Comparative analysis of Genetic algorithm, Simulated annealing and Cutting Angle method for artificial neural networks

Ranadhir Ghosh, Moumita Ghosh, John Yearwood, Adil Bagirov,

School of InformationTechnology and Mathematical Sciences, University of Ballarat,
PO Box 663, Ballarat – 3353, Australia
{r.ghosh, m.ghosh, j.yearwood, a.bagirov}@Springer.de

Abstract. Neural network learning is the main essence of ANN. There are many problems associated with the multiple local minima in neural networks. Global optimization methods are capable of finding global optimal solution. In this paper we investigate and present a comparative study for the effects of probabilistic and deterministic global search method for artificial neural network using fully connected feed forward multi-layered perceptron architecture. We investigate two probabilistic global search method namely Genetic algorithm and Simulated annealing method and a deterministic cutting angle method to find weights in neural network. Experiments were carried out on UCI benchmark dataset.

1 Introduction

Artificial neural networks (ANN) are the interconnection of basic units called artificial neurons. Those are capable of performing classification, learning and function approximation. Learning is the main essence of ANN. Learning can be considered as a weight-updating rule of the ANN. Most of the neural learning method strictly depends on the architecture of the ANN. The nonlinearity of ANN results in the existence of many sub-optimal solutions. There are many problems associated with the multiple local minima in neural networks [1][2][3]. Some of the aspects with existing learning methods for MLP can be summarized as the convergence tends to be extremely slow, learning constants must be guessed heuristically, convergence to the global minimum is not guaranteed. [4]. The global search method guarantees the global solution.

There exist solutions that include multiple starts from randomly chosen initial points. Those are simulated annealing, random search, and evolutionary computing [5-14]. These methods are probabilistic in nature and they can find the globally optimal solution with a certain probability. Hence the solution partly depends on the number of iterations of the algorithm. In contrast, there exist deterministic techniques which are capable of finding global optimal solution. Deterministic methods include tabu search, branch-and-bound, generalized cutting plane and systematic search [11,12]. But the computational costs of these methods are extremely high.

In this paper we investigate three different global optimization methods to find the weights of ANN. Two of them are probabilistic global search method namely genetic algorithm, and simulated annealing method respectively. The third one is a recently developed cutting angle method of deterministic global optimization [15-17].

2 Research Methodology

In this section we describe Genetic algorithm, Simulated annealing and Cutting angle method.

2.1 Genetic Algorithm

Genetic algorithm (GA) learning provides an alternative way to learn for the ANN. The task involves controlling the complexity by adjusting the number of weights of the ANN. The use of GA for ANN learning can be viewed as follows:

1. Search for the optimal set of weights
2. Search over topology space
3. Search for optimal learning parameters
4. A combination to search for all the above [18]

The fundamental work in this area was done by Holand, Rechenberg, Schwefel and Fogel during the 1970s [19]. Much of the research has focused on the training of feed forward networks [20] [21]. Miller et al, reported that evolutionary algorithm (EA), which is a slight variation of GA, is a better candidate than other standard neural network techniques, because of the nature of the error surface[22] [23]. Those characteristics pointed out by miller are

1. The architecture surface is infinitely large, hence unbounded for possible number of nodes and connections
2. The surface is non-differentiable since changes in the number of nodes and connections are discrete
3. The surface is complex and noisy since the mapping from the architecture to the performance is indirect, strongly epistasis, and dependent on the evaluation method used.
4. The surface is deceptive since similar architectures may have quite different results
5. The surface is multi-modal since different architectures may have similar performance

The steps in genetic algorithm are described as follows:

Step 1: Initialize all the hidden layer weights using a uniform distribution of a closed interval range of [-1, +1]. A sample genotype for the lower half gene from the population pool for *n* input, *h* hidden units, *m* output, and *p* number of patterns can be written as $\left| w_{11} w_{12} \ldots w_{1n} w_{21} w_{22} \ldots w_{2n} \ldots w_{h1} w_{h2} \ldots w_{hn} \right|$ where, range(w) initially is set between the closed interval [-1 +1]. Also, the sample genotype will vary depending on the connection type as described later.

