The Minimum Number of Hidden Neurons Does Not Necessarily Provide the Best Generalization

Jason M. Kinser
The Institute for Biosciences, Bioinformatics, and Biotechnology
George Mason University, MSN 4E3
10900 University Blvd
Manassas, VA 20110 USA

Voice: 703-993-3758
Fax: 703-993-8447
Email: jkinser@gmu.edu

Abstract

The quality of a feedforward neural network that allows it to associate data not used in training is called *generalization*. A common method of creating the desired network is for the user to select the network architecture (largely based on the selecting the number of hidden neurons) and allowing a training algorithm to evolve the synaptic weights between the neurons. A popular belief is that the network with the fewest number of hidden neurons that correctly learns a sufficient training set is a network with better generalization. This paper will contradict this belief. The optimization of generalization requires that the network not assume information that does not exist in the training data. Unfortunately, a network with the minimum number of hidden neurons may require assumptions of information that does not exist. The network then *skews* the surface that maps the input space to the output space in order to accommodate the minimum architecture which then sacrifices generalization.

I. INTRODUCTION

It has been previously proposed that the feedforward neural network solution with the fewest possible number of hidden neurons is that network that best generalizes to the problem encompassed by the training data set. [Reed,93] [Murata,94] It will be proposed here that this is not the case. A contradictory case with discussion will be presented that demonstrates that too few hidden neurons deteriorates generalization even though the training data is correctly mapped by the network. The cause of this deterioration is that the network, by having too few hidden neurons, is forced to have improper hidden neuron activation. This requires the network to infer information that does not exist in the training data and thus generalization is compromised.

II. CONTRADICTORY EXAMPLE

The networks considered here are feedforward networks with simple neurons. The network is shown in Fig. 1 and the operation of the network is described by,

$$z_i = \Gamma\left\{\sum_j u_{ij} y_j + \sum_k v_{ik} x_k\right\} \qquad (1)$$

where x_k is the state of the k^{th} input neuron, y_j is the state of the j^{th} hidden neuron, u_{ij} are the weights from the input layer to the output neuron, and v_{ik} describes the connections between the hidden neurons and the output neuron. $\Gamma\{u\}$ is a hard threshold function which is 1 if u is greater than a constant γ and 0 otherwise. The hidden neurons are similarly computed by,

$$y_j = \Gamma\left\{\sum_l w_{jl} x_l\right\} \qquad . \qquad (2)$$

To demonstrate the assertion of this paper a single example problem shown in Table 1 is considered. There are eleven training pairs each with four input neurons and a single output neuron. Following the logic in [Kinser,96] conflicts exist in the data set and it cannot be solved by a single layer system. At least one hidden neuron is required to fully map this data set, or in other words, this data set is higher order.

TABLE 1.

T	Input	Output
1	0000	0
2	0001	1
3	0010	1
4	0100	1
5	1000	1
6	0101	1
7	1001	1
8	0110	1
9	1010	1
10	0011	0
11	1100	0

A solution of the network correctly learns all of the training data. The two solutions that will be used for discussion are:

Solution 1: One hidden neuron.

$$u = (-0.8)$$

$$v = (0.68, 0.61, 0.64, 0.64)$$

$$w = (0.3, 0.21, 0.27, 0.24)$$

Solution 2: Two hidden neurons

$$u = (-1, -1)$$

$$v = (0.6, 0.6, 0.6, 0.6)$$

$$w = \begin{pmatrix} 0.3 & 0.3 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.3 \end{pmatrix}$$

Both solutions completely associate the training data. However, solution 1 infers information that does not exist in the training data. To explore this, consider the symmetry in the training data. In this set the reverse of each input vector produces the same output. This data set is complete in that every training input vector has its reverse in the training set. The data is wholly symmetric.

The concern with solution 1 is that it is not symmetric. In fact, it is not possible to create a single hidden neuron solution to this data set that is symmetric. For this explanation each training pair is denoted by T_k, and T_1 implies that the output neuron is not receiving a steady state bias.

Following the logic in [Kinser,96] the hidden neuron is not added to the system until it is necessary. This provides a starting point for the algorithm, and deviations from this do not alter the necessity or role of a hidden neuron, but just alter connection weights.

T_1, T_2, T_3 and T_4 each require that each element in **v** is greater than γ. The starting position of not activating the hidden neuron until necessary requires that all elements of **w** be less than γ. In order to produce a symmetric network $w_1 = w_4$, $w_2 = w_3$, $v_1 = v_4$, and $v_2 = v_3$. T_{10} without a hidden neuron requires that $v_1 + v_2 < \gamma$ which can not simultaneously be true with $v_1 > \gamma$ and $v_2 > \gamma$. Therefore, T_{10} (and similarly T_{11}) requires the creation and activation of the hidden neuron. Thus, $v_1 + v_2 + u < \gamma$ and $w_1 + w_2 > \gamma$ (and similarly $v_3 + v_4 + u < \gamma$ and $w_3 + w_4 > \gamma$).

