A Method for Optimizing the Structure of Neural Networks Based on Information Entropy

Yuan Hongchun Xiong Fanlun Bai ShiLei

Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031
hcyuan@mail.iim.ac.cn flxiong@163.net baisl@mail.iim.ac.cn

Abstract

The number of hidden neurons of the feed-forward neural networks is generally decided on the basis of experience. The method usually results in the lack or redundancy of hidden neurons, and causes the shortage of capacity for storing information or learning overmuch. This research proposes a new method for optimizing the number of hidden neurons based on information entropy. Firstly, an initial neural network with enough hidden neurons should be trained by a set of training samples. Second, the activation values of hidden neurons should be calculated by inputting the training samples that can be identified correctly by the trained neural network. Third, all kinds of partitions should be tried and its information gain should be calculated, and then a decision-tree for correctly dividing the whole sample space can be constructed. Finally, the important and related hidden neurons that are included in the tree can be found by searching the whole tree, and other redundant hidden neurons can be deleted. Thus, the number of hidden neurons can be decided. In the case of building a neural network with the best number of hidden units for tea quality evaluation, the proposed method is applied. And the result shows that the method is effective.

Keywords :

Neural network ; Information entropy ; Optimization-structure

Introduction

Neural networks are widely used in many areas because of its strong capacity for non-linear mapping, its high accuracy for learning and its good robustness. A multi-layer feed forward neural network is a type of very important network, but it is very difficult to effectively decide the number of hidden neurons. At present, in many applications of networks, the main method to select the number of hidden neurons is as follows. Firstly, a series of neural networks with different number of hidden neurons are tested, then their inductive errors are evaluated, finally, one of them, which is the best one, is selected. This method cost much time and has great limitation. Some people propose that such experience as "the number of input neurons add the number of output neurons, then be divided by 2" could be taken to decide the number of hidden neurons. This method depended on the number of input neurons and output neurons is not scientific, because factors to affect networks' structure are the number of samples in a training set, the noisy size of samples and the complex degree of function or classification to learn, and so on. Some people also suggest that the maximal number of hidden neurons can be made according to the number of samples. This experiential method only can solve the problem of over-learning. Reference 1 puts forward a monotone index based method to directly estimate the number of hidden neurons in a three-layer feed forward network. This method requires a good set of sample set, and is not suitable to a small set of samples or multi-input and multi-output issue.

This research has proposed a novel method that the number of hidden neurons can be decided based on a decision tree algorithm. The goal of this research is to overcome above mentioned problems, and to avoid the over-learning problem because of over number of the hidden neurons, and to avoid the shortage of capacity because of less number of hidden neurons. Next parts will discuss the proposed method and present an example.

Problem Description and Algorithm Designation

Problem description: if there is a training set $S=<C,r>$, $C=\{C_i|1\leq i\leq k,\ i\in N,\ k=\text{the number of condition attributes}\}$, $r\in R$, $R=\{R_j|1\leq j\leq m, j\in N, m=\text{the number of classes}\}$, then a three-layer feed forward neural network can be built to approximate the training data. The number of input neurons is k. The number of output neurons is decided on the basis of m. If m>2, then the number of output neurons is m. If m=2, then the number of output neurons is

Supported by National Nature Science Fund of China (No.69835010)

1. How to estimate the number of hidden neurons is what this paper will discuss.

Toward above problem, this research proposes a method that using decision tree algorithm to decide the number of hidden neurons. The basic idea is as follows. Firstly construct an initial network with enough hidden neurons according to the number of training samples. Secondly train the network. Thirdly built up a decision tree, which can be used to divide the whole sample space based on the activation values of hidden neurons. And then search through the whole tree to find out important hidden neurons, which take part in correctly dividing sample space. Fourthly, take these neurons as hidden neurons of the final network. Thus the best number of hidden neurons can be made.

