Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.2, 2012

Comparison of Hybrid PSO-SA Algorithm and Genetic Algorithm for Classification

S. G. Sanjeevi[1*] A. Naga Nikhila[2] Thaseem Khan[3] G. Sumathi[4]

1. Associate Professor, Dept. of Comp. Science & Engg., National Institute of Technology, Warangal, A.P., India
2. Dept. of Comp. Science & Engg., National Institute of Technology, Warangal, A.P., India
3. Dept. of Comp. Science & Engg.,National Institute of Technology, Warangal, A.P., India
4. Dept. of Comp. Science & Engg., National Institute of Technology, Warangal, A.P., India

* E-mail of the corresponding author: sgsanjeevi@yahoo.com

Abstract

In this work, we propose and present a Hybrid particle swarm optimization-Simulated annealing algorithm and compare it with a Genetic algorithm for training respectively neural networks of identical architectures. These neural networks were then tested on a classification task. In particle swarm optimization, behavior of a particle is influenced by the experiential knowledge of the particle as well as socially exchanged information. Particle swarm optimization follows a parallel search strategy. In simulated annealing uphill moves are made in the search space in a stochastic fashion in addition to the downhill moves. Simulated annealing therefore has better scope of escaping local minima and reach a global minimum in the search space. Thus simulated annealing gives a selective randomness to the search. Genetic algorithm performs parallel and randomized search. The goal of training the neural network is to minimize the sum of the squares of the error between the target and observed output values for all the training samples and to deliver good test performance on the test inputs. We compared the performance of the neural networks of identical architectures trained by the Hybrid particle swarm optimization-simulated annealing and Genetic algorithm respectively on a classification task and noted the results obtained. Neural network trained by Hybrid particle swarm optimization-simulated annealing has given better results compared to the neural network trained by the Genetic algorithm in the tests conducted by us.

Keywords: Classification, Hybrid particle swarm optimization-Simulated annealing, Simulated Annealing, Genetic algorithm, Neural Network etc.

1. Introduction

Classification is an important activity of machine learning. Various algorithms are conventionally used for classification task namely, Decision tree learning using ID3[1], Concept learning using Candidate elimination [2], Neural networks [3], Naïve Bayes classifier [4] are some of the traditional methods used for classification. Ever since back propagation algorithm was invented and popularized by [3], [5] and [6] neural networks were actively used for classification. However, since back-propagation method follows hill climbing approach, it is susceptible to occurrence of local minima. Hence we examine the use of alternative methods for training neural networks. We examine the use of i) Hybrid particle swarm optimization-simulated annealing algorithm ii) Genetic algorithm to train the neural networks. We study and compare the performance of the neural networks trained by these algorithms on a classification task.

2. Architecture of Neural Network

Neural network designed for the classification task has the following architecture. It has four input units, three hidden units and three output units in the input layer, hidden layer and output layer respectively. Sigmoid activation functions were used with hidden and output units. Figure 1 shows the architectural diagram of the neural network. Neurons are connected in the feed forward fashion as shown. Neural

Network has *12* weights between input and hidden layer and *9* weights between hidden and output layer. So the total number of weights in the network are *21*.

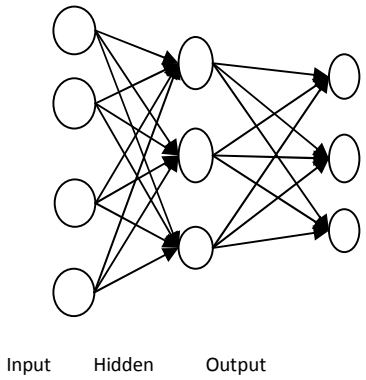

Figure 1 Architecture of Neural Network used

2.1 .Iris data

Neural Network shown in figure 1 is used to perform classification task on IRIS data. The data was taken from the Univ. of California, Irvine (UCI), Machine learning repository. Iris data consists of 150 input-output vector pairs. Each input vector consists of a 4 tuple having four attribute values corresponding to the four input attributes respectively. Based on the input vector, output vector gives class to which it belongs. Each output vector is a 3 tuple and will have a '1' in first, second or third positions and zeros in rest two positions, thereby indicating the class to which the input vector being considered belongs. Hence, we use 1-of-n encoding on the output side for denoting the class value. The data of 150 input-output pairs is divided randomly into two parts to create the training set and test set respectively. Data from training set is used to train the neural network and data from the test set is used for test purposes. Few samples of IRIS data are shown in table 1.

