[Heren Chellam G, 5(8): August, 2016]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 4.116

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

IDENTIFYING OPTIMAL NUMBER OF ORTHONORMALISATION IN LEARNING ALGORITHM USING WEATHER FORECASTING

Dr. Heren Chellam G. *

* Department of Computer Science, Rani Anna Government College for women, Tirunelveli, Tamilnadu, India – 627 008

DOI: 10.5281/zenodo.60844

ABSTRACT

Artificial neural networks are more powerful than any other traditional expert system in the classification of patterns, which are non linear and in performing pattern classification tasks because they learn from examples without explicitly stating the rules. Multilayered feed forward neural networks possess a number of properties, which make them particularly suited to complex problems. Their applications to some real world problems are hampered by the lack of a training algorithm which finds a globally optimal set of weights in a relatively short time. Genetic algorithms are a class of optimization procedures, which are good at exploring a large and complex space in an intelligent way to find values close to the global optimum. In this study, conversion of random weights into orthonormal weights (orthonormalisation) in feed forward network algorithm is proposed and also optimal number of orthonormalisation of weight is evaluated. Orthonormal weight based artificial neural network algorithms not only succeed in its task but also outperforms the genetic algorithm. This success comes from orthonormal weights instead of random and genetic weights. This paper has documented the evolution and ultimate success of this algorithm using weather forecasting data.

KEYWORDS: Artificial neural network, Genetic, Optimal, Orthonormal, Weather forecasting.

INTRODUCTION

Artificial Neural Network (ANN) is an information processing paradigm, inspired by the way the biological nervous systems, such as the brain, process information. ANN is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve specific problems [1]. ANNs, like people, learn from examples. They cannot be programmed to perform a specific task. ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of artificial neural networks as well. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Self-organization in an ANN can create its own organization or representation of the information it receives during learning time, Real Time Operation that is ANN computations may be carried out in parallel. Special hardware devices are being designed and manufactured which take advantage of this capability, some network capabilities may be retained even after major network damage.

ANN takes a different approach to problem solving than that of conventional computers. Computers could do things that we don't exactly know how to do. So ANN cannot be programmed to perform a specific task. The examples must be selected carefully otherwise time will be wasted or even worse the network might function incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

In ANN, an artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the testing mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. Backpropagation algorithm is used to train the network in most of the neural network architecture.

In the testing mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong to the taught list of input patterns, the firing rule is used to determine whether to fire or not. ANN is very sophisticated modeling techniques capable of modeling extremely complex functions, because artificial neural networks are nonlinear.

This paper describes a set of experiments with backpropagation algorithm performed on data from weather forecasting. Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere for a given place and using scientific understanding of atmospheric processes to project how the atmosphere will evolve in that place. Modern weather forecasting involves a combination of computer models, observation, and knowledge of trends and patterns [2]. Using these methods, reasonably accurate forecasts can be made up to five days in advance. So in this study, an attempt has been made to analyze the performance of weather forecasting which helps to infer parameters for desired performance measures, using backpropagation algorithm of artificial neural network architecture with random, genetic and orthonormal weights.

The motivations of this paper are to analyze the performance of weather forecasting, to study the optimal number of orthonormalisation in the backpropagation algorithm of ANN architecture by implementing orthonormalisation in weights, and to study the effectiveness of orthonormal matrix weights over weights evolved by genetic algorithm. The paper concludes with a section comparing, the better learning of backpropagation algorithms with orthonormal weights and the future possibilities.

MATERIALS AND METHODS

Artificial Neural Network

Architecture of ANN is a set of very simple processing neurons with a high degree of interconnections between them and is able to learn and adapt to perform a specific task better, by modifying the strength of connections between the neurons. In ANN, units are connected to one another as shown in figure1.

