DETERMINATION OF OVER-LEARNING AND OVER-FITTING PROBLEM IN BACK PROPAGATION NEURAL NETWORK

Gaurang Panchal[1], Amit Ganatra[2], Parth Shah[3], Devyani Panchal[4]

Department of Computer Engineering, Charotar Institute of Technology (Faculty of Technology and Engineering), Charotar University of Science and Technology, Changa, Anand-388 421, INDIA

[1]gaurangpanchal.ce@ecchanga.ac.in
[2]amitganatra.ce@ecchanga.ac.in
[3]parthshah.ce@ecchanga.ac.in
[4]devyanipanchal.it@ecchanga.ac.in

ABSTRACT

A drawback of the error-back propagation algorithm for a multilayer feed forward neural network is over learning or over fitting. We have discussed this problem, and obtained necessary and sufficient Experiment and conditions for over-learning problem to arise. Using those conditions and the concept of a reproducing, this paper proposes methods for choosing training set which is used to prevent over-learning. For a classifier, besides classification capability, its size is another fundamental aspect. In pursuit of high performance, many classifiers do not take into consideration their sizes and contain numerous both essential and insignificant rules. This, however, may bring adverse situation to classifier, for its efficiency will been put down greatly by redundant rules. Hence, it is necessary to eliminate those unwanted rules. We have discussed various experiments with and without over learning or over fitting problem.

KEYWORDS

Neural Network, learning, Hidden Neurons, Hidden Layers

1. INTRODUCTION TO NEURAL NETWORK

Artificial Neural Network (ANN) is a computational model, which is based on Biological Neural Network. Artificial Neural Network is often called as Neural Network (NN). To build artificial neural network, artificial neurons, also called as nodes, are interconnected. The architecture of NN is very important for performing a particular computation. Some neurons are arranged to take inputs from outside environment. These neurons are not connected with each other, so the arrangement of these neurons is in a layer, called as Input layer. All the neurons of input layer are producing some output, which is the input to next layer. The architecture of NN can be of single layer or multilayer. In a single layer Neural Network, only one input layer and one output layer is there, while in multilayer neural network, there can be one or more hidden layer.

An artificial neuron is an abstraction of biological neurons and the basic unit in an ANN. The Artificial Neuron receives one or more inputs and sums them to produce an output. Usually the sums of each node are weighted, and the sum is passed through a function known as an activation

DOI : 10.5121/ijsc.2011.2204

or transfer function. The objective here is to develop a data classification algorithm that will be used as a general-purpose classifier. To classify any database first, it is required to train the model. The proposed training algorithm used here is a Hybrid BP-GA. After successful training user can give unlabeled data to classify

The synapses or connecting links: that provide weights, wj, to the input values, xj for j = 1 ...m;
An adder: that sums the weighted input values to compute the input to the activation function

$$v = w_0 + \sum_{j=1}^{m} w_j x_j$$

Where,
w0 is called the bias, is a numerical value associated with the neuron. It is convenient to think of the bias as he weight for an input x0 whose value is always equal to one, so that;

$$v = \sum_{j=0}^{m} w_j x_j$$

An activation function g: that maps *v* to *g*(*v*) the output value of the neuron. This function is a monotone function. The logistic (also called the sigmoid) function $g(v) = (e^{v}/(1+e^{v}))$ as the activation function works best. The practical value of the logistic function arises from the fact that it is almost linear in the range where g is between 0.1 and 0.9 but has a squashing effect on very small or very large values.

2. INTRODUCTION TO OVER LEARNING

Over learning (in Neural Networks): When an iterative training algorithm is run, over fitting which occurs when the algorithm is run for too long (and the network is too complex for the problem or the available quantity of data).

Over parameterized Model: An over parameterized model uses the indicator variable approach to represent effects for categorical predictor variables in general linear models and generalized linear models. To illustrate indicator variable coding, suppose that a categorical predictor variable called Gender has two levels (i.e., Male and Female). A separate continuous predictor variable would be coded for each group identified by the categorical predictor variable. Females might be assigned a value of 1 and males a value of 0 on a first predictor variable identifying membership in the female Gender group, and males would then be assigned a value of 1 and females a value of 0 on a second predictor variable identifying membership in the male Gender group.

Over-Learning and Generalization

One major problem with the approach outlined above is that it doesn't actually minimize the error that we are really interested in - which is the expected error the network will make when new cases are submitted to it. In other words, the most desirable property of a network is its ability to

generalize to new cases. In reality, the network is trained to minimize the error on the training set, and short of having a perfect and infinitely large training set, this is not the same thing as minimizing the error on the real error surface - the error surface of the underlying and unknown model (see Bishop, 1995).