Step 2: The fitness for the population is calculated based on the phenotype and the target for the ANN.

$$netOutput = f(hid * weight)$$

where **hid** is the output matrix from the hidden layer neurons, **weight** is the weight matrix output neurons and f is the sigmoid function

$$RMSError = \sqrt{\frac{\sum_{i=1}^{n} (netOutput - net)^2}{n * p}}$$

$$popRMSError_i = norm(RMSError_i)$$

norm function normalized the fitness of the individual, so the fitness of each individual population is forced to be within certain range.

Step 3: Generate a random number *x* from a Gaussian distribution of mean 0 and standard deviation 1.

```
If ($x$ < $crossOverRate$)
    Select two parents from the intermediate population
    ApplyCrossOver
End If
Generate another random number $y$ from the same distribution
If ($y$ < $mutationRate$)
    ApplyMutation
End If
```

The crossover method that is used for this algorithm is known as linear interpolation with two point technique. Let's consider two genes $w_1 w_2 .. w_h$ and $w_1^{/} w_2^{/} ... w_h^{/}$

Two points are selected randomly, lets assume *point1* and *point2*, where *point1*<*point2*, and *point1*>1, *point2*<*h*

The two child after the crossover operation will be

$$\frac{2w_1 + w_1^{/}}{3}\,\frac{2w_2 + w_2^{/}}{3} \ldots \frac{2w_{point1} + w_{point1}^{/}}{3}\,\frac{w_{point1+1} + 2w_{point1+1}^{/}}{3} \ldots \frac{w_{point2\text{-}1} + 2w_{point2\text{-}1}^{/}}{3}\,\frac{2w_{point2} + w_{point2}^{/}}{3} \ldots \frac{2w_h + w_h^{/}}{3}$$

$$\frac{w_1 + 2w_1^{/}}{3}\,\frac{w_2 + 2w_2^{/}}{3} \ldots \frac{w_{point1} + 2w_{point1}^{/}}{3}\,\frac{2w_{point1+1} + w_{point1+1}^{/}}{3} \ldots \frac{2w_{point2-1} + w_{point2-1}^{/}}{3}\,\frac{w_{point2} + 2w_{point2}^{/}}{3} \ldots. \frac{w_h + 2w_h^{/}}{3}$$

For mutation, a small random value between 0.1 and 0.2, is added to all the weights. Let us assume a parent string $w_1 w_2 .. w_h$. After mutation the child string becomes $w_1 + \varepsilon \mid w_2 + \varepsilon \mid .. \mid w_h + \varepsilon$, where ε is a small random number [0.1 0.2], generated using a uniform distribution.

Step 4: If the convergence for the GA is not satisfied then goto step 2.2 Manuscript Preparation

2.2 Simulated Annealing

In this section we will describe the Simulated Annealing method. Let us consider the following global optimization problem:

$$\textit{minimize } f(x) \text{ subject to } x \in X \tag{1}$$

where $X \subset R^n$ is a compact set. We describe a version of the simulated annealing (SA) method and its pseudo-code for solving this problem.

Simulated annealing [24-27] is one of the few successful stochastic methods for the practical large-scale problems. Numerical experiments show that SA is successful for many discrete optimization problems. However, for some continuous optimization problems in high-dimensional space SA meets difficulties.

Simulated annealing method differs from the traditional descent methods in that local search algorithm for a neighbourhood solution search allows not only downhill moves, while in an attempt to escape from it allows occasional uphill moves as well. The name "simulated annealing" comes from a physical process called annealing, the process for growing crystals.

