APPLICATIONS OF ARTIFICIAL NEURAL NETWORKS IN MUSHROOM EDIBILITY CLASSIFICATION

GUSTAVO J. MUNOZ, SUNGMOON JUNG

Faculty Advisor: Sungmoon Jung
Department of Civil Engineering

We report the accuracy of a two-layer, back-propagation artificial neural network in identifying edibility of a set of random mushrooms. Mushrooms edibility was synthesized using many different characteristics. Tests were run using different combinations of number of hidden nodes, separation of training, validation, and test data and number of iterations. Qualitative identification of an optimal combination of network parameters will provide a basis toward applications of artificial neural networks in future civil engineering endeavors.

Artificial intelligence has become a very important topic since the mid twentieth century[1]. An artificial neural network is a computational model used to mimic the processes of natural, biological neural networks, such as the respiratory system and learning functions in the brain. The artificial neural network, or more colloquially known as the neural network, has been used in many computational applications due to its pattern recognition capabilities. Many studies, such as bridge degradation patterns (using Self-Organizing Map, a type of unsupervised artificial neural network)[2], crab sex classification[3] , and abnormality identification using CT scans have been

investigated using Artificial Neural Networks[4].

Neural Networks have a property of learning, which allows for many applications. The learning is based on values given for a training set of data. Once the training is finished, the network has "learned" the given data sets, and, therefore, it can be used as a prediction tool. The usage of the network is often called as testing. Depending on the application, testing is conducted with known target values so that an error is calculated, proving the accuracy of the network.

The question of edibility of mushrooms has been a long-lived query. There have not been any particular "rules of thumb" to follow in order to classify whether a mushroom is poisonous or edible. Because such a system (the mushroom) contains so many variables (i.e. shape of bell, color, smell, etc.) a multi-variable computational approach, such as neural networking, may alleviate the problem.

Preliminary Data Processing

The data used for classification purposes contains target values, that is, values to which the network is trying to calculate to. In this classification, twenty-two characteristics for 8,124 separate mushrooms are given in the data set[5]. Each of these mushrooms has already been determined to be poisonous or edible through investigation prior to this one. The data labels are in character form that cannot be read by Matlab and so must be represented in a form that can be processed [See Table 1].

Mushroom characteristics were first in word form then converted to character form (i.e. bell = b), but none of these forms are read by Matlab. These forms of representation

were then converted to numerical form to allow for usage in Matlab [See Table 2]. Edible is represented as -1 while poisonous is represented as +1.

It is important to note that in representing each characteristic by increasing numerical values, a bias is introduced. This means that some numbers that are larger in value than others (7 vs. 1) may create a bias and so skew the final result. This limitation is strongly considered when evaluating the final results and is understood to be the *less-accurate* way of representing non-numerical characters. Correct representation would prove exhaustive in the time frame given. Correct representation requires converting each attribute to multiple bits, so that the bias is not introduced.

Artificial Neural Network

The neural network used in pattern recognition of mushroom edibility is a two-layer feed forward network. That is, twenty-two inputs, a hidden layer with an arbitrary amount of nodes, and an output (see Figure 1).

The initial tiny circles represent the input of 8,124 data points. The two middle circles represent the hidden layer with two nodes and the final circle represents the output; that is, the classification as "edible" or "poisonous." The hidden layer contains neurons (nodes) with hyperbolic activation functions that compute a weighted summation of the inputs to pass through those functions.

The two-layer network was written into Matlab using the Neural Network Toolbox. This enabled us to use a few commands in order to execute the task of training the

network.

Various trials were used in order to identify the optimal usage of network parameters. Table 3 tabulates all the different combinations and parameters tested for mushroom pattern recognition using the two-layer feed-forward network. Data are split in three groups: training data, which the network uses to train; validation data set, which the network uses as a preliminary testing set; and test data, which finds a value closest to the target value based on the training done by the training set. Each partition is further separated by five hidden nodes or twenty hidden nodes. Two groups are observed, one being run at 1,000 iterations (epochs) and one being run at 10,000 iterations (epochs).