Table 1.Sample Of Iris Data Used.

S. No.	Attr. 1	Attr. 2	Attr. 3	Attr. 4	Class 1	Class 2	Class 3
1	0.224	0.624	0.067	0.043	1	0	0
2	0.749	0.502	0.627	0.541	0	1	0
3	0.557	0.541	0.847	1	0	0	1
4	0.11	0.502	0.051	0.043	1	0	0
5	0.722	0.459	0.663	0.584	0	1	0

3. Genetic Algorithm to Train the Neural Network

Error back propagation algorithm is conventionally used to train the neural networks. Error-Back propagation algorithm uses gradient descent search which is based on the concept of hill climbing. Main disadvantage of neural network using back propagation algorithm is that since it uses hill climbing approach it can get stuck at local minima.

Hence, here we explore the usage of alternative algorithms instead of conventional backpropagation algorithm to optimize the performance of training a neural network and compare them.

The objective function required to be minimized for training the neural network is the error function *E.*

We define the error function E as follows. The error function E is given by

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{x \in X}\sum_{k \in outputs}(t_{kx} - o_{kx})^2 \qquad (1)$$

where x is a specific training example and X is the set of all training examples, t_{kx} denotes the target output for the k_{th} output neuron corresponding to training sample x, o_{kx} denotes the observed output for the k_{th} output neuron corresponding to training sample x. Hence, error function computed is a measure of the sum of the squares of the error between target and observed output values for all the output neurons, across all the training samples.

Genetic algorithms (GAs) were proposed and enunciated by John Holland at the University of Michigan and popularized by David Goldberg [9, 10]. They have the characteristics of parallel search being undertaken through potential solutions with a random exploration of the search space. They use the evolutionary operators of selection, crossover and mutation on the population of the potential solutions. Unlike backpropagation, GAs are stochastic and hence do not get into the problem of local minimum. Genetic algorithms can be used for optimization. In the problem we have to optimize the error function defined in equation (1) by minimizing it. Hence, the objective function we use is the *error function* in equation (1) which needs to be *minimized.* To do this we define the fitness function f as $\boldsymbol{A - E}$ where A is a positive number which is appropriately selected to be sufficiently large so that $\boldsymbol{A - E}$ is positive during the search task undertaken through GA. Task of minimizing the error function is converted to maximizing the value of $\boldsymbol{A - E}$ which is the **fitness function** f used with the genetic algorithm during the training of the neural network.

We are using the genetic algorithm for the purpose of efficiently training the neural network. The idea of efficiently training the neural network is achieved if the neural network predicts with higher accuracy the test outputs corresponding to the test inputs given to the network. GAs use set of binary strings called chromosomes to form a population of binary strings. Each *chromosome* which is a binary string is a potential solution to the network. Hence, each chromosome needs to encode the set of weights present in a neural network. The neural network shown in figure 1 has *21* weights in it. Hence each member of the population (chromosome) needs to encode these complete set of *21* weights so that we can evaluate the fitness function defined above. Below we describe the encoding of the weights for the neural network shown in figure 1 for each chromosome.

3.1.Encoding of the Weights in the Neural Network

The Neural Network in figure 1 has *12* weights between input and hidden layer and *9* weights between hidden and output layer. So the total number of weights in the network are *21*. We coded these weights in binary form. Each weight was assumed to vary between *+12.75* and *-12.75* with a precision of *0.05*. This is because weights learned by a typical neural network will be small positive or negative real numbers. Any real number between *-12.75* and *+12.75* with a precision of *0.05* can be represented by a *9*-bit binary string. Thus, *-12.75* was represented with a binary string *'000000000'* and *+12.75* was represented with *'111111111'*. For example, number *0.1* is represented by *'100000010'*. Each of the twenty one weights present in the neural network was represented by a separate *9* bit binary string and the set of all the weights present in the neural network is represented by their concatenation having *189* bits binary string. These *189* bits represent one possible assignment of values to the weights in the neural network.

In the initial population a set of *50* randomly generated binary strings were created, each binary string having *189* bits. Different binary strings in population represent different potential solutions to the problem of training the neural network. Each binary string represents one possible assignment of weights to the neural network.

3.2. Genetic Algorithm

Here we describe the Genetic algorithm used for training the neural network.

1. A set of *50* binary strings, each string having *189* bits were randomly created to form the initial

population.

