International Journal of Engineering Research and Advanced Technology (IJERAT)

ISSN: 2454-6135 **[Volume. 02 Issue.03, March-2016]**

Speeding Up Back-Propagation Learning (SUBPL) Algorithm: A New Modified Back-Propagation Algorithm

Rasha M. Hassoon1, Safana H. Abbas2

1 Department of computer science-College of Education-University of al-mustansiriya, Baghdad, Iraq

(majedrasha937@yahoo.com)

2 Assist. Prof., Department of computer science-College of Education-University of al-mustansiriya, Baghdad, Iraq

(safanahyder@yahoo.com)

Abstract

The convergence speed is the most important feature of Back-Propagation (BP) algorithm. A lot of improvements were proposed to this algorithm since its presentation, in order to speed up the convergence phase. In this paper, a new modified BP algorithm called Speeding up Back-Propagation Learning (SUBPL) algorithm is proposed and compared to the standard BP. Different data sets were implemented and experimented to verify the improvement in SUBPL.

Key Words: Back-Propagation, Cuckoo Search, Neural Network, Speeding up Back-Propagation Learning (SUBPL).

---

1. INTRODUCTION

Artificial neural networks (ANNs) has been used for a wide range of different real-world applications, recently many complex areas of science utilize neural network for different kinds of learning problems. Recent research activities have shown that ANNs have powerful pattern classification and pattern recognition capabilities, inspired by biological systems, particularly by research into human brain. ANNs are able to learn and generalize from experience. Currently ANNs are used for a wide variety of tasks in many different fields of business, industry and science such as sonar target recognition, car navigation, and image compression [1].

Traditional BP algorithms have some disadvantages like slow speed of convergence and being stuck in local minima. The learning rate is relatively small constant that indicates the relative change in weights. If the learning rate is too small, the network will train very slowly, and if the learning is too big, the network may oscillate around minimum point. Overshooting the lowest point with each weight adjustment, but never actually reaching it. Usually the learning rate is very small, with 0.01 not an uncommon number. Some modifications to the BP algorithm allows the learning rate to decrease from a large value during the learning process. It has many advantages, since it is assumed that the network initiates at a state that is distant from the optimal set of weights, training will initially be rapid. As learning progresses the learning rate decreases as it approaches the optimal point in the minima slowing the learning rate near the optimal point encourages the network converge to a solution while reducing the possibility of overshooting [3].

Adjust the weight values of several parameters are of importance in implementing the BP algorithm. The initial value of weights should be small and randomly chosen and avoid the symmetry problem [6].

2. BACK-PROPAGATION ALGORITHM (BP)

The efficient supervised training of feed forward neural networks is a subject of considerable on-going research and numerous algorithms have been proposed to this end**.** The error back propagation method is the most effective and most widely-used learning method for the training of multilayer neural networks(MLP), which is based on error correction learning rule. It is a descending gradient method and was first described in details in 1986 by Hinton, Rumelhart, Williams [1].

Generally, the BP network has two stages, training and testing, in his thesis will speed up the training stage only. The following figure shows the topology of the BP neural network that includes input layer, one hidden layer and output layer. It should be noted that BP neural networks can have more than one hidden layer.

The following Pseudo coding describes the BP algorithm [2-3].

```
Assign all network inputs and outputs:
Initialize all weights with small random numbers typically between -1 and 1
Repeat
    For every pattern in the training set
        Present the pattern to the network
//      propagated the input forward through the network:
        For each layer in the network
            For each node in the layer
                1. Calculate the weight sum of the inputs to the node.
                2. Add the threshold to the sum.
                3. Calculate the activation function for the node
            End
        End
//      propagate the errors backward through the network
        For every node in the output layer calculate the error signal
        End
        For all hidden layers
            For every node in the layer
                1. Calculate the node's single error.
                2. Update each node's weight in the network.
            End
        End
//      calculate global error
        Calculate the error function
        End
While ((maximum number of iterations < than specified) and
```

(Error function is > than specified))

3. THE PROPOSE (SUBPL) ALGORITHM

The SUBPL algorithm is one of the modified of BP algorithm [7]. The methodology that been used in constructing SUBPL algorithm are illustrated in the following steps [7][8]:

1. Trying to get the best set of initial weights.
2. Changing the BP parameters values.
3. Testing the modified algorithm with variable-input logic functions.
4. Comparing the results with other modified BP algorithms.
5. Analyzing, discussing, and summarizing the results.