After tests are run, a comparison is made in search of the least error with regards to testing vs. validation vs. training. A semi-qualitative analysis is made to find the best option for network parameters

Results

Figure 4 shows the best validation performance is at 7.4448 x 10^-14. This plot shows that in this particular case (1,000 iterations, 5 nodes), a separation of 20% training, 20% validation, and 60% testing gives us minimum error. Only 37 iterations to reach the minimum gradient were needed. The simplicity of the problem may contribute to such low error and such low iteration.

Figure 8 proved the best performance in terms of the 1,000-iteration, 20-node, category. An error of 8.204 x 10^-16 are noticed from this plot. A minimum gradient is also reached at 211 iterations. The minimum gradient is found at a plot different than the previous example – 5 nodes.

With 10,000 iterations, a very high error, relative to the other tests, is seen in Figure 11. An error of 0.024018 vs. 2.5332 x 10^-15 (Figure 10) is significantly different. The minimum error was found using a separation of 20% training, 20% validation, and 60% testing.

Figure 12 shows the minimum gradient of 1.1225 x 10^-27. This corresponds to a separation of data of 60% training, 20% validation and 20% testing. The other two plots illustrate a similar amount of error as other runs.

Conclusion

The two-layer feedforward artificial neural network has proven very useful in classification of edibility of mushrooms. With the training of the data and learning capacities, the network has provided a test, using various network parameters, to provide values very near the target values. The errors were all on the order of at least 10^-11 except for 1 plot (Figure 11).

Limitations due to incorrect representation of data may have caused a skewed error and possible discrepancies in the plots. A more thorough study of this would provide much better results in terms of accuracy and best option. No clear option was made based on the limited amount of tests run.

Tests should each be run several times, due to the randomly generated order of values. With large numbers of trials tested in the future, data would be plotted to show standard deviation. A quantitative study would also have to be done to understand the pattern of error of each individual plot in order to apply it to a more generalized understanding of an optimal combination of parameters.

With the skills garnered through the study of Artificial Neural Networks, future applications can be investigated with regard to structural failure, degradation and surrounding factors. The application of neural networks also seems to promise various fields of research in other subdisciplinary areas of engineering.

Endnotes

1 Bishop, C.M. Neural Networks for Pattern Recognition, Oxford University Press, 1995

2 Jung, S., Sobanjo, J., Munoz, G.J. Visualization and Assessment of the Aging Infrastructure Using Self-Organizing Map

3 Neural Network Toolbox 6.0.4, Crab Classification, The Mathworks Inc., 2010.

4 Sinha, M., Kennedy, C.S, Ramundo, M.L. Artificial neural network predicts CT scan abnormalities in pediatric patients with closed head injury, U.S. National Library of Medicine, 2001.

5 Asuncion, A., Newman, D.J. UCI Machine Learning Repository, Irvine, CA

Bibliography

Asuncion, A., Newman, D.J. UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science, 2007.

Bishop, C.M. Neural Networks for Pattern Recognition, Oxford University Press, 1995.

Jung, S., Sobanjo, J., Munoz, G.J. Visualization and Assessment of the Aging Infrastructure Using Self-Organizing Map, 19 Computational Specialty Conference, 2010.

Neural Network Toolbox 6.0.4, Crab Classification, The Mathworks Inc., 2010.

Sinha, M., Kennedy, C.S, Ramundo, M.L. Artificial neural network predicts CT scan abnormalities in pediatric patients with closed head injury, U.S. National Library of Medicine, 2001.

APPENDIX

Table 1: Character Representation of Mushroom Characteristics

Mushroom Characteristics								
Cap-Shape	Cap-Surface	Cap-Color	Bruises	Odor	Gill Attachment	Gill Spacing	Gill Size	Gill Color
bell	fibrous	brown	yes	almond	attached	close	broad	black
conical	grooves	buff	no	anise	descending	crowded	narrow	brown
convex	scaly	cinna-mon		creosote	free	distant		buff
flat	smooth	gray		fishy	notched			choco-late
knbbed		green		foul				gray
sunken		pink		musty				green
		purple		none				orange
		red		pungent				pink
		white		spicy				purple
		yellow						red
								white
								yellow