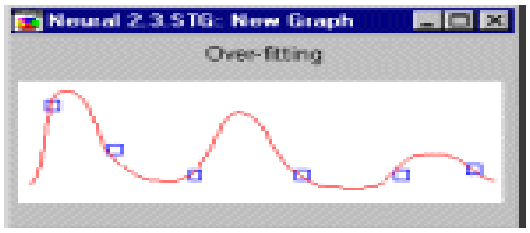
The most important manifestation of this distinction is the problem of over-learning, or over-fitting. It is easiest to demonstrate this concept using polynomial curve fitting rather than neural networks, but the concept is precisely the same.

A polynomial is an equation with terms containing only constants and powers of the variables.

For example:

$$y=2x+3$$
$$y=3x2+4x+1$$

Different polynomials have different shapes, with larger powers (and therefore larger numbers of terms) having steadily more eccentric shapes. Given a set of data, we may want to fit a polynomial curve (i.e., a model) to explain the data. The data is probably noisy, so we don't necessarily expect the best model to pass exactly through all the points. A low-order polynomial may not be sufficiently flexible to fit close to the points, whereas a high-order polynomial is actually too flexible, fitting the data exactly by adopting a highly eccentric shape that is actually unrelated to the underlying function. See illustration below.



Neural networks have precisely the same problem. A network with more weights models a more complex function, and is therefore prone to over-fitting. A network with less weight may not be sufficiently powerful to model the underlying function. For example, a network with no hidden layers actually models a simple linear function.

How then can we select the right complexity of network? A larger network will almost invariably achieve a lower error eventually, but this may indicate over-fitting rather than good modelling.

The answer is to check progress against an independent data set, the selection set. Some of the cases are reserved, and not actually used for training in the back propagation algorithm. Instead, they are used to keep an independent check on the progress of the algorithm. It is invariably the case that the initial performance of the network on training and selection sets is the same (if it is not at least approximately the same, the division of cases between the two sets is probably biased). As training progresses, the training error naturally drops, and providing training is minimizing the true error function, the selection error drops too. However, if the selection error stops dropping, or indeed starts to rise, this indicates that the network is starting to overfit the

data, and training should cease. When over-fitting occurs during the training process like this, it is called over-learning. In this case, it is usually advisable to decrease the number of hidden units and/or hidden layers, as the network is over-powerful for the problem at hand. In contrast, if the network is not sufficiently powerful to model the underlying function, over-learning is not likely to occur, and neither training nor selection errors will drop to a satisfactory level.

The problems associated with local minima, and decisions over the size of network to use, imply that using a neural network typically involves experimenting with a large number of different networks, probably training each one a number of times (to avoid being fooled by local minima), and observing individual performances. The key guide to performance here is the selection error. However, following the standard scientific precept that, all else being equal, a simple model is always preferable to a complex model, you can also select a smaller network in preference to a larger one with a negligible improvement in selection error.

A problem with this approach of repeated experimentation is that the selection set plays a key role in selecting the model, which means that it is actually part of the training process. Its reliability as an independent guide to performance of the model is therefore compromised - with sufficient experiments, you may just hit upon a lucky network that happens to perform well on the selection set. To add confidence in the performance of the final model, it is therefore normal practice (at least where the volume of training data allows it) to reserve a third set of cases - the test set. The final model is tested with the test set data, to ensure that the results on the selection and training set are real, and not artifacts of the training process. Of course, to fulfill this role properly the test set should be used only once - if it is in turn used to adjust and reiterate the training process, it effectively becomes selection data!

This division into multiple subsets is very unfortunate, given that we usually have less data than we would ideally desire even for a single subset. We can get around this problem by resampling. Experiments can be conducted using different divisions of the available data into training, selection, and test sets. There are a number of approaches to this subset, including random (monte-carlo) resampling, cross-validation, and bootstrap. If we make design decisions, such as the best configuration of neural network to use, based upon a number of experiments with different subset examples, the results will be much more reliable. We can then either use those experiments solely to guide the decision as to which network types to use, and train such networks from scratch with new samples (this removes any sampling bias); or, we can retain the best networks found during the sampling process, but average their results in an ensemble, which at least mitigates the sampling bias.