The algorithm, which decide the number of hidden neurons of network, can be described as follows:
Input: training set S=<C, r>
Output: the best number of hidden neurons of a neural network.
Algorithm steps:
Step 1: data preprocessing, such as filling missing data, standardizing data and so on.
Step 2: construct the topology structure of an initial network. The number of input neurons can be decided based on the number of condition attributes. The number of output neurons can be decided based on the number of classes. And the number of hidden neurons can be given enough according to the number of samples.
Step 3: initialize the original network.
Step 4: train the network until the error is less than predetermined value.
Step 5: select samples, which can be identified by the trained network, from the training set.
Step 6: calculate the activation value of the selected samples.
Step 7: sort all activation values of each hidden neuron, and make any possible partitions, then calculate their information gains to build a decision tree which can be used to divide sample space.
Step 8: search the decision tree to find out important hidden neurons, which are used in the tree.
Step 9: sum up the number of important hidden neurons, and take the result as the best number of hidden neurons.

Next parts will further discuss network training, decision tree constructing and important hidden neuron identification related to the above algorithm.

Network Training

Given a sample p, p=1,2,...,P, then the output value of the network and the activation values of the hidden neurons can be calculated as follows:

$$S_{ip} = \sigma(\sum_{j=1}^{J} V_{ij} H_{jp} - \theta_i) \qquad (1)$$

$$H_{jp} = \sigma(W_j X_p) = \sigma(\sum_{k=1}^{K} W_{jk} X_{kp} - T_j) \qquad (2)$$

Where, $x_{kp} \in [0,1]$ is the value of input neuron k of the given sample. w_{jk} is the connection weight from input neuron k to hidden neuron j. V_{ij} is the connection weight from hidden neuron j to output neuron i, T_j is the threshold value of hidden neuron j. θ_i is the threshold value of output neuron i. And $\sigma(\xi) = 1/(1+e^{-\xi})$. J and K are the numbers of hidden neurons and input neurons respectively. Each sample x_p is one of possible classes such as $C_1, C_2, \ldots, C_c$. t_{ip} can be marked as the teaching value of sample p in the output neuron i. Toward two-value classification issue, a neuron, which value is 1 or 0, can be used. Toward the classification issue of c>2, the number of output neurons can be c. If sample p belongs to c class, then $t_{cp}=1$, $t_{ip}=0(\forall i \neq c)$.

Outstretched cross-entropy error function is adopted in this research (formula 3):

$$\theta(\omega,\upsilon) = F(\omega,\upsilon) - \sum_{i=1}^{C}\sum_{p=1}^{P}[t_{ip}\log s_{ip} + (1-t_{ip})\log(1-s_{ip})] \qquad (3)$$

F(w,v) is an item of punishment:

$$F(\omega,\upsilon) = \varepsilon_1 \sum_{j=1}^{J}(\sum_{i=1}^{C}\frac{\beta\upsilon_{ij}^2}{1+\beta\upsilon_{ij}^2} + \sum_{k=1}^{K}\frac{\beta\omega_{jk}^2}{1+\beta\omega_{jk}^2}) + \varepsilon_2 \sum_{j=1}^{J}(\sum_{i=1}^{C}\upsilon_{ij}^2 + \sum_{k=1}^{K}\omega_{jk}^2) \qquad (4)$$

Where ε_1, ε_2 and β are positive parameters. Reference 2 shows that compared with standard least square error function, cross-entropy error function can be used to speed up the convergence. Reference 3 shows that punishment item can be used to accelerate the attenuation of weights. Thus, unimportant connections have small value. When it is less than a predetermined value, it can be set zero. By the method, the network connections can be simplified.

In the process of training the network, BFGS is used to minimize the outstretched cross-entropy error function, because it can speed up the convergence compared with BP algorithm.

Decision Tree Constructing and Important Hidden Neuron Identification

After training a network, a decision tree can be constructed based on calculating the information gains of all partitions of hidden neurons' activation values, and related and important neurons can be identified by searching the decision rule of no-leave node.

If there is a data set D, then a method which produce a decision tree is as follows:

(1) If there are one or more samples in D, and all samples belong to C_c, then stop partitioning.
(2) If D is NULL, then take the most frequent class of its parent node as the class of this branch, and stop partitioning.
(3) If there are samples, which belong to different classes, then D should be partitioned based on the information gain.