Mushroom Characteristics								
Stalk Shape	Stalk Root	Stalk surface above ring	Stalk surface below ring	Stalk color above ring	Stalk color below ring	Veil type	Veil color	Ring number
enlarg-ing	bulbous	fibrous	fibrous	brown	brown	partial	brown	none
tapering	club	scaly	scaly	buff	buff	univer-sal	or-ange	one
	cup	silky	silky	cinna-mon	cinna-mon		white	two
	equal	smooth	smooth	gray	gray		yellow	
	rhizo-morphs			orange	orange			
	rooted			pink	pink			
	missing			red	red			
				white	white			
				yellow	yellow			

Mushroom Characteristics			
Ring type	Spore print color	Population	Habitat
cobwebby	black	abundant	grasses
evanescent	brown	clustered	leaves
flaring	buff	numerous	meadows
large	chcolate	scattered	paths
none	green	several	urban
pendant	orange	solitary	waste
sheathing	purple		woods
zone	white		
	yellow		

Table 2: Numerical Representation of Mushroom Characteristics (*Simplification*)

Mushroom Characteristics								
Cap-Shape	Cap-Surface	Cap-Color	Bruises	Odor	Gill Attachment	Gill spacing	Gill size	Gill color
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3		3	3	3		3
4	4	4		4	4			4
5		5		5				5
6		6		6				6
		7		7				7
		8		8				8
		9		9				9
		10						10
								11
								12

Mushroom Characteristics								
Stalk Sahpe	Stalk Root	Stalk surface above ring	Stalk surface below ring	Stalk color above ring	Stalk color below ring	Veil type	Veil color	Ring number
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
	3	3	3	3	3		3	3
	4	4	4	4	4		4	
	5			5	5			
	6			6	6			
	7			7	7			
				8	8			
				9	9			

Mushroom Characteristics			
Ring type	Spore print color	Population	Habitat
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7		7
8	8		
	9		

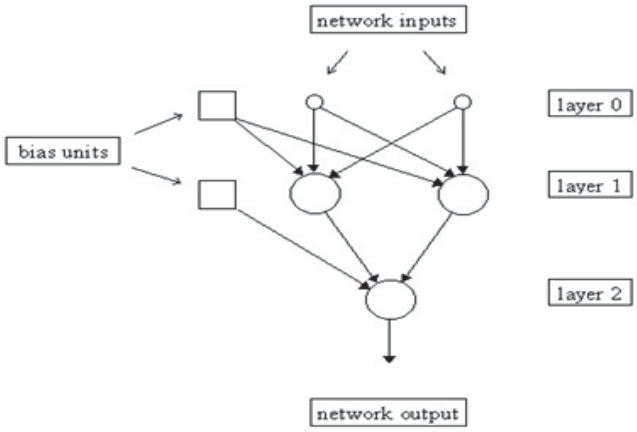

Figure 1: Two-Layer Feed-forward Artificial Neural Network

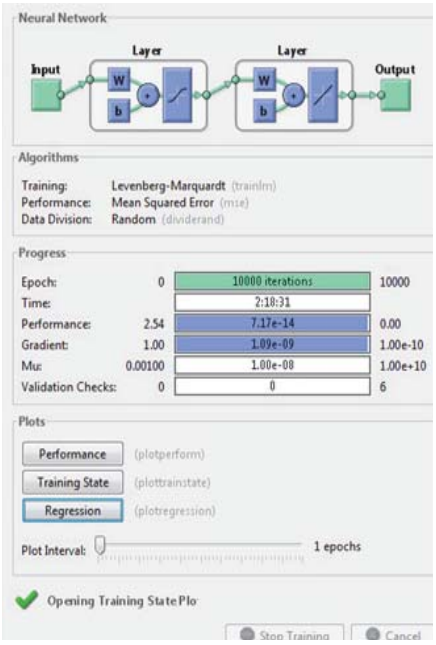

Figure 2: Neural Network Toolbox Matlab

1,000 iterations	5 Nodes	60% Training; 20% Validation 20% Testing	20% Training; 20% Validation 60% Testing	20% Training; 60% Validation 20% Testing
	20 Nodes	60% Training; 20% Validation 20% Testing	20% Training; 20% Validation 60% Testing	20% Training; 60% Validation 20% Testing
10,000 iterations	5 Nodes	60% Training; 20% Validation 20% Testing	20% Training; 20% Validation 60% Testing	20% Training; 60% Validation 20% Testing
	20 Nodes	60% Training; 20% Validation 20% Testing	20% Training; 20% Validation 60% Testing	20% Training; 60% Validation 20% Testing