The following Pseudo coding describes the SUBPL algorithm [1] [2][4][7][8]:

Assign all network inputs and outputs:
//initialize the weight by CS algorithm
Initialize all weights
For each weight in the algorithm
//the upper and lower band depend on the input of the network
Step 1: Generate an initial population of an N host nest I =1... N
Between specific upper and lower band
Step 2: **While (**f min < Max-Generation) or (stop criterion)
Step 3: **Do** Get a Cuckoo randomly by Levy flights and evaluate its fitness f_i.
$x^{t+1} = x_i^t + \alpha \oplus levy(\lambda)$
$levy \backsim u = t^{-\lambda}, \quad 1 < \lambda \leq 3$
Step 4: Choose randomly a nest $_j$ among N.
Step 5: **If** $F_i > F_j$ **Then**
Step 6: Switch j by the new solution, **End If**
Step 7: A segment ($p_{a)}$ of worse nest is abandoned and new ones are built.
Step 8: Keep the optimal solutions (or nest with quality solutions).
Step 9: Rank the solutions and find the current best nest.
Step 10: **end while**
Step 11: $weight = \frac{(best\ nest - 0.5)}{10}$
End

```
    //chose the number of hidden neuron be the following step
        N= calculating the number of input node.
        If the number of input node is small
            Using equation: number of hidden neuron= 2N-1
        Else
            Using equation: number of hidden neuron= N^2
        end
Repeat
    For every pattern in the training set
        Present the pattern to the network
//      propagated the input forward through the network:
        For each layer in the network
            For each node in the layer
                4. Calculate the weight sum of the inputs to the node.
                5. Add the threshold to the sum.
                6. Calculate the activation function for the node
            End
        End
//      propagate the errors backward through the network
        For every node in the output layer calculate the error signal
        End
        For all hidden layers
            For every node in the layer
                3. Calculate the node's single error.
                4. Update each node's weight in the network.
                   (use this learning rate when update the weight :
                    blr = alr/9
                   //Where: alr= learning rate for the hidden layer, is a specific value like 0.11, and
                   (0.1<= alr <=0.2)
                   //blr= learning rate for the output layer
            End
        End
//      calculate global error
        Calculate the error function
        End
While ((maximum number of iterations < than specified) and
        (Error function is > than specified))
```

Where $\alpha>0$ is the step size, which should be related to the scales of the problem of interest. The product $\oplus$ means entry wise multiplication [4].

In this research will apply BP algorithm and SUBPL algorithm to see the different in number of epochs and time between these two algorithms.

4. EXPERIMENTAL RESULTS

To verify the convergence speed of SUBPL algorithm, different logic gates with different inputs were implemented as illustrated below:

4.1. 2-INPUT (XOR, AND)

That illustrate in this paper all the parameter (initial weight, Learning Rate, number of hidden neurons, etc.) and how it changed when 2-XOR input is used[7]. All the parameter can used with 2-input AND because it has the same number of input, when SUBPL algorithm is used.

In the 2-input XOR that the number of epochs is 19 and time is 0.4 seconds when the SUBPL algorithm is used as shown in figure (2), is faster than the number of epochs is 1053 and time is 32.9 seconds when the BP algorithm is used as shown in figure(1) [7].

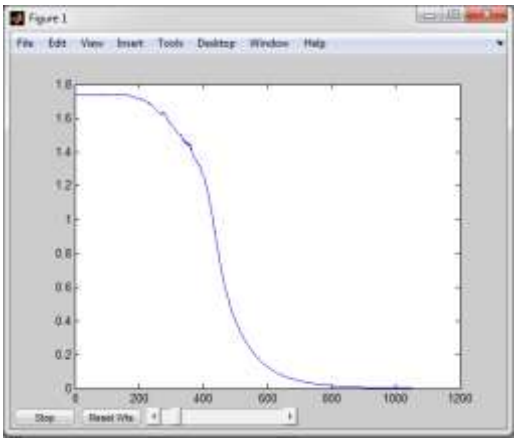


Figure (1):2-XOR used BP algorithm.