2. Evaluate the **fitness function** *f* for each of the binary strings in the present population *P*.
3. **Elitism:** Two of the binary strings with highest fitness values from the population are carried over to the new population being created $P_{new.}$
4. **Selection:** The selection operation selects the parent strings from the present population for the crossover operation. This is done using the selection probability $p_i = f_i / \Sigma f_i$. The roulette wheel concept is used to make the selection of binary strings. The probability p_i *is* determined as the ratio of fitness of a given binary string with the sum of fitness values for all the *50* binary strings in the population *P.*
5. **Crossover:** Pairs of chromosomes were selected from population *P* using selection probability defined above and a single point crossover is performed on the pair to form two offspring. Selection probabilities encourage the selection of high fitness individuals to be selected as a pair. Crossover point was chosen at random and the two strings are interchanged at the point. This process was repeated for each of the 24 pairs of strings selected. Selection of each pair was done using selection probability calculations as explained above. Strings generated after crossover are kept in the new population being created P_{new}.
6. **Mutation:** Mutation involves flipping a randomly selected bit. Mutation was done with a probability of *5%* on the strings in P_{new} . In the string selected for mutation, a randomly selected bit is flipped.
7. On completing above step the new population P_{new} is formed. Make the new population created P_{new} as the present population *P* for the next iteration of *GA* ie. Set $P \leftarrow P_{new}$. Repeat steps *2* to *7* for sufficient number of iterations. Steps *2* to *7* were repeated for *15000* iterations in our work. The string with the highest fitness among the population is returned as the *solution* and gives the *assignment of weights* to the *neural network* after training through the Genetic Algorithm.

4. Hybrid PSO - SA Algorithm

In particle swarm optimization [11, 12] a swarm of particles are flown through a multidimensional search space. Position of each particle represents a potential solution. Position of each particle is changed by adding a velocity vector to it. Velocity vector is influenced by the *experiential knowledge* of the particle as well as *socially exchanged information.* The *experiential knowledge* of a particle *A* describes the distance of the particle *A* from its own best position since the particle *A*'s first time step. This best position of a particle is referred to as the *personal best* of the particle. The *global best position* in a swarm at a time *t* is the best position found in the swarm of particles at the time *t.* The socially exchanged information of a particle *A* describes the distance of a particle *A* from the global best position in the swarm at time *t*. The experiential knowledge and socially exchanged information are also referred to as *cognitive* and *social components* respectively. We propose and present here the *hybrid PSO-SA* algorithm in table 2.

Table 2. Hybrid Particle Swarm Optimization-Simulated Annealing Algorithm

Create a n_x dimensional swarm of n_s particles;
repeat
for *each particle i* = 1, . . .,n_s **do**
// $\mathbf{y}_i$ denotes the *personal best* position of the particle *i* so far
// set the *personal best* position
if $f(\mathbf{x}_i) < f(\mathbf{y}_i)$ **then**

$\mathbf{y}_i = \mathbf{x}_i$;
end

// $\hat{\boldsymbol{y}}$ denotes the *global best* of the swarm so far
// set the *global best* position
if $f(\mathbf{y}_i) < f(\hat{\mathbf{y}})$ **then**
$\hat{\mathbf{y}} = \mathbf{y}_i$;
end
end

for *each particle* $i = 1, \ldots, n_s$ **do**
update the *velocity* $\boldsymbol{v}_i$ of particle i using *equation (2);*

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + c_1 r_1(t)[\boldsymbol{y}_i(t) - \boldsymbol{x}_i(t)] + c_2 r_2(t)[\hat{\boldsymbol{y}}(t) - \boldsymbol{x}_i(t)] \quad (2)$$

// where $\boldsymbol{y}_i$ denotes the personal best position of the particle i
// *and* $\hat{\boldsymbol{y}}$ *denotes the global best position of the swarm*
// *and* $\boldsymbol{x}_i$ *denotes the present position vector of particle i.*

update the *position* using *equation (3)*;
$\boldsymbol{b}_i(t) = \boldsymbol{x}_i(t)$ *//storing present position*
$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \boldsymbol{v}_i(t+1)$ *(3)*

// applying **simulated annealing**
compute $\Delta E = (f(\mathbf{x}_i(t)) - f(\mathbf{x}_i(t+1))$ *(4)*

// ΔE = (E value for the previous network before weight change) - (E value for the present network with changed configuration of weights).

• **if** ΔE is positive then

// new value of E is smaller and therefore better than previous value. Then

// accept the new position and make it the current position

else accept the new position $\boldsymbol{x}_i(t+1)$ (even though it has a higher E value) with a probability p defined by

$$p = e^{-\Delta E / T} \quad (5)$$

• Revise the temperature T as per schedule defined below.