Since $w_1 + w_2 > \gamma$ then either w_1 or w_2 must be greater than $\gamma/2$, and similarly w_3 or w_4 must be greater than $\gamma/2$. The fact that there are four possibilities from which to chose (w_1, $w_3 > \gamma$; w_2, $w_3 > \gamma$; w_1, $w_4 > \gamma$; or w_2, $w_4 > \gamma$) will be discussed later. So, for the sake of argument, w_1 and w_3 are chosen as the weights that must be greater than $\gamma/2$. With this selection T_9 must activate the hidden neuron which produces $v_1 + v_3 + u > \gamma$. Due to symmetry ($v_2 = v_3$) this becomes $v_1 + v_2 + u > \gamma$, which cannot be simultaneously true with inequalities and the end of the previous paragraph. The selection of which of the w weights is greater than $\gamma/2$ does not alter the existence of the contradictory inequalities. This means that a symmetric single-hidden neuron solution does not exist.

The data set is symmetric, but the single hidden neuron solution is necessarily not symmetric. For symmetric data a non-symmetric solution imposes information that does not exist in the training data. The mapping surface created by the network is skewed by this unwarranted infusion of improper information. A network with optimal

generalization will not have such skews in the mapping surface. In the example, generalization is compromised by the single neuron solution.

In the two hidden neuron solution the weights and therefore the mapping surface are symmetric and the generalization is not compromised. The conclusion drawn from this is that Solution 2 is a better generalizing network than Solution 1. Thus, a network solution for this problem with the minimum number of hidden neurons can not construct the network that is optimal in the sense of generalization.

III. Symmetry Discussions

The question then becomes, if the data has symmetry in it then where did the symmetry go in the case of the single hidden neuron?

In the example problem there are four single hidden neuron solutions. The particular solution selected used w_1 and w_3 as the weights chosen to be greater than $\gamma/2$. The other three solutions would have selected different weights to be greater than $\gamma/2$. Each of these solutions is unique from each other in that they have different training vectors activating the hidden neuron. In each of these cases, T_{10} and T_{11} activate the hidden neuron as well as two from the set $\{T_6, T_7, T_8, T_9\}$ depending on which solution was chosen. The symmetry does still exist in that there are four solutions symmetrically located in a solution space. However, each particular solution does not have symmetry within itself.

Activation of the hidden neuron by T_6, T_7, T_8, or T_9 is improper since none of these are required to activate the hidden neuron. This is easily seen by removing T_{10} and T_{11} from the data set and finding a easy solution for the rest of the data with a network that has no hidden neuron ($v_i = 0.3$ for all i). T_{10} and T_{11}, however, are required to activate the hidden neuron as determined in the previous section, but their presence should not require the unnecessary activation of the hidden neuron by other training vectors.

Incorrect activation again is a sign that damage is being done to the generalization ability of the network. The use of hidden neurons is required for a mapping problem that requires a higher-order mapping. In the case of T_{10} and T_{11} being removed from the training data a first-order mapping (no hidden neurons) is sufficient, and therefore all of these trainers are considered first-order or less. In the case of Solution 1, two first order trainers are forced to activate the hidden neuron even though they are not the higher-order trainers in the data set. Again, information not existent in the data set is being imposed by improper activation of the hidden neuron.

In Solution 2 only T_{10} and T_{11} activate the hidden neurons and thus no damage is imposed on the generalization of the network. There is no indication that non-existent information is being imposed on the network.

III. SUMMARY

The best generalizing neural network is not necessarily the network with the fewest number of hidden neurons. This was demonstrated by presenting a case in which the fewest number of hidden neurons was deleterious to the generalization ability of the system. This destruction of the generalization occurs when the mapping surface of the network is skewed (or is forced to assume information that does not exist in the training data). Skewing is product of the improper activation of the hidden neuron.

IV. REFERENCES

R. Reed, "Pruning Algorithms - A Survey", IEEE Neural Nets NN-**4**(5) 74-747 (1993).

N. Murata, S. Yoshizawa, S. Aman, "Network Information Criterion - Determining the Number of Hidden Units for an Artificial Neural Network Model", IEEE Trans on Neural Nets, NN-**5**(6), 865-872 (1994).

J. M. Kinser, “The Determination of Hidden Neurons”, Optical Memories and Neural Networks, **5**(4), 245-262 (1996).