To summarize, network design (once the input variables have been selected) follows a number of stages:

- Select an initial configuration (typically, one hidden layer with the number of hidden units set to half the sum of the number of input and output units).
- Iteratively conduct a number of experiments with each configuration, retaining the best network (in terms of selection error) found. A number of experiments are required with each configuration to avoid being fooled if training locates a local minimum, and it is also best to resample.

- On each experiment, if under-learning occurs (the network doesn't achieve an acceptable performance level) try adding more neurons to the hidden layer(s). If this doesn't help, try adding an extra hidden layer.
- If over-learning occurs (selection error starts to rise) try removing hidden units (and possibly layers).
- Once you have experimentally determined an effective configuration for your networks, resample and generate new networks with that configuration.

3. EXPERIMENTS AND RESULTS

1. **Experiment No. 1** : No of Input Neurons: 2, No of Hidden Neurons: 7, No of Output Neurons: 1

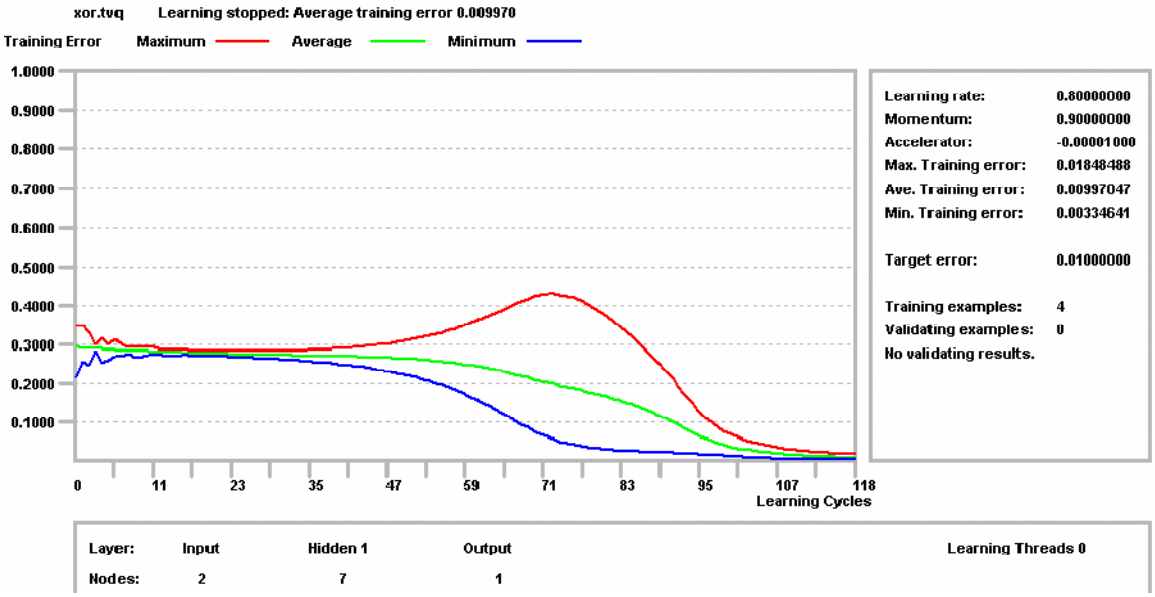


Fig. 1 Training with Number of Hidden Neurons: 7

The above mention figure shows the Learning cycle of XOR Problem with 118 epochs. Initially we have tested the problem with the l-m-n configuration of Neural network is [2-7-1], which is some what higher number of hidden neurons. After the training session we get the training error is 0.01848 and minimum is 0.003346.As soon as training goes on we get higher number of error. The same experiment has been done with the KDD CUP'99 data for Training purpose with more than 600 records and more than 17 columns which is discussed in later section.

2. **Experiment No. 2** : No of Input Neurons: 2, No of Hidden Neurons: 5, No of Output Neurons: 1