Starting with an initial solution x, and an initial “temperature’’ T0, which is a parameter, we obtain a neighbouring solution x' and compare its cost with that of x. If the cost of x' ' is smaller than that of x, i.e. $f(x') < f(x)$, we accept the new solution x'. The same thing would happen if we were applying the local descent method. On the other hand, if $f(x')$ is greater than $f(x)$ (in which case any local descent algorithm will not accept x'), the SA algorithm may accept x', but with a probability $e^{-\frac{\Delta_{x'x}}{T_0}}$ where $\Delta_{x'x}$ is the difference in the costs of x' and x, i.e. $\Delta_{x'x} = f(x') - f(x)$. This process is carried out for a certain number of times, which we call iterations, for each temperature. Then we reduce the temperature according to a particular schedule, and repeat. An essential element of the SA algorithm is the probability $e^{-\frac{\Delta_{x'x}}{T_0}}$ of an uphill move of size $\Delta_{x'x}$ being accepted when the current temperature is T. This is dependent on both $\Delta_{x'x}$ and T. For a fixed temperature T, smaller uphill moves $\Delta_{x'x}$ have a higher probability of being accepted. On the other hand, for a particular uphill move $\Delta_{x'x}$, a higher temperature results in a higher probability for that uphill move to be accepted. In the words of [27], at a high temperature any uphill move might be indiscriminately accepted with a high probability so that the objective function and the tumbles around the space are not very important; as T is decreasing the objective function becomes more and more significant; until as T goes to zero the search becomes trapped in the lowest minima that it has reached. Simulated Annealing algorithm for solving a practical problem is typically implemented in two nested loops: the outer loop and the inner loop. The outer loop controls temperatures, while the inner loop iterates a fixed number of times for the given temperature. The inner loop is for the problem of specific decisions. The decisions of the outer loop involve the setting of initial temperature (T_0), the cooling schedule, the temperature length, which is the number of outer loop iterations performed at each temperature, as well as the stopping criterion of the outer loop. The inner loop of SA typically consists of the following parts: feasible solution space, initial feasible solution, neighbourhood move, objective function values, and the decision, which decides whether the decision is found acceptable or probability acceptable according to the so-called Metropolis criterion. Denote *renew* the counts of the solution being accepted in the inner loop, *N_factor* as an input parameter, which can be any positive integer, and *frozen_num* the stopping condition for the outer loop.

The strength of the simulated annealing is that it can deal with highly nonlinear models, chaotic and noisy data and many constraints. It is a robust and general technique. Its main advantages over other local search methods are its flexibility and its ability to approach global optimality. The algorithm is quite versatile since it does not rely on any restrictive properties of the model. The other advantage is that, it allows not only downhill moves while in an attempt to escape from local minima, occasionally it also allow uphill moves. Hence it doesn't get stuck to any narrow or broad local minima and can improve it further.

2.3 Cutting Angle Method

In this section we will describe the Cutting Angle method. The cutting angle method is based on theoretical results in abstract convexity [15]. The method calculates the value of the objective function at certain points. The points are selected in such a way that the algorithm does not return to unpromising regions where function values are high. The new point is chosen where the objective function can potentially take the lowest value. The function is assumed to be Lipschitz, and the value of the potential minim is calculated based on both the distance to the neighboring points and function values at these points. Let us consider the following global optimization problem:

$$\textit{minimize } f(x) \text{ subject to } x \in S \tag{2}$$

where the objective function f is an increasing positively homogeneous of degree one and the set S is the unit simplex in R^n .:

$$S = \left\{ x \in R_+^n : \sum_{i=1}^{n} x_i = 1 \right\} \tag{3}$$

where $R_+^n = \{x \in R^n : x_i \geq 0, i = 1,....,n\}$.

A function f defined R_+^n is called increasing if $x \geq y$ implies $f(x) \geq f(y)$. The function f is positively homogeneous of degree one if $f(\lambda, x) = \lambda f(x)$ for all $x \in R_+^n$ and $\lambda > 0$.

For a given vector $l \in R_+^n, l \neq 0$, we consider $I(l) = \{i = 1,....,n : l_i > 0\}$. We use the following notation for $c \in R$ and $l \in R_+^n$:

$$(c/l)_i = \begin{cases} c/l_i \; if \; i \in I(l) \\ 0 \; if \; i \notin I(l) \end{cases} . \tag{4}$$

An IPH function is nonnegative on R_+^n. We assume that $f(x) > 0$ for all $x \in S$. It follows from the positiveness of f that $I(l) = I(x)$ for all $x \in S$ and $l=f(x)/x$. Let e^k be the kth orthant vector.

The cutting angle method is as follows:

Step0: Initialization: Consider the points $x^k \in S$, $k=1,\ldots,m$, where $m \geq n$, $e^k = x^k \; for \, k = 1,...,n$ and $x^k \geq 0 \; for \, k = n+1,....,m.$ Let $l^k = f(x^k)/x^k, k = 1,...,m.$ Define the function h_m :

$$h_m(x) = \max_{k \le m} \min_{i \in I(l^k)} l_i^k x_i = \max\left\{ \max_{k \le n} l_k^k x_k, \max_{n+1 \le k \le m} \min_{i \in I(l^k)} l_i^k x \right\} \quad (5)$$

And set $j=m$.