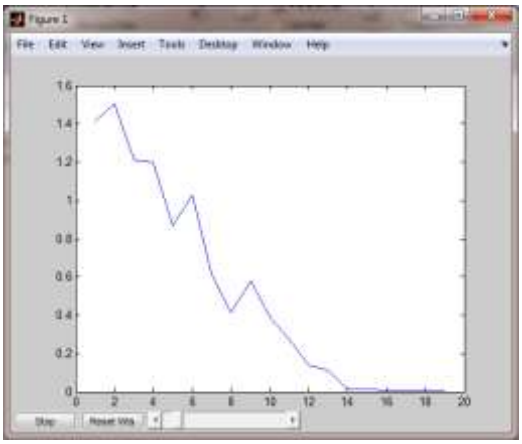


Figure (2): 2-XOR used SUBPL algorithm

In figure (3) and figure (4) is illustrate when used 2-input AND that the number of epochs is 138 and time is 2.7 seconds when the SUBPL algorithm is used, is faster than the number of epochs is 6265 and time is 139.8 seconds when the BP algorithm is used.

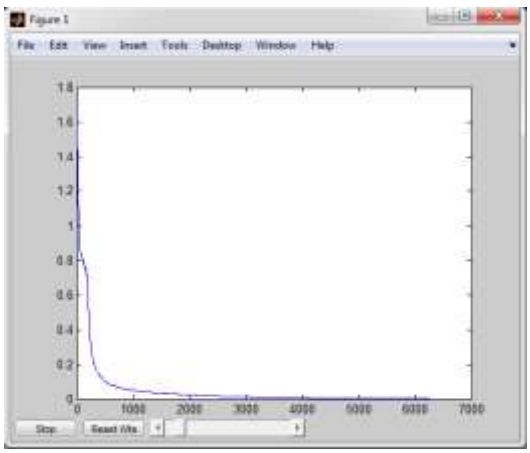
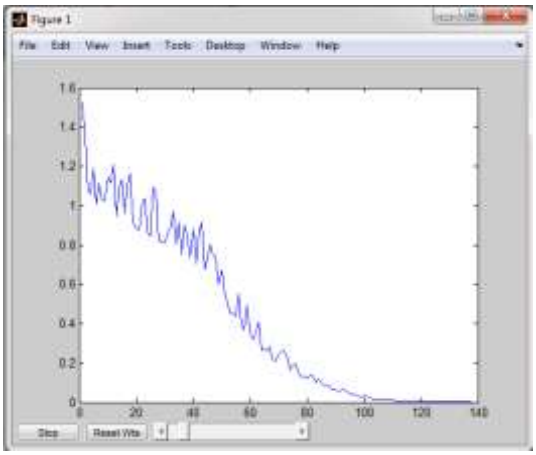


Figure (3): 2-AND used BP algorithm



Figure(4): 2-AND used SUBPL algorithm

4.2 3-Input OR:

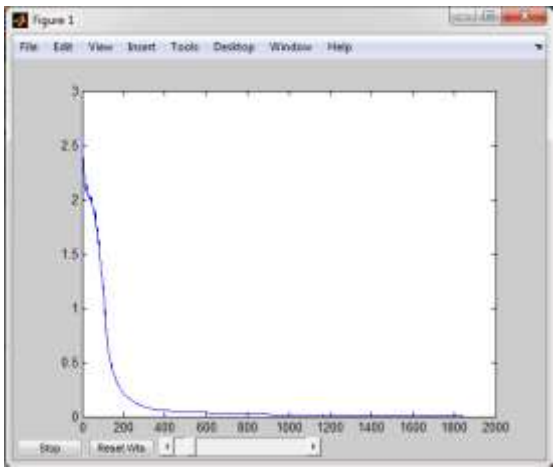
When 3-input OR is used in SUBPL algorithm as data set all the parameter will stay fixed except the:

1. The number of hidden neurons : because the number of input neurons is low then will know the proper number of input by used the equation 2n-1 [8], as follow:

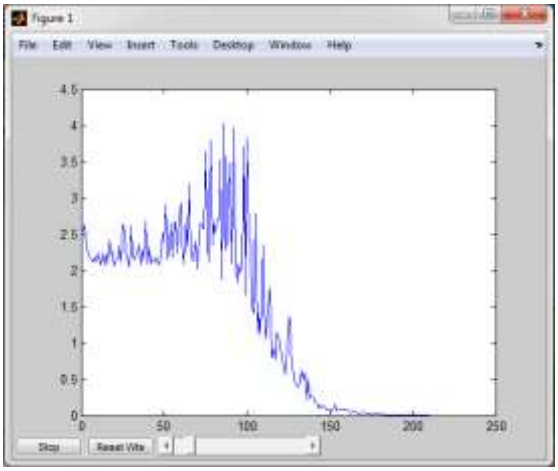
2*n-1 = 2*3-1= 5 is the number of hidden neurons

2. The upper and lower band will be [2 -2] because the number of input is low in the CS algorithm in SUBPL algorithm.

In figure (5) and figure (6) can illustrate when used 3-input OR that the number of epochs is 210 and time is 6.0 seconds when the SUBPL algorithm is used, is faster than the number of epochs is 1844 and time is 39.5 seconds when the BP algorithm is used.



Figure(5): 3-OR using BP algorithm



Figure(6): 3-OR using SUBPL algorithm

4.3 4-input NAND :

When 4-input NAND is used in SUBPL algorithm as data set all the parameter will stay fixed except the:

1. The number of hidden neurons : because the number of input neurons is low then will know the proper number of input by used the equation 2n-1 [8], as follow:

2*n-1 = 2*4-1= 7 is the number of hidden neurons

2. The upper and lower band will be changed from [2 -2] to [5 -5] because the number of input number is bigger than before in the CS algorithm in SUBPL algorithm and it's effect the number of initial weight in SUBPL algorithm.

In figure(7) and figure (8) can illustrate when used 4-input NAND that the number of epochs is 95 and time is 1.6 seconds when the SUBPL algorithm is used, is faster than the number of epochs is 1555 and time is 36.3 seconds when the BP algorithm is used.

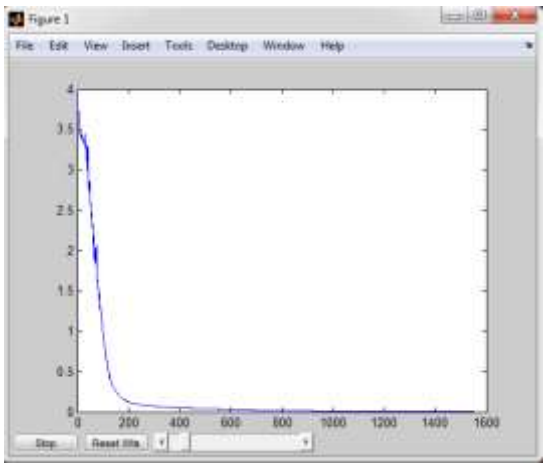


Figure (7):4-NAND using BP algorithm

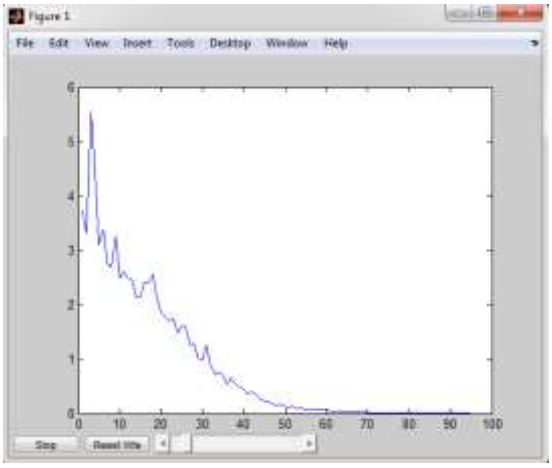


Figure (8):4-NAND using SUBPL algorithm

4.4 Simple Image:

The simple image is used in both BP algorithm and SUBPL algorithm, to find the different number of epochs in both algorithms and which algorithm is faster.