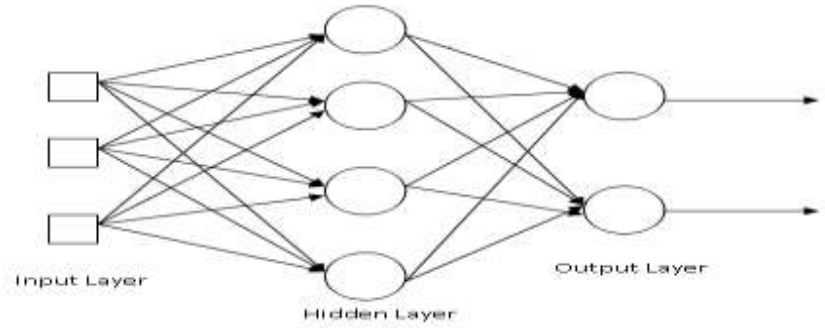


Figure 1. Artificial Neural Network Architecture

There is a real number associated with each connection, which is called the weight of the connection. We denote W_{ij} the weight of the connection from unit u_i to unit u_j as W_{ij} . It is then convenient to represent the pattern of connectivity in the network by a weight matrix W whose elements are the weights W_{ij}. A positive weight represents an excitatory connection whereas a negative weight represents an inhibitory connection. The pattern of connectivity characterizes the architecture of the network. Backpropagation algorithm is used to train ANN architecture. In this study, to train artificial neural network architecture, backpropagation training algorithm [3] with random, genetic and orthonormal weights are used.

Artificial Neural Network Backpropagation Algorithm

In backpropagation algorithm, the weights and threshold values for network are assigned as random, initially. Input values from a training set are presented to the network input layer neurons and the expected output values are declared within the qualified set. The number of hidden layer neurons are give an as input values. Output vectors are calculated using sigmoid function. The error between network input and the target of the learning set [4] is calculated. Then the weights of backpropagation neural network are updated through the process of propagating

backwards the error related to the output neuron's results. The weights of neurons in hidden and output layers are adjusted using calculated error, until the training is completed.
Step 1: The initial weights for network in input and hidden layers are assigned as random values that are uniformly distributed over a small range.
Step 2: Input values from a training set are presented to the network input layer neurons and the expected output values that are declared within the qualified set. The network hidden layer neurons then calculate their output. Calculate output vectors using sigmoid function.
Step 3: This is the step in which the weights of backpropagation neural network are updated through the process of propagating backwards the error related to the output neurons results. Calculate the error between network input and the target of the learning set. Adjust the weights of neurons in hidden and output layer by the value of 'DELTA'
Step 4: Repeat step 2 and 3 for each vector in the learning set until 'DELTA' is acceptably low. When this terminating condition is reached, the training is completed and algorithm terminates.

Artificial Neural Network with Genetic Algorithm

A genetic algorithm (GA) is a search technique used in computing exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms [5] are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover.

A genetic algorithm operates as follows: The population is initialized, using the procedure or randomly. The result of the initialization is a set of chromosomes. Each member of the population is evaluated. Evaluations may be normalized; the important thing is to preserve relative ranking of evaluations. The population undergoes reproduction until a stopping criterion [6] such as, a solution that satisfies minimum criteria or fixed number of generations or computation time reached, etc. is met. Reproduction consists of a number of iterations with 3 steps:
(a) One or more parents with the highest evaluations are chosen to reproduce.
(b) Selection, recombination and mutation are applied to the parents to produce children.
(c) The children are evaluated and inserted into the population.

In genetic algorithm, the entire population is replaced in each cycle of reproduction. When a genetic algorithm is run using a representation, operators can generate better children from good parents. The algorithm can produce populations of better individuals, converging finally on results close to a global optimum. Since genetic algorithms are excellent at searching a state-space, searching for ANN weights is an ideal application. Simply set up the genetic algorithm to evolve a string of floating point numbers within the range that you specify, that can be used as weights for the network. The biggest trouble in using a GA is specifying the range of the weights.

Orthonormal

- An orthogonal set of nonzero vectors $\{u_1, u_2, \dots, u_n\}$ is orthonormal if $|| u_i|| = 1,\ i = 1,2, \dots, n.$ (1)
- If u and v belong to an inner product space V and v is not equal to zero, then the vector projection of u along v is $((u.v)/||v||^2)) * v$ (2)

Gram-Schmidt Theorem **is** a procedure to evaluate orthonormal matrix.

Gram-Schmidt Theorem

Every finite-dimensional inner product space has an orthogonal basis [7, 8].