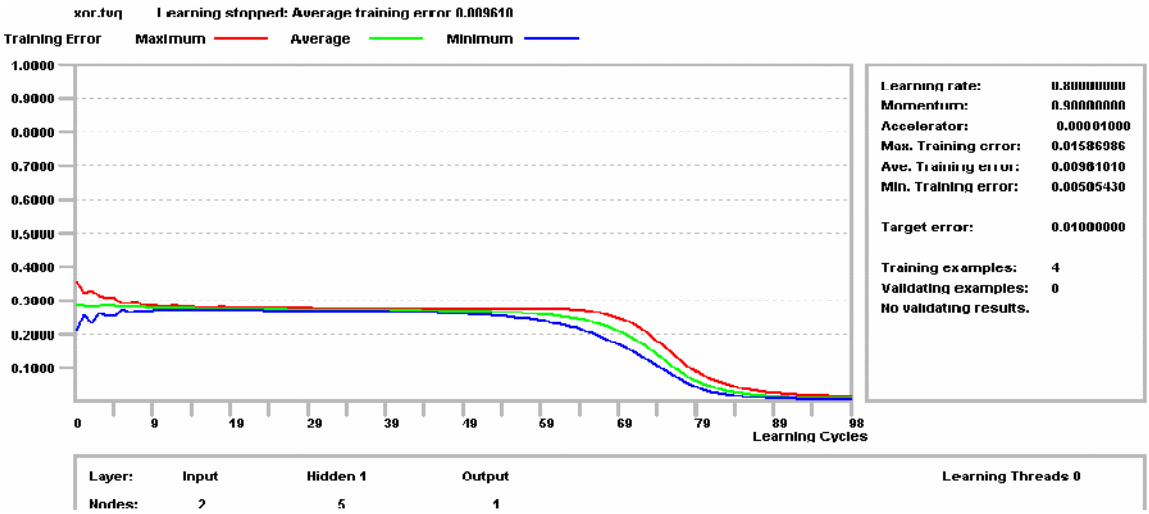


Fig. 2 Training with Number of Hidden Neurons: 5

Figure 2 shows the learning process of same problem with 5 hidden neurons. As a comparison with previous experiment , it is quit better. We have reduced the number of errors.

3. **Experiment No. 3** : No of Input Neurons: 2 ,No of Hidden Neurons: 4, No of Output Neurons: 1

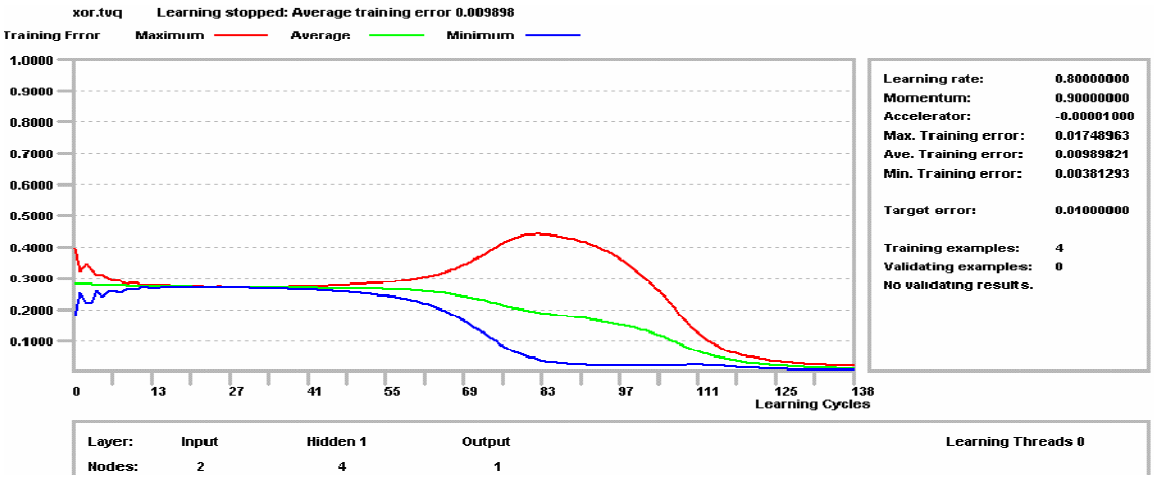


Fig 3.Training with Number of Hidden Neurons: 4

The Figure 3 shows the learning process of same problem with 4 hidden units also getting better result in terms of error.

4. **Experiment No. 4 :** No of Input Neurons: 2, No of Hidden Neurons: 3, No of Output Neurons: 1