The temperature schedule followed is defined as follows:

The starting temperature is taken as $T = 1050$;

Current temp = *T/log(iterations+1)*;

Current temperature was changed after every *100* iterations using above formula.

end

until *stopping condition is true* ;

In *equation (2)* of *table 2*, $\boldsymbol{v_i}$,$\boldsymbol{y_i}$ *and* $\hat{\boldsymbol{y}}$ are vectors of n_x dimensions. v_{ij} denotes scalar component *of* $\boldsymbol{v_i}$ in dimension *j*. v_{ij} is calculated as shown in *equation 6* where r_{1j} and r_{2j} are random values in the range *[0,1]* and *c1* and *c2* are learning factors chosen as *c1 = c2 = 2*.

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)\ ] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij})\ (t)] \qquad (6)$$

4.1. Implementation details of Hybrid PSO-SA algorithm for training a neural network

We describe here the implementation details of *hybrid PSO-SA* algorithm for training the neural network. There are *21* weights in the neural network shown in figure 1.

Hybrid PSO-SA algorithm combines particle swarm optimization algorithm with simulated annealing approach in the context of training a neural network. The swarm is initialized with a population of *50* particles. Each particle has *21* weights. Each *weight value* corresponds to a *position in a particular dimension* of the particle. Since there are *21* weights for each particle, there are therefore *21* dimensions. Hence, *position vector* of each *particle* corresponds to a *21 dimensional weight vector*. Position (*weight*) in each dimension is modified by adding velocity value to it in that dimension. Fitness function corresponding to a position $\boldsymbol{x_i}$, is $f(\mathbf{x_i})$ and it denotes the distance of position $\boldsymbol{x_i}$ from the final solution. In the *hybrid PSO-SA* algorithm this fitness function $f(\mathbf{x_i})$ is defined as error *function E(**w**)*. Error *function E(**w**)* is a function of the weights in the neural network as defined in equation *(1)*. In the optimization problem of training neural networks fitness function which is error function needs to be minimized.

Each particle's velocity vector is updated by considering the personal best position of the particle, global best position of the entire swarm and the present position vector of the particle as shown in equation *(2)* of table 2. Velocity of a particle *i* in dimension *j* is calculated as shown in equation *(6)*.

Hybrid PSO-SA algorithm combines *pso algorithm* with *simulated annealing* approach. Each of the *50* particles in the swarm is associated with *21* weights present in the neural network. The error function *E(**w**)* which is a function of the weights in the neural network as defined in equation *(1)* is treated as the *fitness function*. Error *E(**w**)* (fitness function) needs to be minimized. For each of the *50* particles in the swarm, the *solution (position)* given by the particle is accepted if the *change in error function* ΔE as defined in *equation 4* is positive, since this indicates error function is reducing in the present iteration compared to the previous iteration value. If $\Delta E$is not positive then new position is accepted with a probability *p* given by formula in *equation (5)*. This is implemented by generating a random number between *0* and *1*. If the number generated is lesser than *p* then the new position is accepted, else the previous position of particle is retained without changing. Each particle's personal best position and the global best position of the swarm are updated after each iteration. *Hybrid PSO-SA algorithm* was run for *15000* iterations. After *15000* iterations global best position *gbest* of the swarm is returned as the solution. Hence, stopping criterion for the algorithm was chosen as completion of *15000* iterations.

Hybrid PSO-SA algorithm combines parallel search approach of *PSO* and selective random search and global search properties of *simulated annealing* [7, 8] and hence combines the advantages of both the approaches.

5. Experiments and Results

We have trained the neural network with architecture shown in figure *1* with genetic algorithm described in section 3 on training set taken from the *IRIS* data. Training set was a subset of samples chosen randomly from the *IRIS* data. Remaining samples from *IRIS* data were included in the test set. Performance was observed on the test data set for predicting the class of each sample.

Table 3. Results Of Testing For Neural Network Using Genetic Algorithm

Sl. No	Samples in Training Set	Samples in Test Set	Correct classifications	Misclassifications
1	100	50	44	6
2	95	55	45	10
3	85	65	53	12
4	75	75	61	14

We have chosen similarly, training sets of varying sizes from the *IRIS* data and included each time the samples which were not selected for training set into test set. Neural network was trained by each of the training sets and tested the performance of the network on corresponding test sets. Training was done for *15000* iterations with each of the training sets using the genetic algorithm. Results are shown in table 3.
We have also trained the neural network with architecture shown in figure *1* with *Hybrid PSO-SA algorithm* described in section 4, with each of the training sets chosen above in table 3 and tested the performance of the network on corresponding test sets. Training was done using *Hybrid PSO-SA algorithm* for *15000* iterations with each of the training sets. The results are shown in table 4.

