Optimal Size of a Feedforward Neural Network: How Much does it Matter?

Lipo Wang
College of Information Engineering
Xiangtan University
Xiangtan, Hunan, China
lipowang@126.com

Hou Chai Quek
School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, 50 Nanyang Avenue, Singapore 639798
elpwang@ntu.edu.sg

Keng Hoe Tee
School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, 50 Nanyang Avenue, Singapore 639798
ecrwan@ntu.edu.sg

Nina Zhou
School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, 50 Nanyang Avenue, Singapore 639798
ZHOU0034@ntu.edu.sg

Chunru Wan
School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, 50 Nanyang Avenue, Singapore 639798
ecrwan@ntu.edu.sg

Abstract

In this paper, we attempt to answer the following question with systematic computer simulations: for the same validation error rate, does the size of a feedforward neural network matter? This is related to the so-called Occam's Razor, that is, with all things being equal, the simplest solution is likely to work the best. Our simulation results indicate that for the same validation error rate, smaller networks do not tend to work better than larger networks, that is, Occam's Razor does not seem to apply to feedforward neural networks. In fact, our results show no trend between network size and performance for a given validation error.

1. Introduction

Artificial neural networks represent a technology rooted in many disciplines. They are endowed with some powerful attributes: universal approximation of any arbitrary input-output mapping, the ability to learn from and adapt to their environment, and the ability to generalize from incomplete knowledge based on input data.

A feedforward neural network, or multilayer perceptron (MLP), shown Fig.1 has an input layer, an output layer, and one or more hidden layers. This type of networks can be trained with the error back propagation algorithm and its many variants. Since a neural network with one hidden layer is capable of approximating any arbitrary function, in this paper we will consider only neural networks with only a single layer of hidden layer.

One of the most important problems in neural network design is deciding on the optimal size of the hidden layer (or the optimal number of hidden neurons) in order to achieve the best performance for a given application. In this paper, we will concentrate on classification. For a given data set, how does one decide on the optimal size of the hidden layer (or the optimal number of hidden neurons) that leads to the highest classification accuracy for the $test$ data set after training?

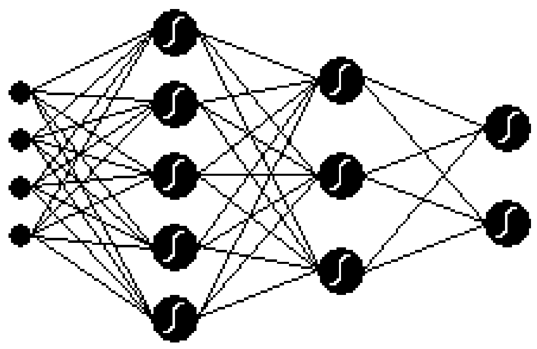

Figure 1. A Simple Multilayer Perceptron.

In general, a neural network with too small a hidden layer is unable to learn the input-output mapping, i.e., the network is not able to work sufficiently well even for the training data, no matter how many epochs the network is trained. However, if too many hidden neurons are used, overfitting can occur, that is, although the network can be trained to work very well for the training data, it performs poorly for test data not used in training, or in other words, the network is not able to generalize well.

In order for the network to learn the desired mapping and at the same time avoid overfitting, validation or early stopping is often used: one uses a sufficiently large network, and one uses a part of the training data to check the classification accuracy during training, but these validation data are not used to adjust network weights. Training stops if this validation error (rather than the error for the training data set) reaches a minimum.

To minimize the risk of overfitting, one is tempted to use the smallest network possible. This is also consistent with the Occam's razor, the essence of which is that all things being equal, the simplest solution tends to work the best. As pointed out by Poggio and Girosi, smoothness of input-output mapping is closely related the Occam's razor, since the smoothest mapping is the simplest. A regularization term may be added in the training criterion to explicitly penalize the network complexity, which is also equivalent to adding noise in training data, as shown by Bishop.

Caruana presented a tutorial at [5] with generalization results on a variety of problems as the size of the networks was varied. Caruana reported that large networks rarely do worse than small networks on the problems investigated. Caruana thus suggested that "backpropagation ignores excess parameters".

