a first analysis will be used in comparison to the proposed algorithm, a training algorithm that uses the PSO technique directly applied to minimize the error surface with respect to the vector of parameters.

This method uses exactly the same meta-heuristic of the proposed algorithm, and also the same implementation, differing only in the application approach.

The configuration of both methods are described as follows:

Neural Network:

- Topology: 3 neurons in sensorial layer, 9 neurons in hidden layer and 1 neuron in output layer.
- Linear output.
- Hyperbolic tangent activation functions.

Particle Swarm Optimization: PSO

- 136 particles.
- 400 iterations.
- Speed modulation [3],[5].

Particles Gradient: PG

- 136 particles.
- 5 epocs.
- 80 iteration per epoc.
- Speed modulation.

Due the stochastic characteristics of both methods tested, it is necessary a statistical analysis of the results. Has been chosen in this paper the completion of 20 repetitions of the training process and subsequent analysis of average results, which are summarized in figure 1.

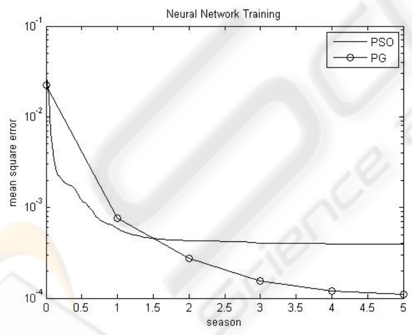

Fig. 1. Average values for mean squared error during the training process.

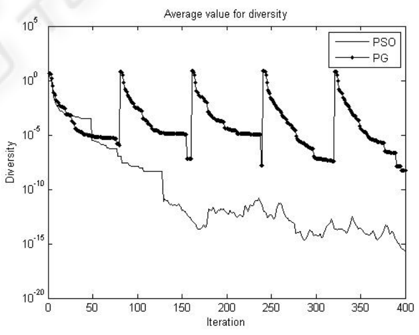

Fig. 2. Average values for diversity during the training process.

The graph shown in Figure 1 represent the average error value obtained by each method in each epoc of training process, whereas for PSO algorithm intermediate values, because it have accomplished 400 iterations, taken to give the same conditions for the two methods.

Is clearly to see that the proposed method is quite superior to the PSO algorithm. Like both use the same heuristic, and more, they have the same implementation, becomes clear the fact that the proposed approach significantly increase the optimization process efficiency, accomplishing the main objective.

An analysis also relevant in this study, is to verify the populations diversity in both methods, what together with the results above, gives an more accurate understanding about the optimization mechanism. The metric here chosen to measure the diversity value, is the variance of the particles fitness. Figure (2) shows the average value of diversity for both methods in each iteration[3] of the training process.

Its is clearly in Figure 2 the great superiority of the proposed method with respect to diversity preservation when compared to the PSO algorithm. This fact is due mainly to restart of the population for each new epoc, which restores its maximum diversity[4].

A joint analysis of both Figures 1 and 2 provides a more complete and mature understanding of the optimization mechanism used by both particles swarms. In the proposed method, the high preservation of diversity, is providing mobility enough to swarm continues finding best solutions, while in the PSO algorithm, the drop in diversity cause a rapid convergence at the beginning of the process, but after some time the population loses its exploratory capacity and becomes incapable of improving its solution.

Thus, is clear that the design here proposed, creates a method with high convergence rate and also high diversity preservation, what are not intrinsic of the PSO algorithm.

Now seeking to compare the efficiency of the PG with the previously mentioned procedural methods, a ANN containing 3 neurons in its hidden layer was used, having the configuration: 3-3-1. The result of the network training can be visualized in the Figure 3 and Figure4.

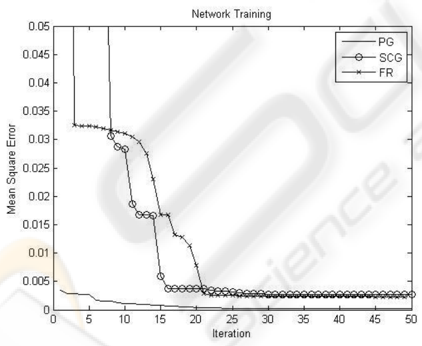

Fig. 3. Average values for mean squared error during the training process.