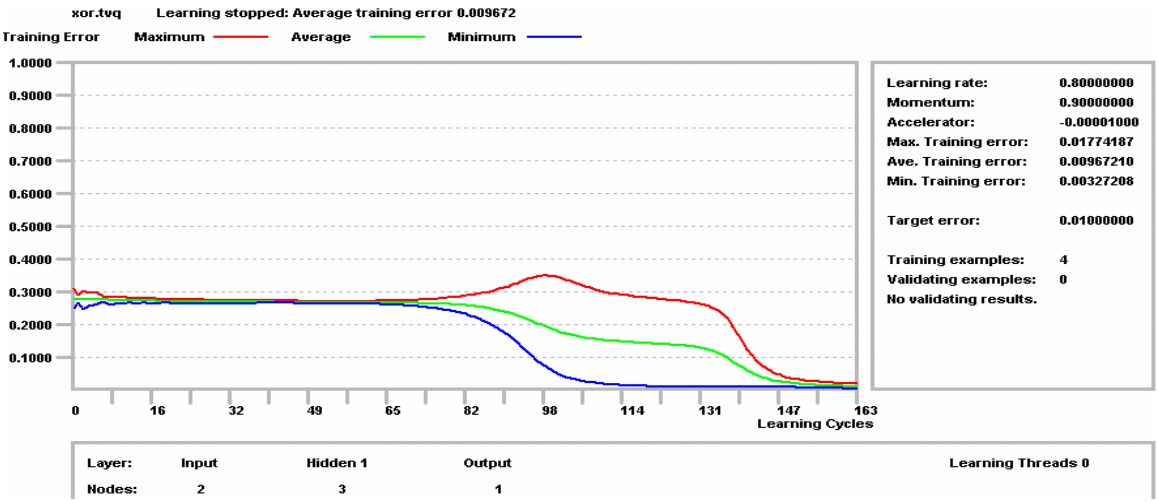


Fig 4.Training with Number of Hidden Neurons: 3 .

Figure 4 shows the learning progress with 3 hidden neurons and we are getting maximum error 0.01774 and minimum error is 0.00327 which is good as compare to previous experiment. As soon as we increase number of hidden neurons the over fitting problem get started. So let's see the result by reducing one more hidden neuron.

5. **Experiment No. 5** : No of Input Neurons: 2, No of Hidden Neurons: 3, No of Output Neurons: 1

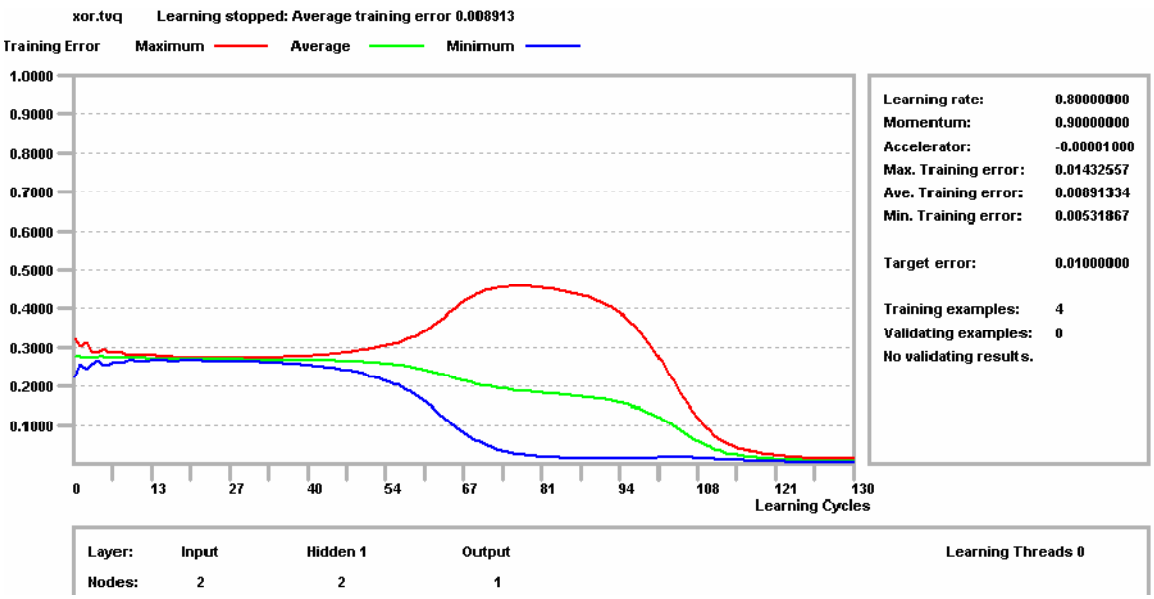


Fig 5.Training with Number of Hidden Neurons: 2

The above figure show the training session with number of hidden neurons is 2. We have reduced the Maximum error but also increased minimum error but as a average error rate this is better.