This research adopts the activation values of hidden neurons to construct a decision tree, and it has following features: because each activation value is obtained from calculating sigmoid function, so all activation values are located in the continuous space (0,1). Because only using the activation values of hidden neurons of correctly identified samples to construct decision tree, so the number of samples in D generally is less than the total number of samples P. Let P' represents the number of samples which can be identified by the trained network. To hidden neuron j, the activation value of sample p can be represented as H_{jp} ($p=1,2,\ldots,P'$). In the process of constructing a decision tree, firstly sort these values, and random divide them to two groups: $D_1=\{H_{j1},\ldots,H_{jq}\}$ and $D_2=\{H_{j,q+1},\ldots,H_{jp}\}$, and calculate standard information gains of all possible partitions among these activation values, then select the partition which has the maximal information gain from all partitions as the dependencies to construct a decision tree. For instance, if the standard information gain of hidden neuron j between mth sample and (m+1)th sample is maximal, and their activation values are H_{jm} and $H_{j,m+1}$ respectively, then the decision rule of root node can take $H_{ji}>(H_{jm}+H_{j,m+1})/2$, so that the activation values set of jth hidden neuron can be accordingly divided into two sub-sets, that means samples in D could be divided into left branch and right branch of the tree. The similar process can be adopted toward the two branches of root node, and such partition is continuous step by step, thus a complete decision tree can be produced.

The method to calculate standard information gain is as follows.

If there is a sample, which belongs to one of classes and exists in D, and n_c is the number of samples which belong to C_c, then its information entropy is:

$$I(D)=-\sum_{c=1}^{C}\frac{n_c}{N}\log_2\frac{n_c}{N} \qquad (5)$$

Where N is the number of samples in D. Toward the two branches of D, their information entropy can be calculated similarly:

$$I(D_1)=-\sum_{c=1}^{C}\frac{n_{c1}}{N_1}\log_2\frac{n_{c1}}{N_1} \qquad (6)$$

$$I(D_2)=-\sum_{c=1}^{C}\frac{n_{c2}}{N_2}\log_2\frac{n_{c2}}{N_2} \qquad (7)$$

Where n_{cj} is the number of samples which belong to C_c in $D_j(j=1,2)$. $N_j=\sum_{c=1}^{C}n_{cj}$. The information gain of dividing D into D_1 and D_2 is:

$$Gain(H_{jq})=I(D)-[I(D_1)+I(D_2)] \qquad (8)$$

The information gain can be standardized as follows:

$$NGain(H_{jq})=$$
$$Gain(H_{jq})/[-\sum_{j=1}^{2}(N_j/N)\log_2(N_j/N)] \qquad (9)$$

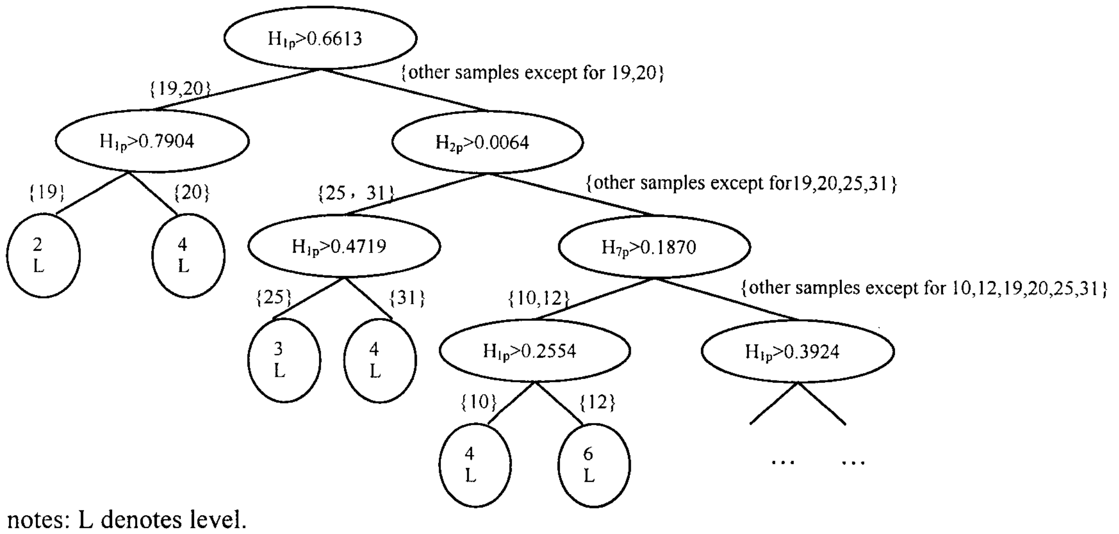
Once a decision tree has been constructed, it is easy to identify the related and important hidden neurons. The important hidden neurons are ones that are applied in the nodes of decision tree.