Several theories for determining the optimal network size have been proposed, e.g. the NIC (Network Information Criterion) [9] which is a generalization of the AIC (Akaike Information Criterion) [2][3] widely used in statistical inference, the generalized final prediction error (GPE) [8], and the Vapnik-Chervonenkis (VC) dimension [7][1][4] which measures the expressive power of a network. NIC relies on a single minimum and can be unreliable when there are several local minima [10]. Very little computational experience of the NIC, the GPE, or the VC dimensions has been published, since their evaluation can be very expensive for large networks.

Lawrence, Giles, and Tsoi carried out a large scale numerical study on the optimal size [6]. Similar to Caruana [5], they also found that networks with sizes larger than optimal size can sometimes generalize better. They noted that this is not in contradiction to Occam's Razor, since these different networks are solutions of different quality.

In this paper, we will carry out comparisons between networks of different sizes but the same quality. If we have trained an assemble of networks with different numbers of hidden neurons, but with the same validation error, Occam's razor indicates that the smaller networks would work better than the larger networks during testing. We perform systematic simulations on some benchmarking data sets to show that this trend is not apparent.

2. Simulations

Tests were conducted on the Iris Plants data from the *Machine Learning Repository of University of California Irvine* (http://www.ics.uci.edu/ mlearn/MLRepository.html).

The Iris Plants data contains 3 classes of 50 instances each, where each class refers to a type of iris plants. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other. The 3 classes are *Iris Setosa, Iris Versicolour* and *Iris Virginica*.

We used 2 training functions in the MATLAB Neural Network Toolbox to train the network, namely, traingdx and trainlm. They implement the gradient descent with momentum and adaptive learning rate, and the Levenberg-Marquardt back-propagation algorithms, respectively. We carried our simulations in the following steps:

1. The data sets were randomly divided into 3 sets, i.e., 50% of the entire data set for training, 25% for validation, and 25% for test.

2. A single-hidden layer MLP with 2 hidden neurons, 3 output neurons (one for each class), and an appropriate number of input neurons (equal to the number of input attributes) was created. The network weights and bias were then initialized randomly.

3. The network was trained using the training data set and was validated using the validation data set. We first used the Levenberg-Marquardt backpropagation algorithms. Early stopping was used and the lowest validation

error was recorded.

4. The network was then tested using the test data set and the test error was recorded.

5. Repeat Steps 3 - 4 using the gradient descent with momentum and adaptive learning rate.

6. Repeat Steps 2 - 5 for 100 different initializations of weights and bias.

7. Repeated Steps 1 - 6 for 3 different random shuffling of the data.

8. Repeat Steps 1 - 7 for different numbers of hidden neurons, i.e., 3 - 15.

9. The recorded validation errors were then sorted at a 2% interval ranging from 0% -20% and the corresponding test errors were plotted against the number of hidden neurons (each simulation was denoted by a $*$ in the following figures). For each size of hidden layer, the mean of all the test errors was calculated and was added to the plots (a circle).

2.1. Results for Iris Data Set and Trainlm

We note that each $*$ in the figures may represent more than one simulations, i.e., these simulations ended up with the same test error. For example, in Fig.2, with 2 hidden neurons, the average test error for all simulations is about 1.2% which means that many simulations resulted 0% test error.

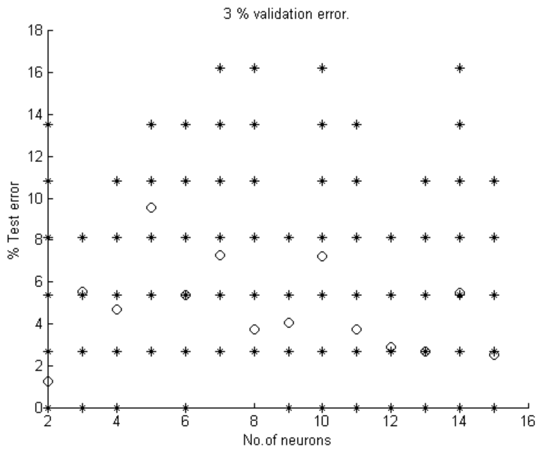

Figure 2. Test errors vs. number of hidden neurons for the Iris Data Set and Trainlm MATLAB function, with 3% validation error. Circles: average test error corresponding to a given number of hidden neurons.

There is no obvious trend in the plots that may indicate for a give validation error whether the classification accuracy during test either increases or decreases with the number of hidden neurons.