Table 3: Different combinations used for Testing

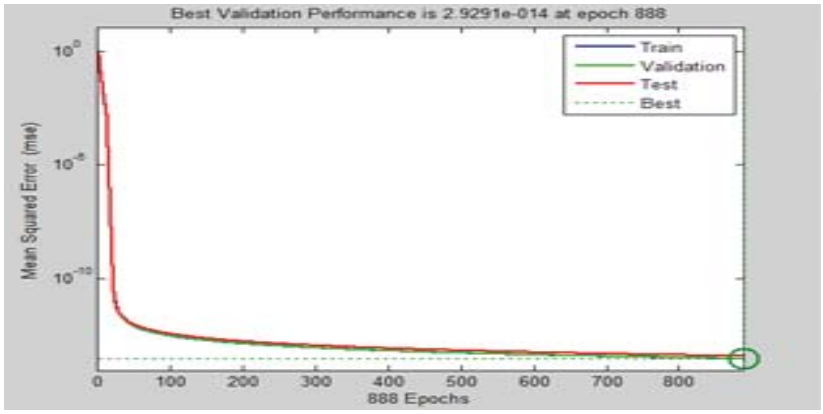

Figure 3: 1,000 iterations, 5 nodes, 60%R, 20%V, 20%T - stopped at 888 iterations

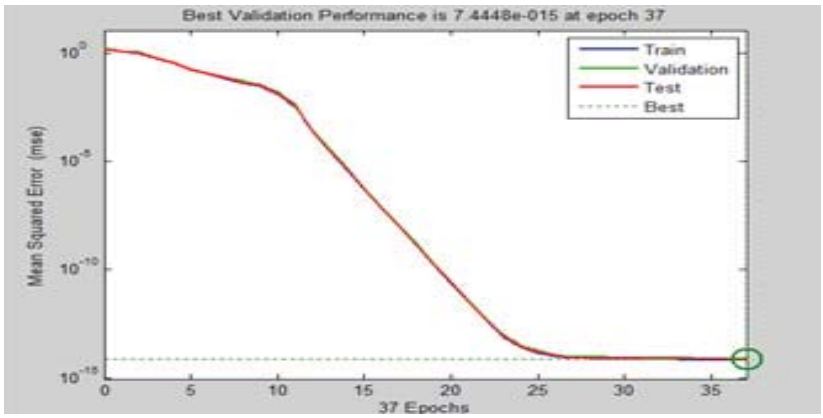

Figure 4: 1,000 iterations, 5 nodes, 20%R, 20%V, 60%T - min. gradient reached at 37 iterations

