Choosing an Optimal Neural Network Size to Aid a Search through a Large Image Database

K. Messer and J. Kittler
University of Surrey
Guildford Surrey GU2 5XH United Kingdom
[K.Messer|J.Kittler]@surrey.ac.uk

Abstract

In this paper a fast method of selecting a neural network architecture for pattern recognition tasks is presented. We demonstrate that our proposed method of selecting both input features and hidden neurons avoids the pitfalls exhibited by other methods reported in the literature. It is also shown that the resulting network architecture is extremely lean while at the same time significantly improving the network performance. The resulting solution provides a very useful tool which is now being incorporated in the operations system used for large image database surveys.

1 Introduction

Pattern recognition neural networks are used to obtain a non-linear function mapping between the input pattern space and the output decision space. In an L-layer feed-forward network (*in which the inputs are not considered a layer)* there are $N = \sum_{l=1}^{L}(n_{l-1}+1)n_l$ free parameters (weights and biases) that try to model this mapping, where n_l is the number of units in layer l. For example, a 25 input, 25 unit hidden layer and 2 unit output layer neural network has 702 free parameters. These parameters are estimated from the training input/output patterns. It is very important that enough training examples are available to estimate these parameters reliably. A generally accepted guideline is to have at least five to ten times the number of training patterns as free parameters. So in the above example a conservative number of samples would be 3500.

If we fail to have an adequate number of training examples there is a danger the network will learn the training set rather than build a statistical model of the process which generated the data, i.e. it will over-fit the data and lose its ability to generalise. The network will be unsuccessful in classifying previously unseen patterns correctly.

It is rarely possible to obtain the required amount of training samples and therefore some techniques are required to find an optimal network size, often referred to as complexity. This can be done by trying to find irrelevant parameters in the network structure and deleting them. It may be that some of the hidden units are not contributing to the output of the network and therefore will not be missed. Some of the feature inputs might contain highly correlated information or even irrelevant information. These can be deleted without a loss in network performance. For example, having a feature input which measured the daily amount of rainfall in China when classifying medical images will add no useful information to the classification problem.

BMVC 1998 doi:10.5244/C.12.24

Reducing network size not only helps improve generalisation. It also increases the computational speed of the system. This is usually very important for most applications. There is roughly a linear relationship between network size and computational speed. Cutting the number of units by half doubles the computational speed. It may also decrease the amount of storage required and the feature extraction time.

In this paper a fast method of selecting a neural network architecture is presented. We show that the proposed method avoids the pitfalls exhibited by other methods in the literature, mainly the potential loss of very discriminative features if the input data is highly correlated. We also demonstrate that if the proposed approach is applied to prune both the input features and the hidden neurons the resulting architecture is extremely lean and at the same time significantly enhances the network performance. The resulting solution provides a very useful tool which is now being incorporated in the operations system used for Seismic image database surveys. The problems encountered with searching through such large image databases and the search technique applied in this paper are described in more detail in [8].

In the next section several of the more common neural network complexity techniques are discussed. In section three the *Input Selection Algorithm* is presented and some results on some synthetic and seismic datasets given. This algorithm is then extended to select an optimum number of hidden units. Finally, some conclusions are drawn in section five.

2 Neural Network Complexity

Detailed reviews of the various network complexity techniques can be found in the books by *Bishop* [1] and *Haykin* [4] and in articles by *Reed* [11] and *Mao* [6]. However, some of the more important techniques are given below.

2.1 Network Growing Techniques

In general, this class of algorithms starts with a small network and adds units or connections until an adequate performance level is obtained. Algorithms such as the *The Upstart Algorithm*[1] and the *The Tiling Algorithm*[1] only work on Boolean networks. The cascade correlation algorithm [2] helps to find the optimal number of hidden units to use in a network. Traditional feature selection algorithms such as the *Sequential Floating Forward Selection*[10] and the *Sequential Forward Selection* algorithm [10] can be seen as network growing algorithms as they selectively add feature inputs to the network based on a defined criterion function.

2.2 Network Pruning Techniques

This class of algorithms starts with a fully trained large network and then attempts to remove some of the redundant weights and/or units. Hopefully, this is done in such a way that the error of the network is not significantly degraded and the generalisation will improve. Generally this involves computing a saliency measure of the individual network weights or units. The saliency can be thought of as a measure of importance of the element to the network when making a classification decision. A low saliency value implies that the weight/node is not contributing much to the networks performance. If this is the case then this element can be deleted from the network.