Example Analysis

This section will take building a neural network model with the best number of hidden neurons for reflecting the relation between tea quality and quality effecting factors as an example to introduce the application of the proposed method.

Reference [6] presents that tea quality is mainly determined by such eight factors as three kinds of chromacity values (L*, a*, b*), four kinds of chemical compositions and unit weight. The data of this example is selected from reference 6. The total number is 36. Each sample has above eight attributes and one class attribute. The quality of CHAOQING green tea is usually expressed as grades, and the total number of grades is 7. So the topology structure of the initial neural network is as follows. The number of input neurons is 8, the number of output neurons is 7, and the number of hidden neurons can be given 10 by estimation and it is greater than it should be.

The above neural network should be initialized, and its outstretched cross-entropy error function can be minimized by BFGS algorithm. When the network is convergent, the trained network can identify 36 samples correctly, so each hidden neuron has 36 activation values. Then sort these activation values, random partition them, and calculate its information gain, finally a decision tree, which correctly divides the samples space, can be produced according to the above mentioned algorithm (see Figure 1). H_{ip} in the nodes of the tree represents the activation value of the pth sample in the ith hidden neuron. Through searching the tree, 31 non-leave nodes are included in the tree, and 7 related and important hidden neurons (1,2,3,5,6,7,9) are involved. So in this case, the best number of hidden neurons is 7.



notes: L denotes level.

Fig. 1. a decision tree that can correctly divide samples space

Table 1 the relation between the number of hidden neurons and learning times

NUM_HID / ID	5	6	7	8	9	10
1	6173	4354	3430	6593	4106	7251
2	7348	7490	7197	4724	3905	6766
3	4469	6296	7409	4163	4483	8116
4	6990	9954	4235	8369	11021	7233
5	18260	11684	4788	3224	7807	3597
average	8648	7956	5412	5415	6264	6593

Notes: NUM_HID stands for the number of hidden neurons, ID stands for the serial number.

To testify the effectiveness of this result, try method is adopted. That is, different neural networks with different hidden neurons are tried, and find out the best one that its learning times is least when what the number of hidden neurons is. To overcome stochastic error, 5 times experiments have been conducted. In the process of experiments, 0.01 is adopted as the convergent threshold. The result of experiments is shown as table1.

From table 1, it can be easily found that the better number of hidden neurons is 7 or 8, and it accords with the result, which is obtained by the proposed method.

Conclusion

This paper proposes a novel method to estimate the best number of hidden neurons of a network by firstly constructing a decision tree based on calculating information gains of all partitions among hidden neurons' activation values, and then searching the decision rule of the tree's nodes to find out important hidden neurons. Related algorithm and an example are presented in the paper. Result shows that the proposed is effective. This approach is latently significant for neural networks to approximate training samples, for researching rule extraction from neural networks and for implementing neural networks using hardware.

References

[1] LI yujian (1999). *a method to directly estimate the number of the hidden neurons in the feed forward neural networks*. Journal of computers(in chinese), Vol.22.No.11, 1204-1208.

[2] A.van Ooyen, A.and B. Nienhuis, B (1992). *Improving the convergence of the backpropagation algorithm*. Neural Networks, vol.5, no.3, PP.465-471.

[3] J.Hertz,A.Krogh,and R.G. Palmer (1991). *Introduction to the theory of neural computation*. Redwood City, CA:Addison Wesley.

[4] R.Setiono(1995). *A neural network construction algorithm which maximizes the likelihood function*, Connection Science,vol.7,no.2,pp.147-166.

[5] R.Battiti(1992). *First- and second-order methods for learning: Between steepest descent and newton's method*. Neural Computation, Vol.4, PP.141-166.

[6] LinGang(1991). using multi-variable analysis method to evaluate the quality of Chinese tea and Japanese tea. doctoral thesis of MingCheng University.