In this simple image:

$$\text{Image}=\begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

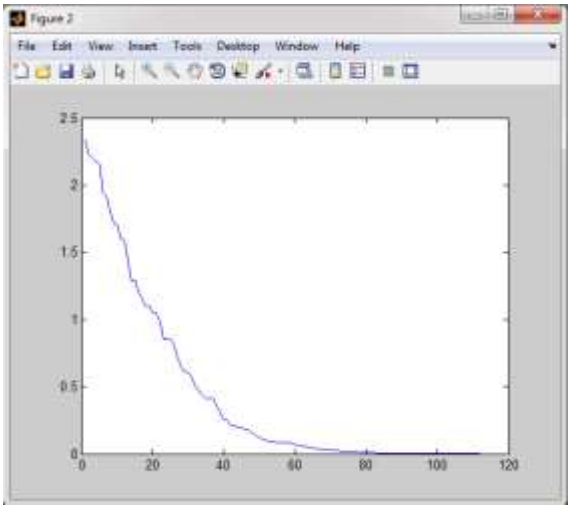
When simple image is used in SUBPL algorithm as data set all the parameter will stay fixed except the:

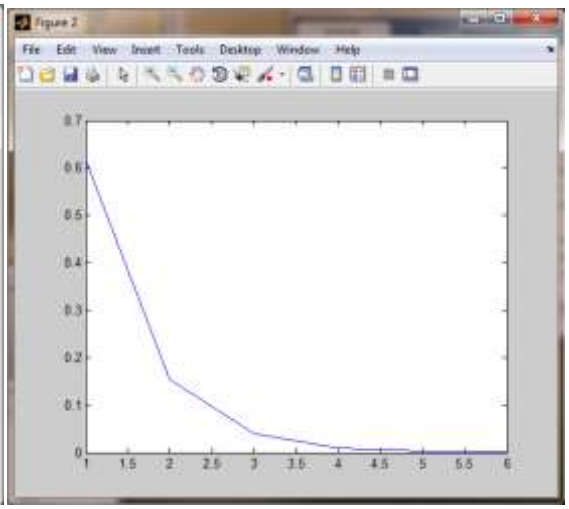
1. The number of hidden neurons: because the number of input neurons is low then will know the proper number of input by used the equation 2n-1 [8], as follow:

2*n-1 = 2*5-1=9 is the number of hidden neurons

2. The upper and lower band will be changed to [2 -2] because the number of input number is low in the CS algorithm in SUBPL algorithm and its effect the number of initial weight in SUBPL algorithm.

In figure (9) and figure (10) can illustrate when used simple image that the number of epochs is 6 and time is 0.8 seconds when the SUBPL algorithm is used, is faster than the number of epochs is 112 and time is 3.4 when the BP algorithm is used.





Figure(9):simple image using BP algorithm **Figure(10):simple image using SUBPL algorithm**

Table (1) : Name of the table ?

Data sets	Number of epochs in the learning stage when use BP algorithm	Time when use BP algorithm (second)	Number of epochs in the learning stage when use SUBPL algorithm	Time when use SUBPL algorithm (second)
2 input XOR	1053	32.9	19	0.4
2 input AND	6265	139.8	138	2.7
3 input OR	1844	39.5	210	6.0
4 input NAND	1555	36.3	95	1.6
Simple image	112	3.4	6	0.8