TABLE I
TRAINING WITH NUMBER OF HIDDEN NEURONS: 2

Changes in Number of Hidden Neurons			
No of Hidden Nodes	**Max Training**	**Avg**	**Min**
7	0.0184	0.0099	0.0033
5	0.0158	0.0096	0.00505
4	0.0174	0.0098	0.0038
3	0.0177	0.0096	0.0032
2	0.0143	0.0089	0.00531

The Above table show the summary of five experiments which is discussed above. The over learning start with some of weakness of Architecture of Network so to solve over learning / over fitting problem, specially in neural network we have to modify the hidden layer structure.
Here we have taken Intrusion Detection Data Set (KDD CUP 99) for training purpose. We took more than 600 rows for the training purpose. It contains 16 data column.

Data Set Details:

No of Rows: **600**.
No of Column: **16**
Class Label Attribute: **Attack_Type**

Other Attribute Name:

Duration,Protocol(udp),Protocol(icmp),Protocol(tcp),Service(IRC),Service(X11),Service(private),Service(telnet),Service(auth),Service(SMTP),Service(eco_i),Service(http),src_bytes(S1),src_bytes(S3),src_bytes(RSTR),src_bytes(SF),dst_bytes,Flags,Lands,wrong_fragment,urgent,HOT,num_failed_logins,logged_in,
num_access_files,num_outbound_cmds,is_hot_login,is_guest_login,Curr_Conn,serror_rate,rerror_rate,same_srv_rate,diff_srv_rate,srv_count,srv_serror_rate,srv_rerror_rate,srv_diff_host_rate

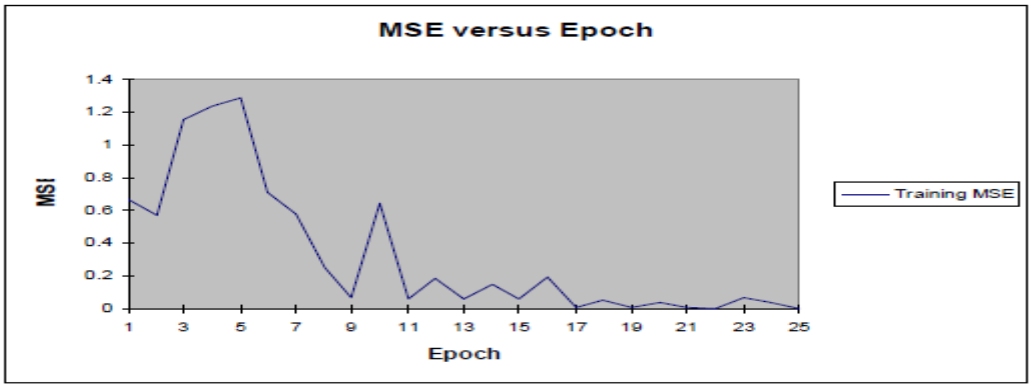


Fig. 6 MSR vs. Epoch for Hidden Neurons 10

The above figure shows the progress report of MSE with various numbers of epochs with hidden neurons 10. Below table show the experiment based on above figure.

TABLE II
TRAINING WITH NUMBER OF HIDDEN NEURONS: 2

Best Network	Training
Epoch #	25
Minimum MSE	0.000514907
Final MSE	0.000514907

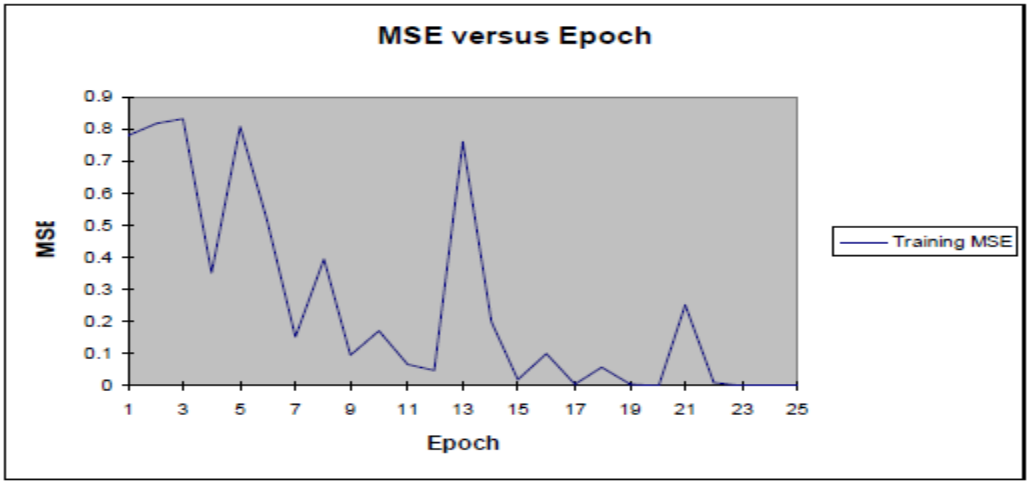


Fig. 7 MSR vs. Epoch for Hidden Neurons 10

The above figure shows the progress report of MSE with various numbers of epochs with hidden neurons 10. Below table show the experiment based on above figure.