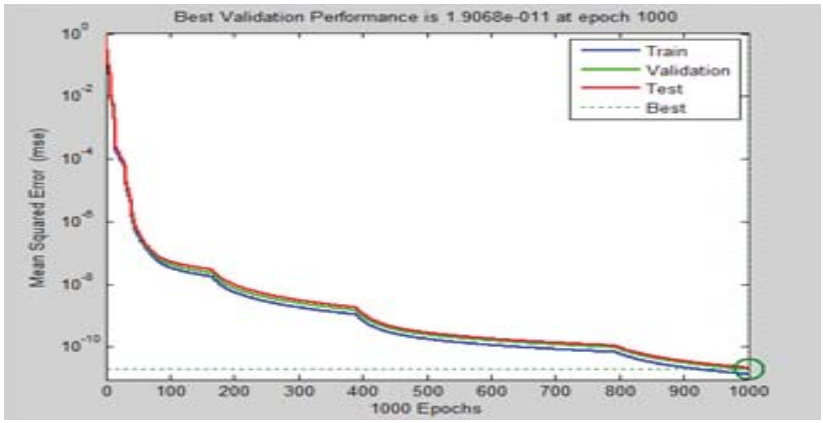

Figure 5: 1,000 iterations, 5 nodes, 20%R, 60%V, 20%T - max. epoch reached

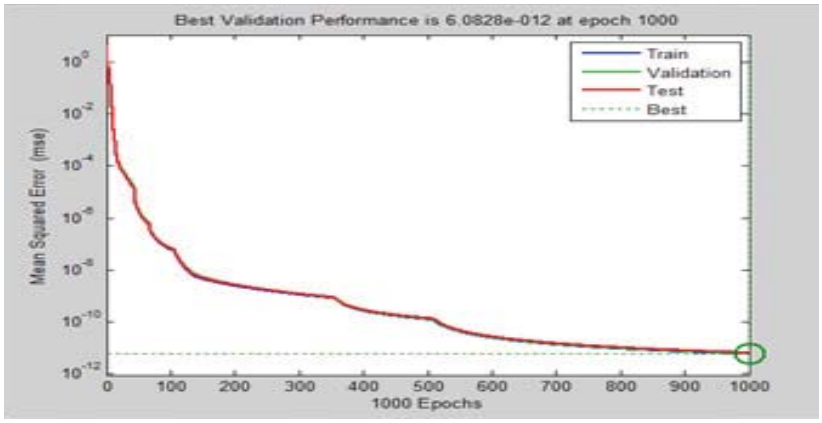

Figure 6: 1,000 iterations, 20 nodes, 60%R, 20%V, 20%T - max. epoch reached

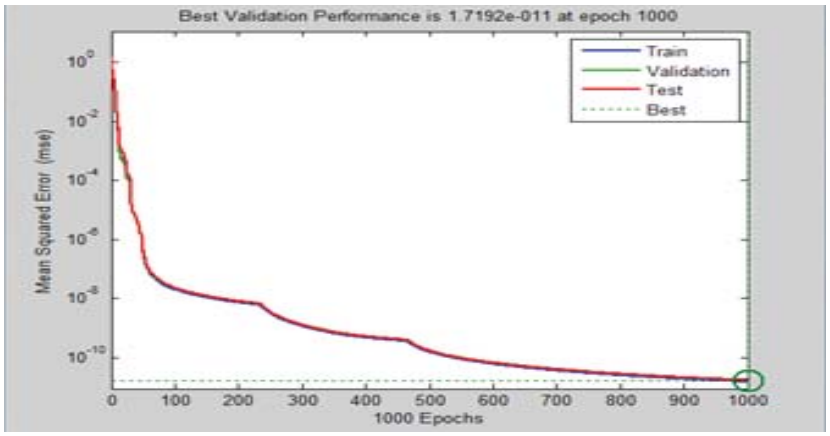

Figure 7: 1,000 iterations, 20 nodes, 20%R, 20%V, 60%T - max. epoch reached

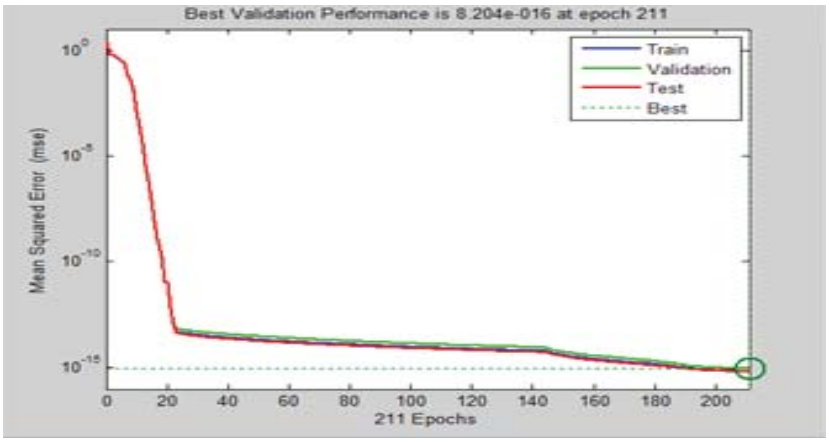

Figure 8: 1,000 iterations, 20 nodes, 20%R, 60%V, 20%T - min. gradient reached - 211 epochs