Proof: Let {u1,u2,…,un} be a basis of the inner product space V. Construct an orthogonal set $\{v_1,v_2,\dots v_n\}$ of vectors in V, which is a basis for V. Write $v_1=u_1$.
To construct v_2, subtract from u_2 its vector projection along v_1.So

$$v_2 = u_2 - \left((u_2.v_1)/||v_1||^2\right) * v_1 \quad (3)$$

This gives $v_2.v_1=0$ and so $\{v_1,v_2\}$ is an orthogonal set.
Now to construct v_3, subtract from u_3 its vector projections along v_1 and v_2. Thus

$$v_3 = u_3 - \left((u_3.v_1)/||v_1||^2\right) * v_1 - \left((u_3.v_2)/||v_2||^2\right) * v_2 \quad (4)$$

This gives $v_3.v_1=0=v_3.v_2$. Hence, $\{v_1,v_2,v_3\}$ is an orthogonal set. Proceeding thus and using all the vectors $u_1,u_2,\dots u_n$, we construct the orthogonal set $B=\{v_1,v_2,\dots,v_n\}$

where, $v_n = u_n - \left(\frac{(u_n.v_1)}{||v_1||^2}\right) * v_1 - \left(\frac{(u_n.v_2)}{||v_2||^2}\right) * v_2 - \cdots$

$$-\left((u_n.v_{n-1})/||v_{n-1}||^2\right) * v_{n-1} \quad (5)$$

The sequence $\{v_1,v_2,\ldots,v_n\}$ is the required system of orthogonal vectors.

$$e_1 = \frac{v_1}{\|v_1\|}, e_2 = \frac{v_2}{\|v_2\|}, \ldots, e_n = \frac{v_n}{\|v_n\|} \quad (6)$$

The normalized vectors $e_1, e_2, e_3, \ldots, e_n$ form an orthonormal set.

Orthonormal-Backpropagation Algorithm

Back propagation is a systematic method for training multilayer artificial neural networks. It is a multi-layer forward network using extended gradient-descent based delta-learning rule. Back propagation provides a computationally efficient method for changing the weights in a feed forward network, with differentiable activation function units and to learn a training set of input–output examples. This training algorithm involves four stages,

1. Initialization of weights 2. Feed forward 3. Back propagation of errors 4. Updating of weights and biases.

During the first stage, which is the initialization of weights, some small random values are assigned [9]. It might influence the net to reach global minima of error and if so, it determines how rapidly it converges. If the initial weight is too large, the initial input signals to each hidden or output unit will fall in the saturation region where the derivative of the sigmoid has a very small value. If initial weights are too small, the net input to hidden or output unit will approach zero, which then causes extremely slow learning. To get the best result the initial weights are set to random and then converted into orthonormal values using Gram-Schmidt orthonormalisation process in this orthonormal-backpropagation algorithm of artificial neural network.

Orthonormal-backpropagation Algorithm

Step 1: Input values of the training set$(p_i, d_i)$are presented to the input layer neurons, where, p_i =input values, d_i =desired output values for i^{th} pattern.

Step 2: Set the initial weights for neural network in input and hidden layers as random values that are uniformly distributed over a small range, learning rate $\eta > 0$ and an acceptable error value Emax.

Step 3: Convert these random weights into orthonormal weights using Gram-Schmidt orthonormalisation process by

$$e_n = v_n/\|v_n\| \quad (7)$$

where, $$v_n = u_n - \left(\frac{(u_n.v_1)}{\|v_1\|^2}\right) * v_1 - \left(\frac{(u_n.v_2)}{\|v_2\|^2}\right) * v_2 - \ \ldots \ - \left(\frac{(u_n.v_{n-1})}{\|v_{n-1}\|^2}\right) * v_{n-1} \quad (8)$$

Step 4: Calculate the activation of i^{th} neuron in the input layer at t^{th} iteration by

$$X_i = p_i(t) \quad (9)$$

Step 5: Calculate the activation of jth neuron in the hidden layer by

$$X_j^h = \sum_i X_i W_{ij}^h \quad (10)$$

where, h - hidden layer,W_{ij}^h - weight between the ith and j^{th} neuron.