The *Sequential Backward Selection*[10] and *Sequential Floating Backward Selection* [10] algorithms start with a network with a large number of nodes and heuristically remove the input nodes according to the value of the criterion function. *Optimal Brain Damage*[5] attempts to remove redundant weights from a trained network by computing the Hessian matrix of the error with the network weights. This was modified by *Hassibi and Stork* [3] in the *Optimal Brain Surgeon* such that it did not assume a diagonal Hessian matrix.

In *Skeletonization, Mozer and Smolensky* [9] estimate the relevance of a unit to a network. They define a saliency measure which is the difference in the error function before and after the unit has been removed. *Mao et al* [7] defined a similar node saliency measure which was the amount of increase in the error function when a node is removed from the network. However, he considered the error E as a function of all the outputs in the network.

Ruck et al [12] proposed a node saliency measure that analysed the sensitivity of the network outputs with changing inputs. If the outputs changed dramatically for a small input change this feature was considered important to the problem.

Setiono et al [14] suggested that instead of using a saliency measure which is a function of the network weights that one could use the network classification performance on a validation dataset directly. In his algorithm the drop in the error on a verification dataset was observed whilst setting different weight values in the network to zero.

2.3 Regularization Techniques

Regularization attempts to reduce the number of effective parameters in a network. This is achieved by adding a penalty term to the network error function when training the network. This term can be considered as complexity measure of the network. A survey of regularization techniques was performed by *Mao* [6]. In general a regularization term is defined as follows.

$$\tilde{E} = E + \lambda\Omega \tag{1}$$

where λ determines the fraction of the penalty term Ω required to find the solution. Although Regularization techniques will not directly reduce the size of the network they do reduce the number of effective parameters of the network. This leads to a smoother network mapping with the benefit that the generalisation of the network will increase. However it is also possible to use Regularization techniques with pruning algorithms which then allow the network size to decrease.

Setino et al [13] proposed a regularization term which encouraged large weights not to take on very large values and small weights to converge to zero.

$$\lambda\Omega(w) = \epsilon_1 \left(\sum_i \frac{\beta(w_i)^2}{1+\beta(w_i)^2} \right) + \epsilon_2 \left(\sum_i w_i^2 \right) \tag{2}$$

3 Input Selection Algorithm

We define an input saliency measure which is the sum of the magnitude of the trained weights connecting that input to the hidden layer. The higher this sum the more important

the feature input. If regularization is used whilst training this assumption is valid. The algorithm which uses this measure to prune the network requires two labelled datasets, a training dataset D_{tr} and a verification dataset, D_{ver}. The user also needs to specify an error level E_{user} he/she is prepared to accept on the verification dataset.

The *ISA* can then be defined as follows:

- *Step 1* Train the network on all the features $Y = (y_1^0, y_2^0 ... y_d^0)$ in dataset D_{tr} using the back-propagation algorithm with the penalty term given by equation 2.
- *Step 2* Calculate the saliency, S_q, for each input node using equation 3.

$$S_q = \frac{\sum_{i=1}^{n_1} \left(w_{qi}^1\right)^2}{n_0} \tag{3}$$

- *Step 3* Find the index, min, of the minimum value of S_q and set $Y = Y - \{y_{min}^0\}$.
- *Step 4* Retrain the network using this reduced feature sub-set and calculate E_{ver} on the verification dataset.

  If $(E_{ver} \leq E_{user})$ then goto step 2,

  else set $Y = Y + \{y_{min}^0\}$, retrain the network and stop.

3.1 The Experiments

In this set of experiments the suggested method of feature input selection by the analysis of the average squared weight magnitude of the inputs is compared to the results obtained by the Sequential Backward Selection algorithm[10].

In all the experiments, the criterion function used for selecting features via the SBS algorithm was the Mean Square Error on the same verification dataset.

3.1.1 Experiment One

In this experiment we want to investigate if the important features in a given pattern recognition problem do tend to have larger weight values.

To do this, two 6-dimensional normally distributed classes were synthetically generated with the parameters shown in Table 1. Each feature was generated such that the features are statistically independent. The distance between the mean values of class A and class B is larger for feature 1 than feature 2 and that in turn exceeds the distance for feature 3 etc. This implies that feature 1 is the most important feature for this classification task. Next is feature 2 then feature 3 etc...