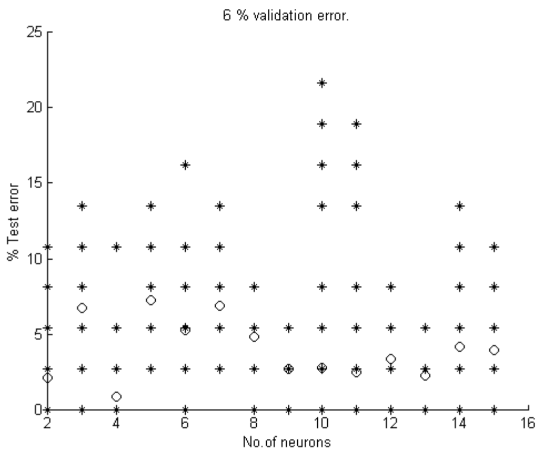

Figure 3. Same as Fig.2, for 6% validation error.

Due to limited space, not all simulation results are included here; however, the above observation stands for all our simulations.

2.2. Results for Iris Data Set and Traindx

Similar results using the gradient descent with momentum and adaptive learning rate training algorithm are presented Figures 8-13.

3. Discussion

In this paper, we have conducted systematic numerical experiments to find out whether small-sized neural networks perform better than larger networks for the same valid accuracy, i.e., whether or not Occam's Razor holds for neural network training. From our results, there are trends at all to support Occam's Razor. This paper presents results only for the Iris data; however, we have also carried out experiments with other datasets and the results are similar. From the result we have obtained, there is no indication to convince us that Occam's Razor is valid for neural networks. We claim from our results that there is no relationship between the network size and its performance, that is, Occam's Razor does not apply for neural networks.

References

[1] Y. Abu-Mostafa. The vapnik-chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1(3):312–317, 1998.

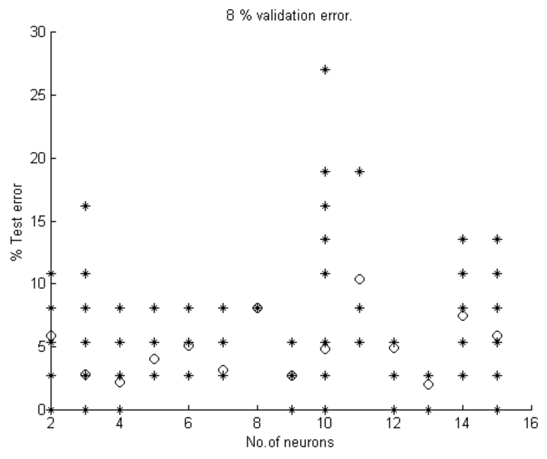

Figure 4. Same as Fig.2, for 8% validation error.

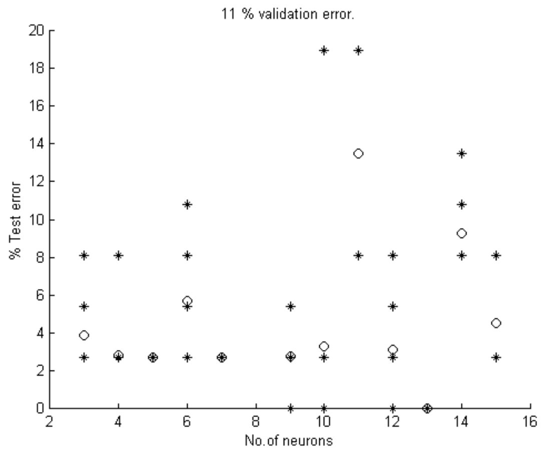

Figure 5. Same as Fig.2, for 11% validation error.

[2] H. Akaike. Statistical predictor identification. *Annals of the Institute for Statistical Mathematics*, 22:203–217, 1970.

[3] H. Akaike. Information theory and an extention of the maximum likelihood principle. *B.N. Petrov and F. Csaki, eds. Proceedings 2nd International Symposium on Information Theory*, pages 267–281, 1973.

[4] P. Bartlett. Vapnik-chervonenkis dimension bounds for two- and three-layer networks. *Neural Computation*, 5(3):371–373, 1993.

[5] R. Caruana. *Generalization vs. net size, Neural Information Processing Systems*. Tutorial, 1993.