TABLE III
TRAINING WITH NUMBER OF HIDDEN NEURONS: 2

Best Network	Training
Epoch #	25
Minimum MSE	0.001563647
Final MSE	0.001563647

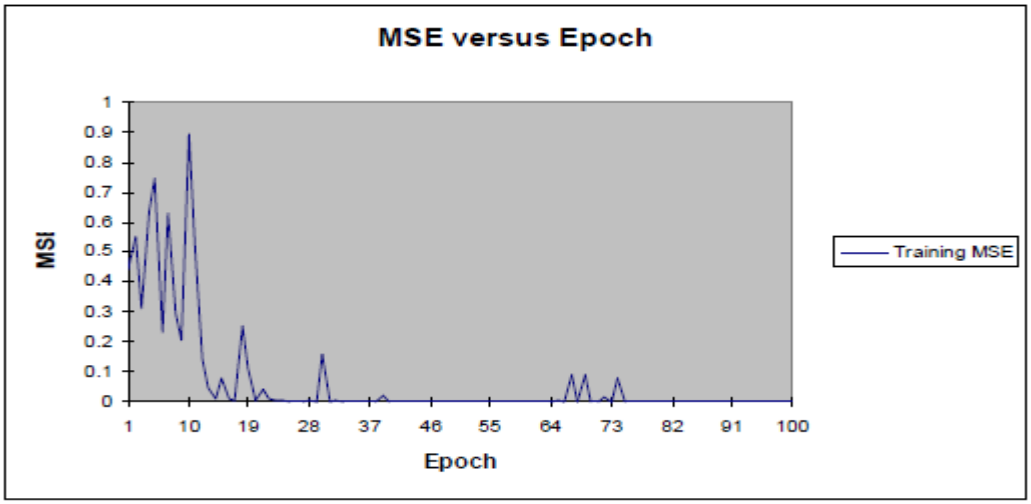


Fig. 8 MSR vs. Epoch for Hidden Neurons 7

The above figure shows the progress report of MSE with various numbers of epochs with hidden neurons 10. Below table show the experiment based on above figure.

TABLE IV
TRAINING WITH NUMBER OF HIDDEN NEURONS: 2

Best Network	Training
Epoch #	100
Minimum MSE	0.000167539
Final MSE	0.000167539

Over learning and over fitting that depend upon the neural network architecture. If data is noisy that time neural network will not train properly. So that we have to go for Data Pre-processing. And if data is not getting train or network enter in to over learning state that time we have to reduce one hidden neurons of one hidden layer. If we don't get solution for over learning that time we have to reduced one hidden layer from the neural network architecture.

4. CONCLUSION

To overcome problem like over Learning or over fitting we have to modify the (l-m-n) configuration of Neural Network. For any classifier it is very important that it learn properly. Because of weak architecture or configuration of neural network, classifier may not learn because of over learning. So over learning or over fitting problem get started. By following the above steps we can resolve the over learning or over fitting problem.

ACKNOWLEDGEMENT

The authors' wishes to thank all the colleagues for their guidance, encouragement and support in undertaking the research work. Special thanks to the Principal for their moral support and continuous encouragement

REFERENCES

[1] Carlos Gershenson , "Artificial Neural Networks for Beginners"

[2] Vincent Cheung ,Kevin Cannons, "An Introduction to Neural Networks", Signal & Data Compression Laboratory, Electrical & Computer Engineering University of Manitoba, Winnipeg, Manitoba, Canada

[3] "Artificial Neural Networks" ocw.mit.edu

[4] Guoqiang Peter Zhang , "Neural Networks for Classification: A Survey", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, VOL. 30, NO. 4, NOVEMBER 2000

[5] V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, "Learning rate adaptation in stochastic gradient descent", ,Department of Mathematics, University of Patras,