Next a neural network was trained using the on-line back-propagation algorithm (*BPP*) with the penalty term given by equation 2. This was repeated 20 times for different network initialisations. Next the saliency for each input was computed as given by equation 3. The average node saliency for each input is given in Table 2.

The results clearly show that the saliencies have the same order as the feature importance, i.e.

$$S_1 > S_2 > S_3 > S_4 > S_5 > S_6.$$

This suggests that the assumption that important features form stronger connections into the network is correct. It also supports the premise that the use of this saliency measure will help to determine which features to select.

Feature	1	2	3	4	5	6
Class A	2.5(1.0)	2.0(1.0)	1.5(1.0)	1.0(1.0)	0.5(1.0)	0.0(1.0)
Class B	0.0(1.0)	0.0(1.0)	0.0(1.0)	0.0(1.0)	0.0(1.0)	0.0(1.0)

Table 1: Table showing the class statistics of sythentically generated data. Number in brackets shows standard deviation.

Input	1	2	3	4	5	6
Saliency S_i	5.68(0.29)	2.85(0.24)	1.12(0.10)	0.62(0.07)	0.39(0.06)	0.28(0.05)

Table 2: Average saliency for each input into the network

3.1.2 Experiment Two

In this experiment we want to investigate what will happen if two highly correlated features are present in the feature set. Two six dimensional classes were generated using the parameters shown in table 3. This time feature f_{6a} was an exact copy of feature f_{1a} except some random noise was added to f_{6a}. Next the network was trained using the *BPP*. The node saliency was then computed for every input and ranked. This was repeated for 20 different network initialisations. The average saliency ranks can be seen in table 4.

From Table 4 we can see that the saliency measures are ranked

$$S_2 > S_6 > S_1 > S_3 > S_4 > S_5.$$

This does not match the feature ranking order of

$$f_1 == f_6, f_2, f_3, f_4, f_5$$

The network still uses the features f_{1a} and f_{6a} in obtaining a solution but instead of taking all the information from one feature and ignoring the second correlated feature its takes half the information from each feature. This can be seen in the approximately equal average magnitude weight values for input 1 and 6 (2.21 and 2.78 respectively).

This supports the ISA algorithm in that only one input feature should be thrown away at a time and after this the network must be retrained and input saliencies recomputed. In Rucks [12] algorithm which uses a similar measure more than one feature is thrown away at one time. This algorithm is dangerous as very discriminative features could be thrown away.

3.1.3 Results on the Seismic Data

A total of 25 texture features were generated for each pixel on the seismic images shown in figure 1(a),(d),(g) and (j). From each of the seismic images two datasets L and T

Feature	1	2	3	4	5	6
Class A	2.5(1.0)	2.0(1.0)	1.5(1.0)	1.0(1.0)	0.5(1.0)	2.5(1.0)
Class B	0.0(1.0)	0.0(1.0)	0.0(1.0)	0.0(1.0)	0.0(1.0)	0.0(1.0)

Table 3: Table showing the class statistics of sythentically generated data.

Input	0	1	2	3	4	5
Saliency S_i	2.21(0.13)	3.17(0.26)	1.32(0.09)	0.77(0.07)	0.28(0.04)	2.78(0.14)

Table 4: Average saliency measure for each input node when trained using the synthetically generated dataset 2.

were obtained. The regions of the seismic images which were used to obtain these training/testing datasets are shown in figure 1.

Next Sequential Backward Selection was performed on all 8 datasets and the features were ranked in the reverse order being thrown away. This experiment was then repeated 20 times for different network initialisations.

Then ISA was performed on the eight datasets. Again the features were ranked according to when they were thrown away. The experiments were repeated a total of 20 times.

Next classification of the datasets was performed, first using all 25 features, then using the top 10 ranked SBS selected features and finally using the top 10 ISA selected features. The results of this classification can be seen in table 5.