Step 6: Calculate the output of j^{th} neuron in the hidden layer by

$$Y_j^h = f_j^h(X_j^h) = \frac{1}{1+e^{-X_j^h}} \quad (11)$$

Step 7: Calculate the activation of k^{th} neuron in the output layer by

$$X_k^o = \sum_j Y_j^h W_{jk}^o \quad (12)$$

where, o - output layer,W_{jk}^o - weight between the j^{th} and k^{th} neuron.

Step 8: Calculate the output of k^{th} neuron in the output layer by

$$Y_k^o = f_k^o(X_k^o) = \frac{1}{1+e^{-X_k^o}} \quad (13)$$

Step 9: The network calculates the error, after the actual output y_j of all output neurons has been found out. Steps in error calculation:

- Start at the output layer neurons and work backward to the hidden layers.
- Adjust the weight for the k^{th} neuron in output layer by

$$W_{jk}^o(t+1) = W_{jk}^o(t) + \Delta W_{jk}^o \quad (14)$$

$$\Delta W_{jk}^o = \eta \delta_k^o Y_j^h \quad (15)$$

$$\delta_k^o = Y_k^o(1 - Y_k^o)(d_k - Y_k^o) \quad (16)$$

- Adjust the weight for the j^{th} neuron in hidden layer by

$$W_{ij}^h(t+1) = W_{ij}^h(t) + \Delta W_{ij}^h \quad (17)$$

$$\Delta W_{ij}^h = \eta \delta_j^h p_i \quad (18)$$

$$\delta_j^h = Y_j^h\left(1 - Y_j^h\right)\sum_k \delta_k^o W_{jk}^o \quad (19)$$

- Calculate the error for the i^{th} pattern by
  $E_i = \frac{1}{2}\sum_k (d_{ik} - Y_k^o)^2$ (20)
- Calculate total error for all patterns by
  $E = \sum_i E_i$ (21)

Step 10: If E $>$ $Emax$ then i=1. Repeat from step 4 to 9 for each training vector in the training set to initiate the new training cycle, until the error for the entire set is minimized to permissible limit Emax.
In orthonormal-backpropagation algorithm initial weights for input and hidden layer are taken as random at step 2 and converted into orthonormal weight [10] using Gram-Schmidt process at step3. These converted values are taken as initial weights in input layer and hidden layer. Here conversion of random weight into orthonormal weight is performed only once in step 3 alone. After that, the input and hidden layers weights are calculated using feed forward technique of backpropagation algorithm. In this study Implementation of optimal number of orthonormalisation is also evaluated.

RESULTS AND DISCUSSION

Artificial Neural Network

This work uses two sets of data approach. The first of these sets is the training set, which is used for the actual training of network and for determination of the network's recall ability. The second data set is the testing data set, which is not used in the training process and is used to test the network level of generalization [11]. The different types of data are in bit string representation and are used in the training and testing of the networks. The bit strings represent the expected activation of the output neurons with regard to a pattern.
The performance analysis of weather forecasting is done using backpropagation algorithm with random, genetic and orthonormal weights in ANN. Initializing the weight matrix in neural network algorithms plays important role in algorithm convergence (cycles). In the backpropagation training algorithm of ANN, the weight matrix is initialized using random values. In genetic algorithm, the weights are calculated using genetic technique. In this study orthonormal weights are used by converting random weights into orthonormal weight (orthonormalisation) in backpropagation algorithm of ANN, also evaluated optimal number of orthonormalisation.

In the present study, weather forecasting is carried out based on data consolidated from meteorological experts and documents. To study the effective factors of weather forecasting, the basic input data for classification needs preprocessing. This procedure helps in the incorporation of factors: temperature, air pressure, humidity, cloudiness, precipitation, wind direction, wind speed and so on,in weather forecasting for neural network based classification.
The inputs (factors) to the artificial neural network are in the form of 1 or 0 based on the presence or absence of the factors. In this study, the different types of weather forecasting are heavy rain, moderate rain and no rain. The output of the neural networks is in the form of 1 or 0, based on presence or absence of the types of weather forecasting.