[6] Wen Jin-Wei Zhao, Jia-Li Luo Si-Wei and Han Zhen " The Improvements of BP Neural Network Learning Algorithm", Department of Computer Science & Technology,Northem Jiaotong University ,BeiJing, 100044, P.R.China,

[7] Wenjian Wang, Weizhen Lu, Andrew Y T Leung, Siu-Ming Lo, Zongben Xu, "Optimal feed-forward neural networks based on the combination of constructing and pruning by genetic algorithms", IEEE TRANSACTIONS ON NEURAL NETWORKS 2002

[8] "A Detailed Comparison of Backpropagation Neural Network and Maximum-Likelihood Classifiers for Urban Land Use Classification",IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, VOL. 33, NO. 4, JULY 199.5

[9] Z. J. Liu C. Y. Wang Z. Niu A. X. Liu "Evolving Multi-spectral Neural Network Classifier Using a Genetic Algorithm". Laboratory of Remote Sensing Information Sciences, the Institute of Remote Sensing Applications,

[10]Fiszelew, A., Britos, P., Ochoa, A., Merlino, H., Fernández, E., García-Martínez "Finding Optimal Neural Network Architecture Using Genetic Algorithms", R.Software & Knowledge Engineering Center. Buenos Aires Institute of Technology.Intelligent Systems Laboratory. School of Engineering. University of Buenos Aires.

[11]M.P.Craven, "A FASTER LEARNING NEURAL NETWORK CLASSIFIER USING SELECTIVE BACKPROPAGATION" Proceedings of the Fourth IEEE International Conference on Electronics, Circuits and Systems

[12]Wenjian Wang, Weizhen Lu, Andrew Y T Leung, Siu-Ming Lo, Zongben Xu, "Optimal feed-forward neural networks based on the combination of constructing and pruning by genetic algorithms", IEEE TRANSACTIONS ON NEURAL NETWORKS 2002

[13]Teresa B. Ludermir, Akio Yamazaki, and Cleber Zanchettin, "An Optimization Methodology for Neural Network Weights and Architectures" IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 17, NO. 6, NOVEMBER 2006

[14]S. Rajasekaran, G.A Vijayalakshmi Pai, "Neural Networks, Fuzzy Logic, and Genetic Algorithms Synthesis and Applications"

[15]Mrutyunjaya Panda and Manas Ranjan Patra, "NETWORK INTRUSION DETECTION USING NAÏVE BAYES" IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.12, December 2007

[16]S. SELVAKANI1 and R.S.RAJESH2, "Escalate Intrusion Detection using GA – NN", Int. J. Open Problems Compt. Math., Vol. 2, No. 2, June 2009

[17]Nathalie Villa*(1,2) and Fabrice Rossi(3), Recent advances in the use of SVM for functional data classification, First International Workshop on Functional and Operatorial Statistics. Toulouse, June 19-21, 2008

[18] KDD Cup'99 Data set , http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Authors Profile

Prof. Gaurang Panchal has received his B.E. degree in Information Technology from Saurashtra University, Gujarat, India in 2003. He has completed his M.E. in Computer Engineering from Dharamsinh Desai University, Gujarat, India. He has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, since 2007, where he is currently working as an Assistant Professor in the Department of Computer Engineering. His current research area interest includes Data Mining.

Prof. Amit P. Ganatra has received his B.E degree in Computer Engineering from Gujarat University, Gujarat, India in 2000 and master Degree from Dharmsinh Desai University, Gujarat, India in 2004. He has joined his Ph.D in the area of Multiple Classifier System (Information Fusion) at Kadi Sarvavishvidhalaya University, Gandhinagar, India in August 2008. Since 2000 he has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, Where he is currently working as an Associate Professor in the Department of Computer Engineering. He has published more than 50 research papers in the field of data mining and Artificial Intelligence. His current research interest includes Multiple Classifier System, Sequence Pattern Mining

Prof. Parth Shah has received his B.E. degree in Information Technology from Saurashtra University, Gujarat, India in 2002. He has completed his M.E. in Computer Engineering from Dharamsinh Desai University, Gujarat, India. He has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, since 2002, where he is currently working as an Head & Assistant Professor in the Department of Information Technology. His current research area interest includes Data Mining.

Prof. Devyani Panchal has received his B.E. degree in Computer Engineering from Sardar Patel University, Gujarat, India in 2005. She has been with faculty of Engineering and Technology, Charotar University of Science and Technology, Changa, Gujarat, since 2007, where she is currently working as an Assistant Professor in the Department of Computer Engineering. Her current research area interest includes Data Mining.