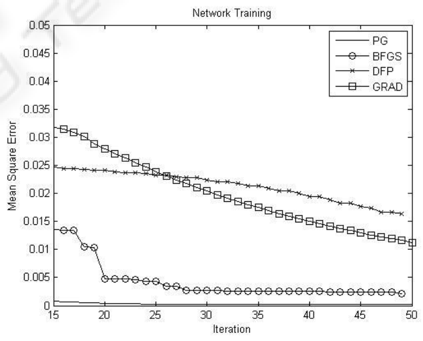

Fig. 4. Average values for diversity during the training process.

[3] Here iteration is used to describe the intermediate steps of training processes, and one iteration of main process, of the proposed method, is referred as epoc

[4] The reader may ask if restarting the population in the method PSO do not lead to the same result, which clearly does not happen for obvious reasons, but these will not be discussed here

Again, due the stochastic characteristic of the PG, these figures show its average behavior for a total of 20 repetitions of the training process. For this example the PG have presented quite superior results when compared with the procedural methods.

In spite of to providing a significant reduction of the network mean square error in each epoc, the proposed algorithm is quite onerous from the computational cost viewpoint, given that a search process should be completed at each iteration. In that way, the execution of the algorithm may become too *slow*, depending of its configuration.

In [15], the author proves the superiority of the method SCG, in respect to computational cost and convergence rate, over the other methods here analyzed. This result is based in the need of unidimensional search required by most methods, that has computational cost $O(N^2)$ per iteration. The SCG method presents a computational cost: $O(2N^2)$ per epoc, that is very inferior to the ones of the methods GRAD, DFP, BFGS and FR that possess computational costs:[5] $O(31N^2)$.

The proposed method presents a total cost $O(n_i n_p N^2)$, that can become quite superior to the ones of the other methods depending on the choice of n_i and n_p. However, the fast convergence of the method compensate, in some cases, this high computational cost.

Now, to build a efficient comparison, let us assume that the function $O(\cdot)$ is linear with respect to the parameters [6] n_p and n_i, we can write:

$$\begin{aligned} C_t^{SCG} &= C_t^{GP} \\ n_e^{SCG} O(N^2) &= n_e^{GP} O(n_i n_p N^2) \\ n_e^{SCG} O(N^2) &= n_e^{GP} n_i n_p O(N^2) \\ n_e^{SCG} &= n_e^{GP} n_i n_p \\ n_e^{GP} &= \frac{n_e^{SCG}}{n_i n_p} \end{aligned} \tag{2}$$

Therefore, if is selected a number of epocs for the algorithm SCG: $n_e^{SCG} n_e =^{GP} n_i n_p$, it is estimated that both algorithms show a very similar computational cost, allowing an analysis of the relative efficiency with respect to convergence rate $\times$ computational cost.

To accomplish this comparison, lets consider the following parameters set:

- Number of epocs: SCG: $n_e^{SCG} = 3200$, GP: $n_e^{GP} = 4$
- Iterations per epoc: $n_i = 60$
- Number of particles: $n_p = 30$
- Number of repetitions: $n_r = 5$

[5] This value is due to the fact that the unidimensional search to makes 30 iterations for each training epoc

[6] This assumption is made to allow a brief analysis of the problem in order to consider the non-linearity of this function to obtain an accurate result, would not add significant information to present analysis, due the fact that this functions are affected by the methods implementation, and not by its definitions

The Figure 5 shows the ANN mean squared error, after been trained by both methods, in each repetition.

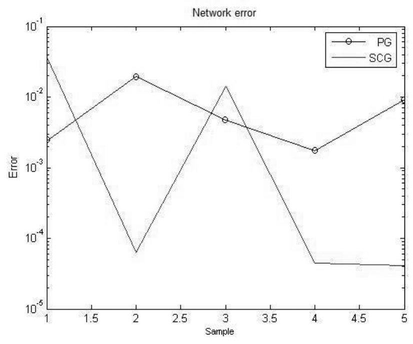

Fig. 5. Comparison between the algorithms: GP and SCG.