In this research work, the effective factors of weather forecasting are studied using backpropagation algorithm of artificial architectures with random, genetic and orthonormal weights. The neural networks are trained with twenty-seven inputs (factors), and three outputs with samples of patterns collected from meteorological department. The number of inputs are limited and outputs are selected based on the advice of meteorological experts and documents.

In weather forecasting, which uses backpropagation algorithm with random weight, the training is terminated after 8236 cycles when an error goal of .01 is achieved. During testing, the neural network's performance comes close to 81% accuracy on every trial. When we use genetic algorithm to calculate weight, training is terminated after 5367 cycles when an error goal of .01 is achieved with 84% accuracy. After converting random weight into orthonormal weight the training is terminated after 3978 cycles when an error goal of .01 is achieved with 84% accuracy. The experimental results are shown in the table 1. When orthonormal weight is used, the convergence cycle is minimized in the experimental results of weather forecasting. These results also prove that orthonormal-backpropagation algorithm performs better than traditional random and genetic weights. The better performance of orthonormal-backpropagation algorithm in developing ANN architecture for weather forecasting is represented in figure 2 and figure 3. These experimental results also prove that execution time is minimized, because the number of cycles in training is reduced and orthonormalisation is done only once initially.

ISSN: 2277-9655
Impact Factor: 4.116

Table 1. Classification of Weather Forecasting in ANN

Weather Forecasting *(Rain Fall Patterns)*	No. of pattern	Percentage of correct classification				
		Expert I	Expert II	ANN Backpropagation Algorithm		
				Random Weights	*Genetic Weights*	***Orthonormal Weights***
No rain	12	92	75	92	92	**92**
Moderate Rain	12	83	58	58	67	**67**
Heavy rain	12	75	92	92	92	**92**
Over all accuracy (%)		83%	75%	81%	84%	**84**%
Convergence (cycles)				8236	5367	**3978**
System Time (seconds)				180.71	105.04	**67.868**

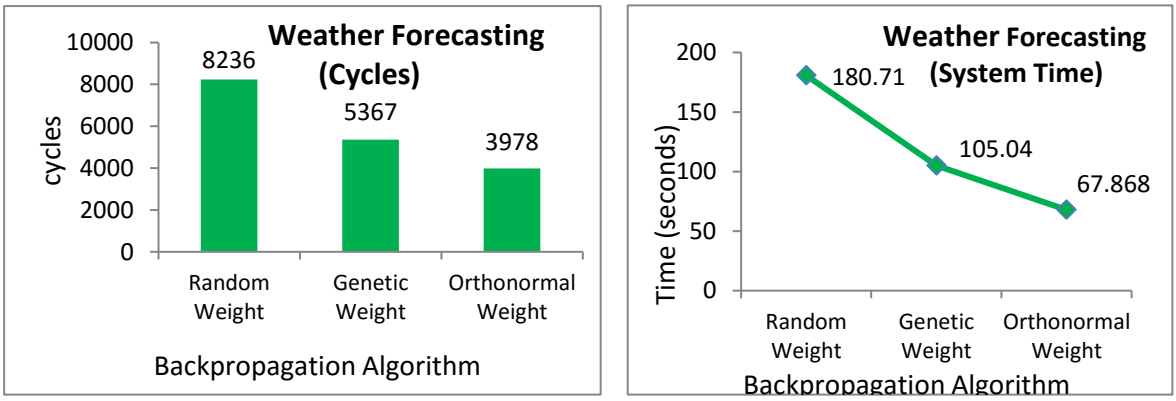


Figure 2. Better Speed (Cycles and System Time) of Orthonormal- backpropagation in ANN