[6] S. Lawrence, C. Giles, and A. Tsoi. What size neural network gives optimal generalization? convergence properties of backpropagation. *University of Maryland Technical Report*, UMIACS-TR-96-22, 1996.

[7] W. Maass. Vapnik-chervonenkis dimension of neural networks. *in M.A. Arbib, ed., The Handbook of Brain Theory and Neural Networks*, 1995.

[8] J. Moody. The effective number of parameters: An analysis of generalization and regularization learning systems. *in J. Moody, S.J. Hanson, and R.P. Lippmann, eds., Advances in Neural Information Processing*, 4:847–854, 1992.

[9] K. Muller, M. Finke, K. Schulten, N. Murata, and S. Amari. A numerical study on learning stochastic multi-layer feedforward networks. *Neural Computation*, 1996.

[10] B. Ripley. Statistical ideas for selecting network architectures. *Invited Presentation, Neural Information Processing Systems 8*, 1995.

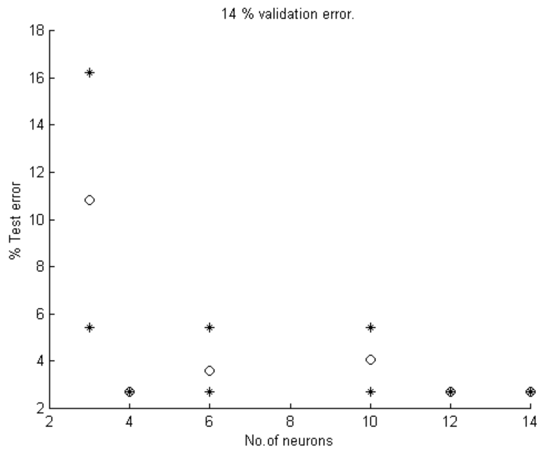

Figure 6. Same as Fig.2, for 14% validation error.

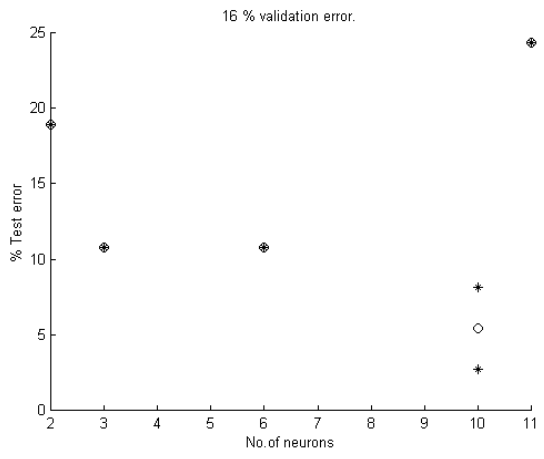

Figure 7. Same as Fig.2, for 16% validation error.

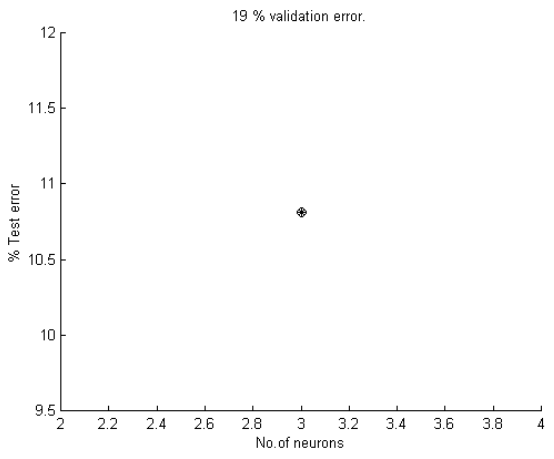

Figure 8. Test errors vs. number of hidden neurons for the Iris Data Set and Traindx MATLAB function, with 3% validation error. Circles: average test error corresponding to a given number of hidden neurons.