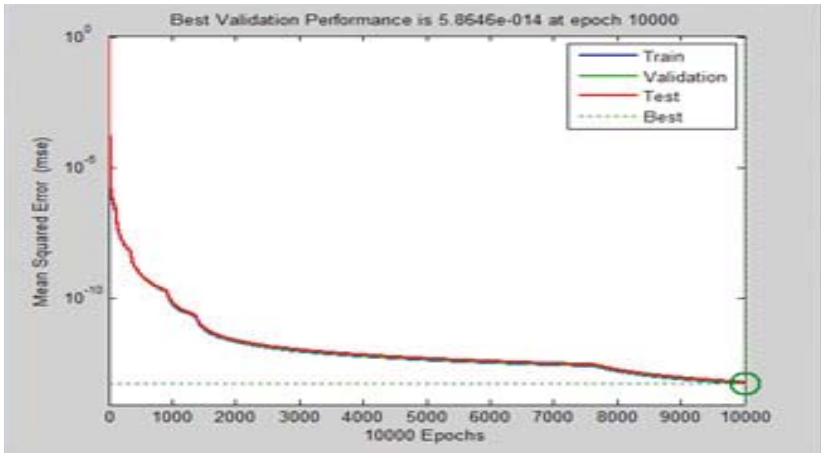

Figure 9: 10,000 iterations, 5 nodes, 60%R, 20%V, 20%T - max. epoch reached

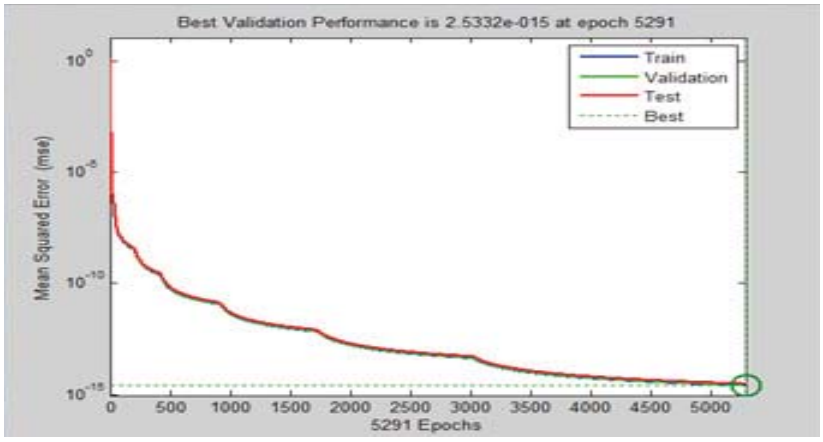

Figure 10: 10,000 iterations, 5 nodes, 20%R, 20%V, 60%T - min. gradient reached - 5,291

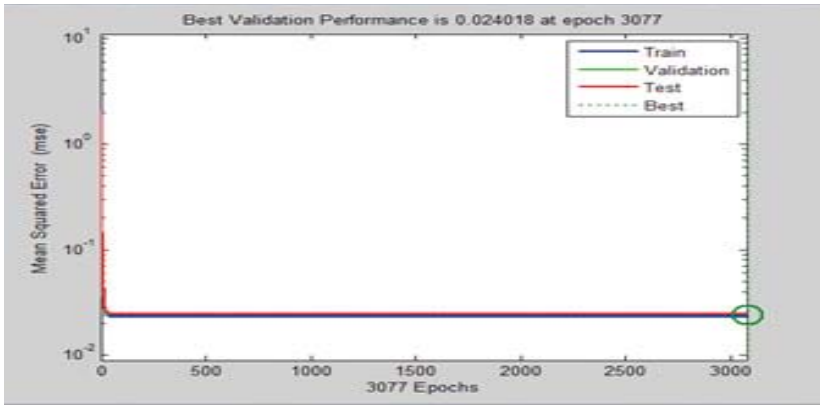

Figure 11: 10,000 iterations, 5 nodes, 20%R, 60%V, 20%T - min. gradient reached - 3,077 epochs