Table 4. Results Of Testing For Neural Network Using *Hybrid PSO-SA Algorithm*

Sl.No	Samples in Training Set	Samples in Test Set	Correct Classifications	Misclassifications
1	100	50	47	3
2	95	55	51	4
3	85	65	61	4
4	75	75	70	5

Two neural networks with same architecture as shown in figure 1 were used for training and testing with the two algorithms of *Hybrid PSO-SA* algorithm and *Genetic* algorithm respectively. Training was performed for same number of *15000* iterations on each neural network. Same training and test sets were used for comparison of neural networks performance with both the algorithms. Results of experiments and testing point out that neural network trained with *Hybrid PSO-SA* algorithm gives better performance over neural network trained with the Genetic algorithm across the training and test sets used.

6. Conclusion

Our objective was to compare the performance of feed-forward neural network trained with *Hybrid PSO-SA* algorithm with the neural network trained by Genetic algorithm. We have trained the neural networks with identical architectures with the *Hybrid PSO-SA* algorithm and the Genetic algorithm respectively. The task we have tested using neural networks trained separately using these two algorithms is the *IRIS* data classification. We found that neural network trained with *Hybrid PSO-SA* algorithm has given better

classification performance among the two. Neural network trained with *Hybrid PSO-SA* algorithm combines *parallel search* approach of *PSO* and *selective random search* and *global search* properties of *simulated annealing* and hence combines the advantages of both the approaches. GA also uses parallel and randomized search. Hence we compared these two global search approaches. *Hybrid PSO-SA algorithm* has given better results for training a neural network than the *Genetic algorithm*.

References

A Quinlan, J.R.(1986). Induction of decision trees. Machine Learning, 1 (1), 81-106.

Mitchell, T.M.(1977). Version spaces: A candidate elimination approach to rule learning. Fifth International Joint Conference on AI (pp.305-310). Cambridge, MA: MIT Press.

Rumelhart, D.E., & McClelland, J.L.(1986). Parallel distributed processing: exploration in the microstructure of cognition(Vols.1 & 2). Cambridge, MA : MIT Press.

Duda, R.O., & Hart, P.E.(1973). Pattern classification and scene analysis New York: John Wiley & Sons.

Werbos, P. (1975). Beyond regression: New tools for prediction and analysis in the behavioral sciences (Ph.D. dissertation). Harvard University.

Parker, D.(1985). Learning logic (MIT Technical Report TR-47).MIT Center for Research in Computational Economics and Management Science.

Kirkpatrick, S., Gelatt,Jr., C.D., and M.P. Vecchi 1983. Optimization by simulated annealing.Science 220(4598).

Russel, S., &Norvig, P. (1995). Artificial intelligence: A modern approach. Englewood Cliffs, NJ:Prentice-Hall.

Mitchell, T.M., 1997.Machine Learning, New York: McGraw-Hill.

K Goldberg, David E. (1989). Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley. p. 41.ISBN 0201157675.

Kennedy, J., Eberhart, R., 1995. “Particle swarm optimization”, IEEE International Conference on Neural Networks, Perth, WA , (pp.1942 – 1948), Australia.

Engelbrecht, A.P., 2007, Computational Intelligence, England: John Wiley & Sons.

Sriram G. Sanjeevi has acquired B.E. (Electronics and Communication Engineering) from Osmania University, Hyderabad, India in 1981, M.Tech (Computer Science and Engineering) from M.I.T. Manipal, Mangalore University, India in 1991 and Ph.D (Computer Science and Engineering) from IIT, Bombay, India in 2009. He is currently Associate Professor & Head of Computer Science and Engineering Department, National Institute of Technology, Warangal, India. His research interests are Neural networks, Machine learning and Soft computing.

A. N. Nikhila has acquired B.Tech (Computer Science and Engineering) from NIT Warangal, India in 2011. Her research interests are Neural networks, Machine learning.

Thaseem Khan has acquired B.Tech (Computer Science and Engineering) from NIT Warangal, India in 2011. His research interests are Neural networks, Machine learning.

G. Sumathi has acquired B.Tech (Computer Science and Engineering) from NIT Warangal, India in 2011. Her research interests are Neural networks, Machine learning.