In the Figure 5 we can easily see that despite of the proposed method present a higher computational cost, its efficiency rivals the efficiency of SCG method, actually considered one of the most efficient method.

3.2 Curve Fitting

In this example a group of 100 test samples of input-output pairs was used for a quadratic function $y = x^2$, where a white noise of width 10^{-4} was inserted in both signs (input and output). The training was accomplished for various networks configurations, using the proposed algorithm and the algorithms previously mentioned. The results can be observed in Tables 1 and 2.

Table 1. Learning Results: PG SCG FR.

Architecture	PG	SCG	FR
1-3-1	0.00016	0.00066	0.00030
1-6-1	0.00019	0.00069	0.00031
1-9-1	0.00018	0.00078	0.00033
1-18-1	0.00020	0.00082	0.00035
1-6-6-1	0.00024	0.01065	0.01395
1-12-12-1	0.00018	0.01135	0.01735
1-6-12-6-1	0.00017	0.00667	0.05779
1-12-18-12-1	0.00014	0.01764	0.06142

Table 2. Learning Results: BFGS DFP GRAD.

Architecture	BFGS	DPF	GRAD
1-3-1	0.00032	0.00079	0.01012
1-6-1	0.00037	0.00529	0.01197
1-9-1	0.00039	0.00617	0.01207
1-18-1	0.00045	0.00657	0.01287
1-6-6-1	0.01161	0.00959	0.01342
1-12-12-1	0.00943	0.00993	0.01377
1-6-12-6-1	0.00070	0.01060	0.01398
1-12-18-12-1	0.00420	0.01254	0.01485

For this problem it is also possible to notice that the final errors obtained by the proposed method, were also quite inferior to the ones of the others tested algorithms. Another outstanding characteristic observed in the exposed results is the robustness of the PG method with respect to variations in the ANN topology. Due the stochasticity of the learning process, it is possible to infer that the PG method has presented the same final errors results for the several tested configurations.

Several other tests were also accomplished [2], based on different problems with different network topologies, including comparisons with genetic algorithms. In all of this tests was founded similar results to those here presented, confirming all conclusions here taken.

4 Conclusions

The method proposed in this paper represents a new approach for MLP ANNs training using meta-heuristic methods, with different features of others similar methods.

The use of a particle swarm optimization algorithms, as another meta-heuristic methods like genetic algorithms, in ANN training was, until now, not competitive given the inferior performance of these methods when compared to procedural optimization techniques. This new approach, however, is competitive in this scenery, reaching results sometimes comparable with the ones of the *faster* MLP ANN training methods, and still preserving characteristics of the heuristic methods.

One of the main advantages of the PG method, is the possibility to train ANNs with the same efficiency of methods as BFGS and SCG, without knowledge about the error gradient, enlarging its application to several problems, as the one proposed in [4],[2], where the ANN is trained to estimate the state of a dynamic system, and is not possible to calculate the ANN error, and so, the error gradient is unavailable.

Another outstanding characteristic for the proposed method is the preservation of the probability to converge to a global optimum, as in the particle swarm optimization, that is superior from that observed in procedural methods, where it is very probable that they will converge to an local optimum closer to where the method was initiated.

The high computational cost, characteristic of heuristic methods as the genetic algorithms, also is present in the proposed method that is more onerous that the other methods here discussed when implemented in a digital machine. However, it is clear from the shown examples, that this high computational cost is compensated by the accelerated convergence rate. Moreover, the intrinsic parallelism of the proposed method, allows its an implementation in a parallel machine, that can be several orders of magnitude faster than the procedural methods here discussed, inherently sequential.

So we may conclude that the Particles Gradient method here proposed, represents a viable alternative solution for ANNs training, in some situations even being the preferable one, and in a more complex and unusual application, mainly when is difficulty or even impossible the construction of the gradient vector, this method can be one of better choices.