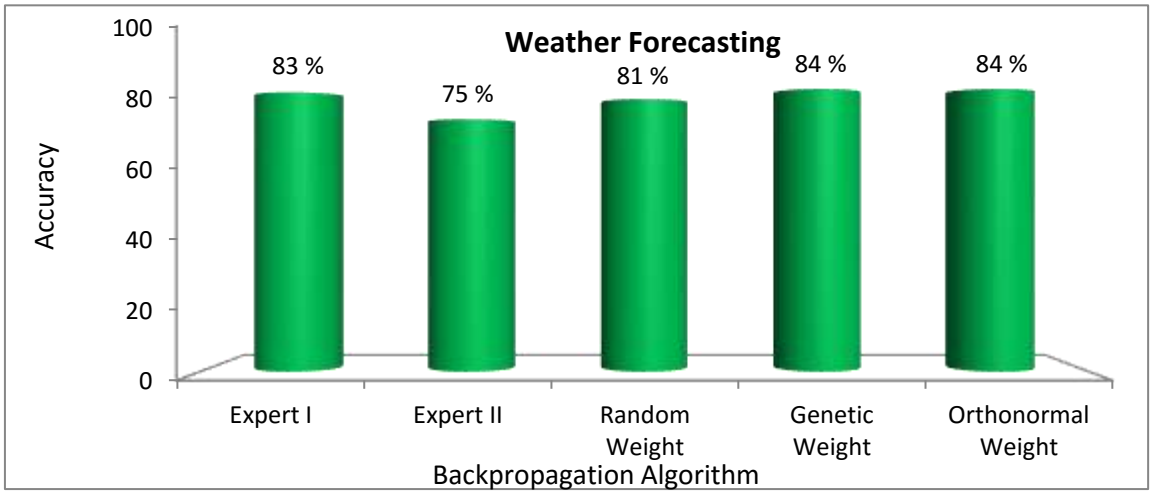


Figure 3. Performance (Accuracy) of Weather Forecasting in ANN

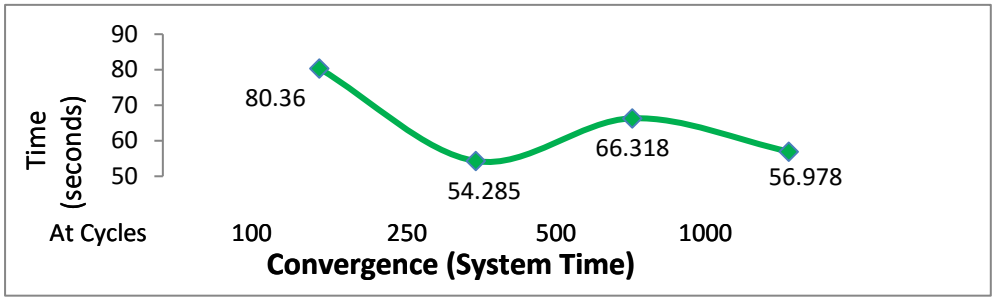
ISSN: 2277-9655
Impact Factor: 4.116

Identifying Optimal Number of Orthonormalisation using Weather Forecasting

Optimal cycle for weight regulation using repeated orthonormalisation process in orthonormal-backpropagation algorithm of ANN is experimented in weather forecasting application. Optimal cycle for repeated orthonormalisation is evaluated by converting random weights into orthonormal weights using orthonormalisation process at 100thor 250th or500th or 1000th cycle in training of neural network. When orthonormalisation is repeated, at 100th cycle training is terminated after 4090 cycles, at 250thcycle training is terminated after 3106 cycles, at 500thcycle training is terminated after 3935 cycles, at 1000th cycle training is terminated after 3245 cycles when an error goal of .01 is achieved with 95% accuracy. In this experiment also, during second time of orthonormalisation, improved convergence occursat250thcycle. So, optimal cycle for repeating orthonormalisation, the second time, according to the experimental results of weather forecasting is at250thcycle. The performance of repeated orthonormalisationin weather forecasting shows that repeated orthonormalisation, during the second time is better at 250thcycle when compared to performing orthonormalisation only once and represented in table 2 and figure 4.