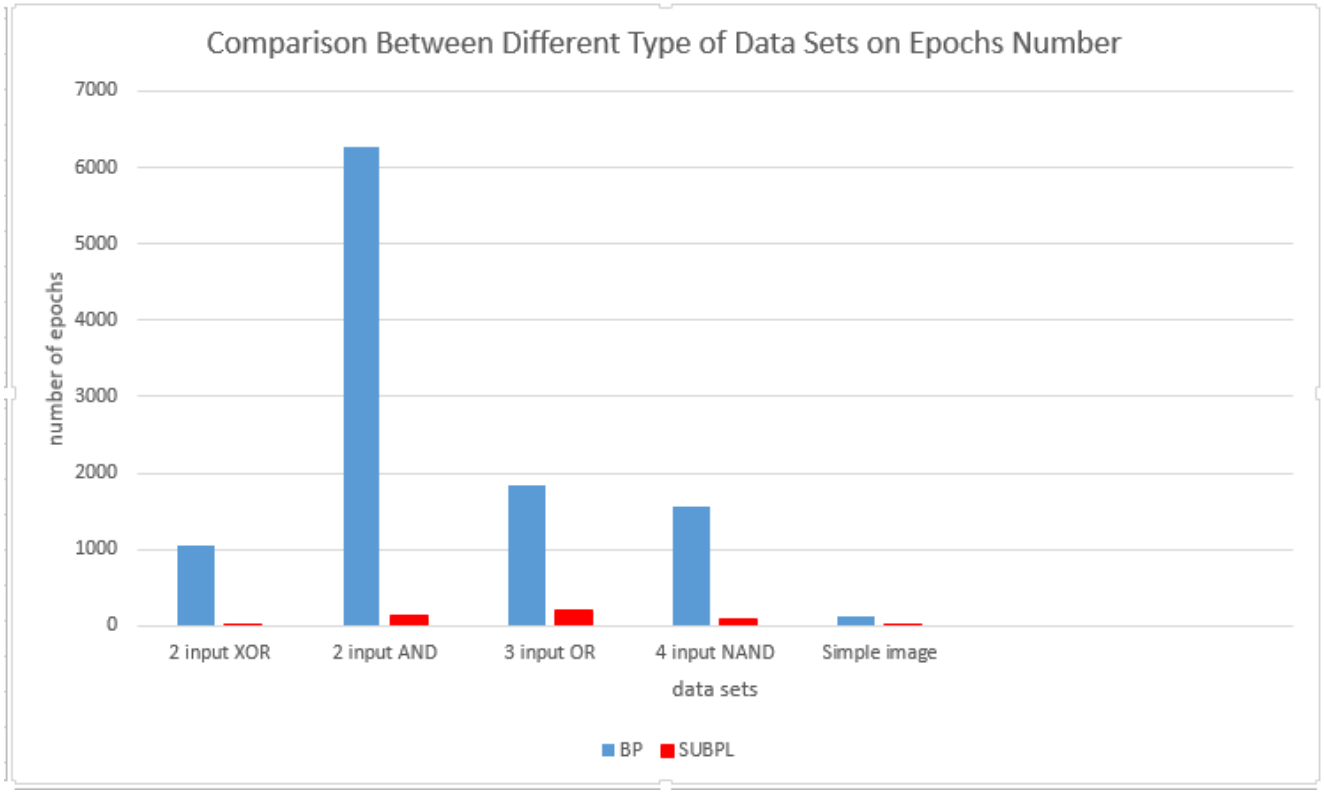


Figure (11): Comparison between different type of data sets in epochs number

As illustrate in the figure (11) and table (1) that the different between algorithms in epochs in different types in data sets is large, so it's clear that SUBPL algorithm is faster than BP in all the data sets.