References

1. R. Battiti and F. Masulli. Bfgs optimization for faster and automated supervised learning. *INNC 90 Paris, International Neural Network Conference,*, pages 757–760, 1990.
2. Csar Dalto Berci. *Observadores Inteligentes de Estado: Propostas*. Tese de Mestrado, LCSI/FEEC/UNICAMP, Campinas, Brasil, 2008.
3. Csar Dalto Berci and Celso Pascoli Bottura. Modulao de velocidade em otimizao por enxame de partculas. *Proceedings of 7th Brazilian Conference on Dynamics, Control and Applications. Presidente Prudente, Brasil*, 7:241–248, 2008.

4. Csar Dalto Berci and Celso Pascoli Bottura. Observador inteligente adaptativo neural no baseado em modelo para sistemas no lineares. *Proceedings of 7th Brazilian Conference on Dynamics, Control and Applications. Presidente Prudente, Brasil*, 7:209–215, 2008.
5. Csar Dalto Berci and Celso Pascoli Bottura. Speed modulation: A new approach to improve pso performance. *Accepted Article: IEEE Swarm Intelligence Symposium 2008, Sheraton Westport at St. Louis, Missouri*, 2008.
6. F. den van Bergh. *An Analysis of Particle Swarm Optimization*. PhD thesis, Departament of Computer Science, University of Pretoria, 2002.
7. Jürgen Branke. Evolutionary algorithms for neural network design and training. In *1st Nordic Workshop on Genetic Algorithms and its Applications*, 1995. Vaasa, Finland, January 1995.
8. D.J. Chalmers. The evolution of learning: An experiment in genetic connectionism. *Proceedings of the 1990 Connectionist Summer School*, pages 81–90, 1990.
9. A. Fiszelew, P. Britos, A. Ochoa, H. Merlino, E. Fernndez, and R. Garca-Martnez. Finding optimal neural network architecture using genetic algorithms. *Software & Knowledge Engineering Center. Buenos Aires Institute of Technology. Intelligent Systems Laboratory. School of Engineering. University of Buenos Aires.*, 2004.
10. C. Fyfe. *Artificial Neural Networks*. Department of Computing and Information Systems, The University of Paisley, Edition 1.1, 1996.
11. F. Heppner and U. Grenander. *A Stochastic Nonlinear Model For Coordinate Bird Flocks*. AAAS Publications, Washington, DC, 1990.
12. J. Kennedy and R.C Eberhart. Particle swarm optimization. *Proceedings of IEEE Int. Conf. on Neural Network*, Piscataway, NJ:1942–1948, 1995.
13. D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2nd edition, 1984.
14. M.F. Mller. Learning by conjugate gradients. *The 6th International Meeting of Young Computer Scientists*, 1990.
15. M.F. Mller. A scaled conjugate gradient algorithm for fast supervised learning. *Computer Science Department, University of Aarhus Denmark*, 6:525–533, 1990.
16. D. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 762–767, 1989.
17. C.W. Reynolds. Flocks, herds, and schools: A distributed behavior model. *Computer Graphics*, 21(4):25–34, 1987.
18. D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin. *Backpropagation: The basic theory*. Lawrence Erlbaum Associates, Inc., 1995.
19. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation, in: Parallel distributed processing: Exploration in the microstructure of cognition. *Eds. D.E. Rumelhart, J.L. McClelland, MIT Press, Cambridge, MA.*, pages 318–362, 1986.
20. Udo Seiffert. Multiple layer perceptron training using genetic algorithms. *ESANN'2001 proceedings - European Symposium on Artificial Neural Networks*, pages 159–164, 2001.
21. Zhi-Hua Zhou, Jian-Xin Wu, Yuan Jiang, and Shi-Fu Chen. Genetic algorithm based selective neural network ensemble. *Proceedings of the 17th International Joint Conference on Artificial Intelligence.*, 2:797–802, 2001.