Table 2. Repeated Orthonormalisation Process in Weather Forecasting

Weather Forecasting (Rain Fall)	**RepeatedOrthonormalisation**				
	Once	**Repeated** (100^{th} / 250^{th} / 500^{th} / 1000^{th}) cycle			
At Cycles	**0**	**100**	**250**	**500**	**1000**
Convergence (Cycles)	3978	4090	**3106**	3935	3245
Overall Accuracy (%)	84%	84%	**84**%	84%	84%
System Time(seconds)	67.868	80.360	**54.285**	66.318	56.978



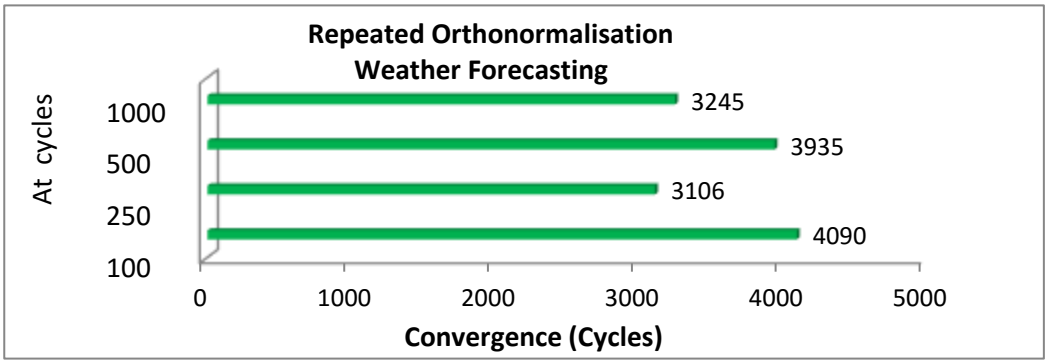


Figure 4. Performance of Repeated Orthonormalisation Process of Orthonormal-backpropagation Algorithm in ANNin terms of Convergence (System Time and Cycles)

CONCLUSION AND SUGGESTION

The experiments in this study illustrate the improvements gained by newly proposed orthonormal weights in neural network algorithms. The orthonormal weights performed better than random and genetic weights in artificial neural network algorithms. It was tested through the problem, performance evaluation of weather forecasting. At the second time of conversion from random weight to orthonormal weight(orthonormalisation), network quickly converges in weather forecasting. The orthonormal weight neural network algorithms have been used to analyze the performance of weather forecasting and the parameters for the desired weather forecasting performance have been inferred. This work presents a solution obtained with the objective of better learning of backpropagation algorithm of artificial neural network using Orthonormal weights and optimal number of orthonormalisation. In future, orthonormal weight implementation can be experimented in various other neural network architectures.

REFERENCES

1. Jensen CA, Reed RD, Marks RJ, El-Sharkawi MA (1999) Inversion of feedforward neural networks: algorithms and applications, Proceedings of the IEEE, Volume 87, Number9, pp. 1536 – 1549.
2. Dabberdt W.,Weather for Outdoorsmen: A complete guide to understanding and predicting weather in mountains and valleys, on the water, and in the woods,Scribner, New York, 1981.
3. Limin Fu (2003) Neural Networks in Computer Intelligence, Tata McGraw hill.
4. Schmidt WF (1993) Initializations, Back Propagation and Generalization of Feed-Forward Classifiers, IEEE International Conference on Neural Networks, pp. 598-604.
5. Yu-Tzu Chang, Jinn Lin, Jiann-Shing Shieh and Maysam F. Abbod (2012) Optimization the Initial Weights of Artificial Neural Networks via Genetic Algorithm, Advances in Fuzzy Systems, pp.1-9.
6. Eiben AE, et al (1994) Genetic algorithms with multi-parent recombination, PPSN III: Proceedings of the International Conference on Evolutionary Computation, The Third Conference on Parallel Problem Solving from Nature: 78-87. ISBN 3-540-58484-6.
7. Krishnamurthy V, et al (1976) An Introduction to LINREAR ALGEBRA, East-West Press Private Limited, New Delhi.
8. Arumugam S, Isaac AT (2006) Modern Algebra, Scitech publication, Chennai.
9. Nguyen D, Widrow B (1990) Improving the Learning Speed of 2-layer Neural Networks by Choosing Initial Values of the Adaptive Weights, Proceedings of the IEEE International Joint Conference on Neural Networks, Volume 3, pp. 21-26.
10. Philip D Wasserman (1989) Neural Computing Theory and Practice, Van nostrand reinhold, New York.
11. Mulgrew B (1994) Orthonormal functions for nonlinear signal processing and adaptive filtering, IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. III/509 - III/512 vol.3.