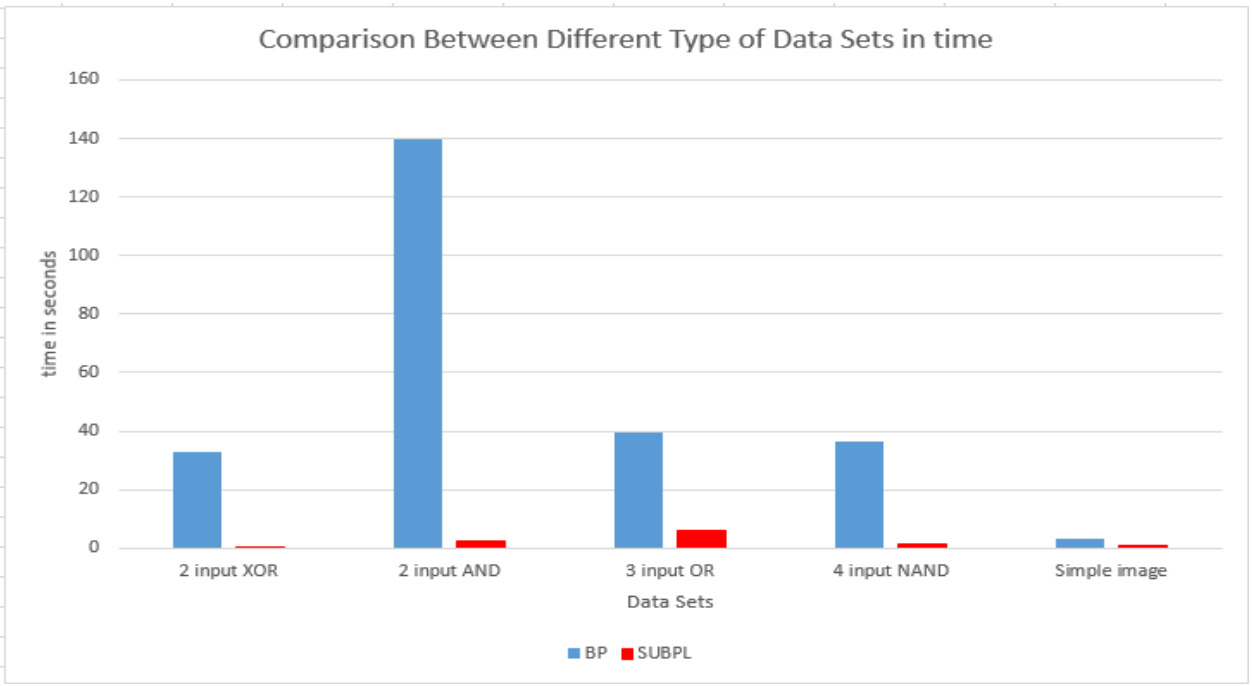


Figure (12): comparison between different types of data sets in time

As illustrate in the figure (12) and table (1) that the different between algorithms in time in different types in data sets is hug, so it's clear that SUBPL algorithm is faster than BP in all the data sets.

5. CONCLUSION:

In this paper, applying different type of data sets in standard Back-Propagation (BP) algorithm and Speeding up Back-Propagation Learning (SUBPL) algorithm and comparing the different in speed. That the SUBPL algorithm is 90% faster than the standard BP algorithm in different type of data sets (*2-Input (XOR, AND), 3-Input OR, 4-input NAND and simple image size (7*5)).*

REFERENCES

[1] Lamia Abdul Lateef Al-Anii (2005), **wind speed forecasting using neural network,** thesis, Supervised By Dr. Salim Ali Abbas Al-Ageele and Dr. Kais Jameel Al-Jumaly, College of science, Al-mustansiriah university, Baghdad, Iraq

[2] Mitchell, T. M. , **Machine Learning** , chapter 4, 1997, Mc Graw-Hill.

[3] Donald R. Tveler , **The Backprop Algorithm**, chapter 2, 2001.

[4] Nazri M. Nawi, Abdullah Khan, and M. Z. Rehman, **A New Back-Propagation Neural Network Optimized with Cuckoo Search Algorithm.** University Tun Hassein on Malaysia (UTHM), September, 2013.

[5] N. W. Nawi , Abdullah and M. Z. Rehman, **A New Levenberg Marquardt Based Back Propagation Algorithm Trained with Cuckoo Search**. Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malasyia, 86400, Johor, Malaysia ,2013.

[6] Dilip Sakar, **Methods to speed up Error Back-Propagation learning algorithm**. University of Miami, ACM Computing Surveys, Vol. 27, No. 4, December, 1995.

[7] Rasha M. Hassoon and Safana H. Abbas (2015**), A Hybrid Cuckoo Search and Back-Propagation Algorithms With Dynamic Learning Rate To Speed Up the Convergence (SUBPL) Algorithm**, Integrated Journal of Engineering Research and Technology, ISSN NO. 2348-6821, vol2 (5).

[8] Rasha M. Hassoon and Safana H. Abbas (2015**), Effect Of Changing Hidden Neurons and Activation Function On Back-Propagation (BP) Speed**, International Journal of Modern Trends in Engineering and Research (IJMTER), *Volume 02, Issue 10,* **ISSN (Online):2349–9745 ; ISSN (Print):2393-8161.**