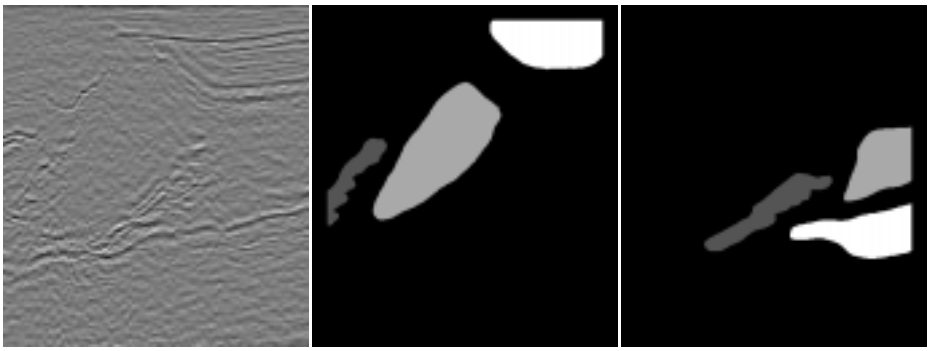
Image	Training Set	Test Set	All Features	SBS	ISA
DO.pgm	L0	T0	81.15(1.05)	82.01(0.99)	80.81(0.85)
	T0	L0	97.70(0.24)	96.98(0.90)	93.00(1.74)
D1.pgm	L1	T1	85.34(2.11)	75.00(1.87)	77.31(2.04)
	T1	L1	82.32(0.76)	89.75(0.38)	88.99(0.42)
D2.pgm	L2	T2	85.39(2.20)	87.72(1.34)	90.11(2.83)
	T2	L2	50.07(0.27)	51.60(0.41)	47.74(1.25)
D3.pgm	L3	T3	72.82(1.28)	75.86(2.76)	76.39(1.52)
	T3	L3	74.51(1.53)	68.30(1.48)	77.00(2.88)
AVERAGE			78.66(13.84)	78.40(14.20)	78.91(14.21)

Table 5: Table showing correct classification percentages on seismic images *D0-D3*, using all 25 features, the best 10 SBS selected features and the best 10 ISA selected features.

The results in table 5 show that even though the dimensionality has been halved the average classification performance has not been affected at all.

The computational time to select the best 10 features by the *SBS* is just under 2000 CPU units whilst for the *ISA* it is just under 250 units, an increase of factor 10.

The disadvantage of this method is that no guidance is given to how many hidden units should be used.



(a) Seismic image *D0.pgm*

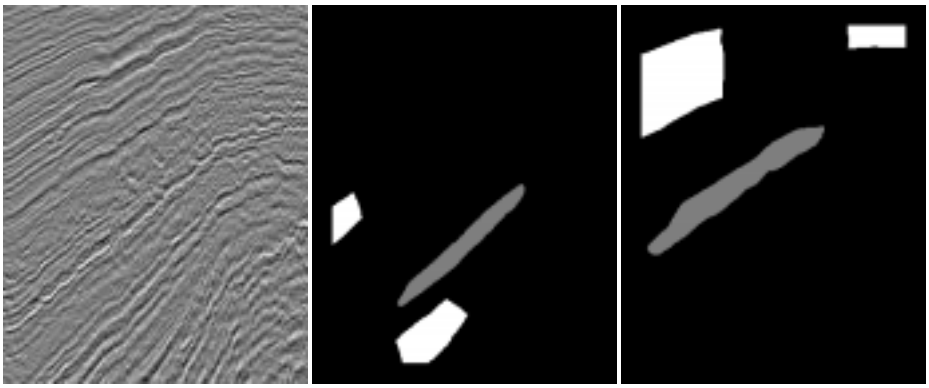
(b) Region showing the location of samples in dataset *L0*

(c) Region showing the location of samples in dataset *T0*

(d) Seismic image *D1.pgm*

(e) Region showing the location of samples in dataset *L1*

(f) Region showing the location of samples in dataset *T1*



(g) Seismic image *D2.pgm*

(h) Region showing the location of samples in dataset *L2*

(i) Region showing the location of samples in dataset *T2*

(j) Seismic image *D3.pgm*

(k) Region showing the location of samples in dataset *L3*

(l) Region showing the location of samples in dataset *T3*

Figure 1: The Seismic images and their corresponding label images.

4 The Unit Selection Algorithm

In this section we extend the *ISA* algorithm so that it also selects the number of units in the hidden layer.

Again the algorithm requires two labelled datasets, a training dataset D_{tr} and a verification dataset, D_{ver}. The user also needs to specify an error level E_{user} he/she is prepared to accept on the verification dataset.

The *USA* then proceeds as follows:

- *Step 1* Construct a network with d inputs. Set the number of hidden units, $n_1 = d$. Train this network on the complete feature set $Y = \{y_1^0, y_2^0 \ldots y_d^0\}$ using the *BPP*.