Step1: Find a solution x^* for the problem

$$\textit{minimize}\, h_j(x) \text{ subject to } x \in S. \quad (6)$$

Step2: Set $j = j+1$ and $x^j = x^*$.

Step3: Compute $l^j = f(x^j)/x^j$, define the function

$$h_j(x) = \max\left\{ h_{j-1}(x), \min_{i \in I(l^j)} l_i^k x_i \right\} \equiv \max_{k \le j} \min_{i \in I(l^k)} l_i^k x_i \quad (7)$$

And go to Step 1.

3 Experimental Result

Experiments were conducted using the following real-world benchmark data sets from UCI Machine Learning repository: Austral, Breast cancer (Wisconsin) and Heart Disease (Cleveland) and Diabetes data. The details of these datasets can be obtained from the UCI website. The datasets are described in Table 1.

Table 1: Dataset details

Dataset	Instances	Class	Attribute
Austral	690	2	14
Wisconsin Breast Cancer Databases	699	2	9
Heart Disease Cleveland	297	2	13
Diabetes	768	2	8

The results are compared in terms of test classification accuracy and computation time. The following tables (Table 2 & 3) show the classification accuracy and the time complexity of the ANN in percentage for all methods and data sets.

Table 2: Classification Accuracy results for all data sets

	Classification Accuracy [%]		
Dataset	GA	SA	CA
Austral	88.5	89	92.2
Breast Cancer	96.5	98.8	100
Cleveland	89.7	87.5	89.7
Diabetes	82.3	79.8	81.5

Table 3: Time Complexity results for all data

Dataset	CPU Time [s]		
	GA	**SA**	**CA**
Austral	89	75.4	85.6
Breast Cancer	75	69.8	70.3
Cleveland	40	35.5	45
Diabetes	51	46.5	49.8

3 Analysis

The following figures (Figure 1, Figure 2) show a comparison of classification accuracy and the time complexity for the three methods. From Figure 1 it is clear that CA performed more efficiently compare to SA for all datasets. But GA performed slightly better in case of diabetes dataset.

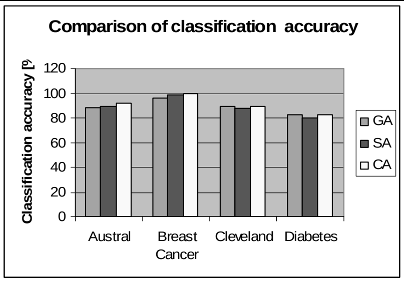

Fig1: Comparison of classification accuracy

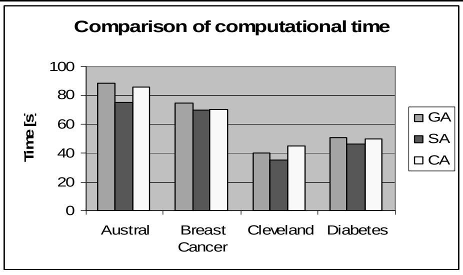

Fig2: Comparison of computational time

Figure 3 shows the convergence of GA, SA, and CA in astral dataset. Figure 3 shows that SA has converged much quicker than GA and CA. This is because the stopping criterion of SA was number restricted to number of iteration, because each iteration in SA takes long time to converge. GA has taken much longer time to converge compare to CA.

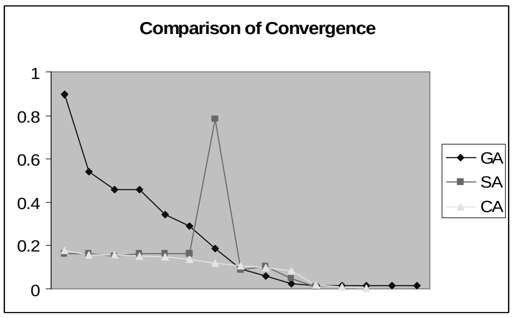

Figure3: Comparison of Convergence

4 Conclusion

This paper presents a comparative analysis of probabilistic and deterministic global search method to find neural network weights. The results show that both Cutting angle method, and Genetic algorithm performed much better than Simulated annealing method for all the dataset. While we compare Genetic algorithm with Cutting angle method, we see that that Cutting angle method performed slightly better that Genetic algorithm in most of the cases. For diabetes and Heart Disease dataset Genetic algorithm performed slightly better than Cutting angle method.

REFERENCES

[1] Whittle, P.: Prediction and regularization by linear least square methods", Van Nostrand, Princeton, N.J. (1963).