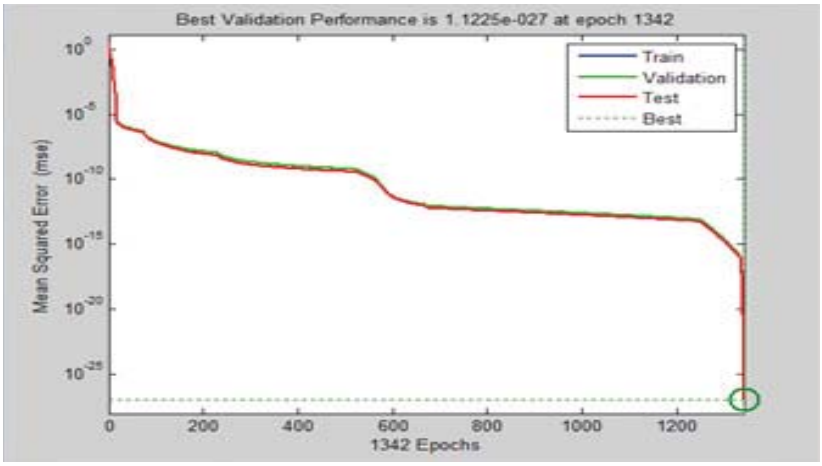

Figure 12: 10,000 iterations, 20 nodes, 60%R, 20%V, 20%T - min. gradient reached - 1,342

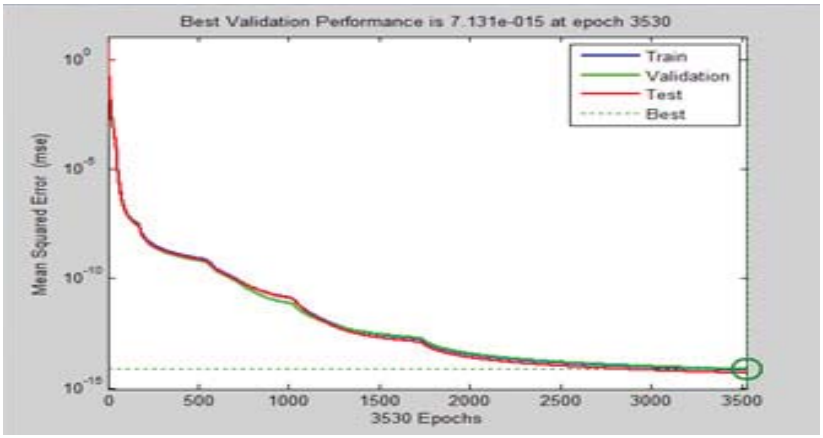

Figure 13: 10,000 iterations, 20 nodes, 20%R, 20%V, 60%T - min. gradient reached - 3,530

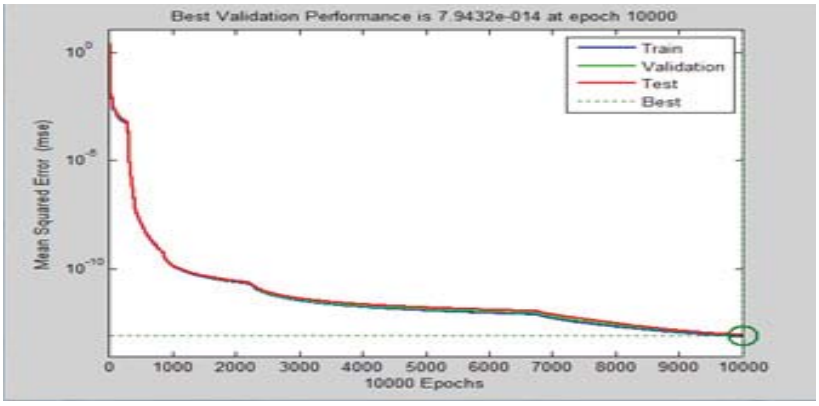

Figure 14: 10,000 iterations, 20 nodes, 20%R, 60%V, 20%T - max. epoch reached