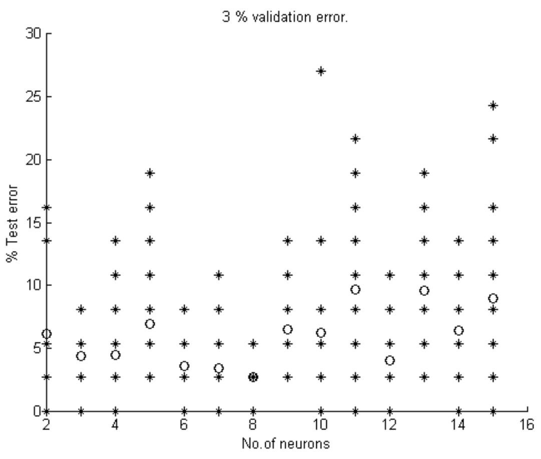

Figure 9. Same as Fig.8, for 6% validation error.

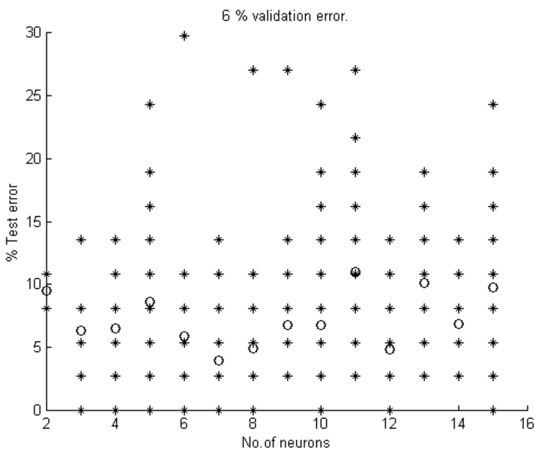

Figure 10. Same as Fig.8, for 8% validation error.

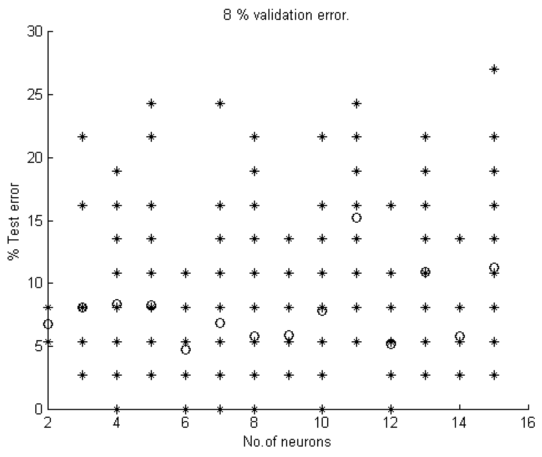

Figure 11. **Same as Fig.8, for 11% validation error.**

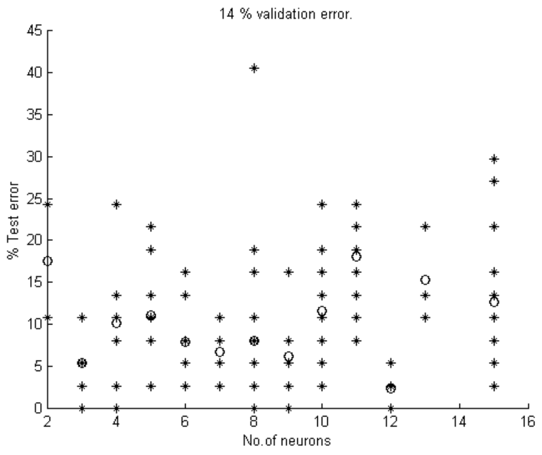

Figure 13. **Same as Fig.8, for 16% validation error.**

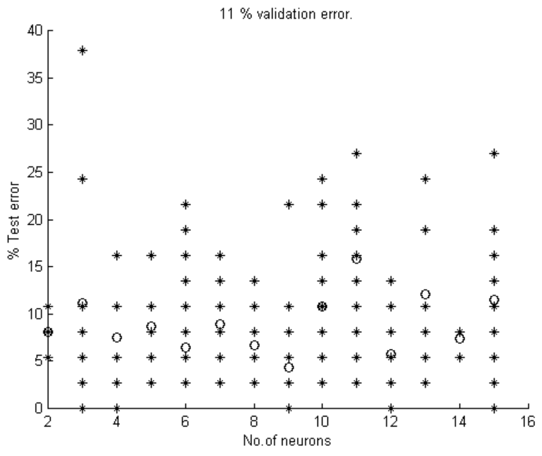

Figure 12. **Same as Fig.8, for 14% validation error.**