[2] Goggin, S. D., Gustafson, K.E., and Johnson, K. M.: An asymptotic singular value decomposition analysis of nonlinear multilayer neural networks. International Joint Conference on Neural Networks, (1991), I-785-I-789,.

[3] Burton, S. A.: A matrix method for optimizing a neural network, Neural comput. Vol 3, no 3.

[4] Lawrence, S., Giles, C. L., Tsoi, A. C.: What size neural network gives optimal generalization? Convergence properties of backpropagatioin. UMIACS-TR-96-22.

[5] Duch, W. and Korczak, J.: Optimization and global minimization methods suitable for neural networks. Neural computing surveys (1999).

[6] Phansalkar V. V.and Thathachar, M. A. L.: Local and Global Optimization Algorithms for Generalized Learning Automata. Neural Computation, Vol. 7. (1995), 950-973.

[7] Sexton, R., Dorsey R., and Johnson, J.: Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. European Journal of Operational Research, Vol. 114. (1999), 589-601.

[8] Sexton, R., Dorsey R., and Johnson, J.: Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation. Decision Support Systems. Vol. 22. (1998), 171-185.

[9] Sexton, R., Dorsey R., and Johnson, J.: Beyond Backpropagation: Using Simulated Annealing for Training Neural Networks. Journal of End User Computing, Vol. 11. (1999), 3.

[10] Y. Shang and B. W. Wah, Global optimization for neural network training, Computer, 29 (1996), pp. p45(10).

[11] PintÈr, J.: Global optimization in action : continuous and Lipschitz optimization--algorithms, implementations, and applications. Kluwer Academic Publishers, Dordrecht ; Boston, (1996).

[12] Trn, A., and Zhilinskas, A.: Global optimization, Springer-Verlag, Berlin ; New York, (1989).

[13] Zhang, X. M., and Chen, Y. Q.: Ray-guided global optimization method for training neural networks, Neurocomputing, Vol. 30. (2000), 333-337.

[14] Zhang, X.-S.: Neural networks in optimization. Kluwer Academic Publishers, Boston, Mass., (2000).

[15] Rubinov, A. M.: Abstract convexity and global optimization. Kluwer Academic Publishers, Dordrecht ; Boston, (2000).

[16] Andramonov, M., Rubinov, A.,and Glover, B.: Cutting angle methods in global optimization. Applied Mathematics Letters, Vol. 12 (1999), 95-100.

[17] Bagirov, A., and Rubinov, A.: Global minimization of increasing positively homogeneous function over the unit simplex, Annals of Operations Research, Vol. 98 (2000), 171-187.

[18] Petridis, V., Kazarlis, S., Papaikonomu, A., and Filelis, A.: A hybrid genetic algorithm for training neural network. Artificial Neural Networks, Vol. 2. (1992), 953-956.

[19] Rechenberg, I.: Cybernatic solution path of an experimental problem. Royal Aircraft Establishment, Library translation no. 1122, Farnborough, Hants, U.K, Aug, (1965).

[20] Whitley, D., Starkweather, T., and BoEArt, C. Genetic algorithms and neural networks - optimizing connections and connectivity. Parallel Computing, Vol. 14, (1990). 347-361.

[21] Montana, D. , and Davis, L.: Training feedforward neural networks using genetic algorithms. Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89, Vol. 1, (1989).

[22] Frean, M.: The upstart algorithm: a method for constructing and training feed-forward neural networks. Neural computation, Vol. 2, (1990).

[23] Roy, A., Kim, L.S., andMukhopadhyay, S.: A polynomial time algorithm for the construction and training of a class of multiplayer perceptrons. Neural networks, Vol. 6, (1993).

[24] Hedar, A.R. and Fukushima, M.: Hybrid Simulated Annealing and Direct Search method for nonlinear unconstrained global optimization. Optimization Methods and Software, Vol. 17, No. 5, (2002). 891-912.

[25] Brooks, D. G. and Verdini, W.A.: Computational experience with generalized simulated annealing over continuous variables. American Journal of Mathematical and Management Sciences, Vol. 8. (1988). 425-449.

[26] Cardoso, M. F., Salcedo, R. L., and de Azevedo, S.F.: The simplex-simulated annealing approach to continuous non-linear optimization. Journal of Computers and Chemical Engineering, Vol. 20 (1996). 1065-1080.