- *Step 2* Calculate the saliency, S_q, for each input node using equation 4.

$$S_q = \frac{\sum_{i=1}^{n_1} (w_{qi})^2}{n_0} \tag{4}$$

- *Step 3* Calculate the saliency, S_i, for each hidden node using equation 5.

$$S_i = \frac{\sum_{j=1}^{n_2} (w_{ij})^2}{n_1} \tag{5}$$

- *Step 4* Find the minimum value of S_q and the minimum value of S_i.

  If $min(S_q) < min(S_i)$,

  - Set $Y = Y - \{y_{min}^0\}$
  - Set $INP = True$
  - Goto step 5.

  else if $min(S_i) \leq min(S_q)$

  - Set $n_1 = n_1 - 1$
  - Set $INP = False$
  - Goto step 5.

- *Step 5* Retrain the network and calculate E_{ver} on the verification dataset.

  If $(E_{ver} < E_{user})$ then goto step 2,

  else

  - If(INP) set $Y = Y + \{y_{min}^0\}$ retrain the network and stop.
  - Else set $n_1 = n_1 + 1$ retrain the network and stop.

Image	Training Set	Test Set	#Inputs	#Hidden Units	Classification %
DO.pgm	L0	T0	$2.05(0.22)$	$2.00(0.00)$	$75.39(3.75)$
	T0	L0	$3.90(0.31)$	$2.05(0.22)$	$97.01(1.46)$
D1.pgm	L1	T1	$4.95(0.83)$	$1.25(0.44)$	$80.11(1.86)$
	T1	L1	$2.00(0.00))$	$1.00(0.00)$	$81.62(0.21)$
D2.pgm	L2	T2	$4.50(0.76)$	$1.35(0.48)$	$89.70(3.40)$
	T2	L2	$25.00(0.00)$	$1.00(0.00)$	$80.31(1.41)$
D3.pgm	L3	T3	$6.80(1.51)$	$4.00(0.85)$	$77.39(6.30)$
	T3	L3	$3.55(0.51)$	$2.00(0.00)$	$73.79(3.46)$
AVERAGE			$6.59(7.16)$	$1.83(0.99)$	$81.92(7.97)$

Table 6: Table showing average network size and correct classification percentages on seismic images *D0-D3* after using *Unit Selection Algorithm*.

4.1 The Experiments

The USA was run on the eight seismic datasets obtained from the images shown on figures 1(a),(d),(g) and (j). The classification of the test dataset was then performed using the reduced network. As always, this was repeated a number of times to obtain an average performance. The results can be seen in table 6.

Table 6 shows that the *Unit Selection Algorithm* has performed very well. The average classification performance on the seismic images has increased from $78.66(13.84)\%$ using all 25 features and 25 hidden units to $81.92(7.97)\%$ whilst using on average $6.59(7.16)$ inputs and $1.83(0.99)$ hidden units.

It is interesting to note that for training set $T2$ and test set $L2$ the performances increase from just under 50.0% to over 80.0% by reducing the number of hidden units to 1 from 25 whilst maintaining all its inputs. This is an increase of 30.0%.

The *USA* does take longer to run than the *ISA* algorithm but this is to be expected as we are now selecting hidden units as well as inputs. However, it is still very computationally advantageous over the traditional techniques. The CPU time taken to reduce a [25, 25, 2] network to a [10, 10, 2] network is 450u compared with 2000u to get to the same stage with the *SBS* algorithm.

5 Conclusions

In this paper a new algorithm that reduces the network complexity was introduced and was successfully applied to the problem of searching through Seismic image datasets. The algorithm significantly reduced the network size such that the network had only ~ 20 free parameters instead of 702 (less than 3%). A corresponding increase in classification performance of unseen patterns was observed using the reduced network size.

Also, reducing the network size also dramatically reduces the amount of time required to classify unseen patterns. This is particularly important in our application of classifying seismic datasets as these are typically of the order of tens of gigabytes in size.

References

[1] C Bishop. *Neural networks for pattern recognition*. Clarendon Press, Oxford, UK, 1996.

[2] S Fahlman and C Lebiere. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, 1990.

[3] B Hassibi and D Stork. Second order derivatives for pruning: optimal brain surgeon. In *Advances in Neural Information Processing Systems*, volume 5, pages 164–171, 1993.

[4] S Haykin. *Neural networks: A comprehensive foundation*. Macmillian College Publishing Company, New York, 1994.

[5] Y leCun, J Denker, and D Henderson. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, 1990.

[6] J Mao and A Jain. Regularization techniques in artificial neural networks. In *Proceedings of World Congress on Neural Networks*, pages IV75–79, 1993.

[7] J Mao, K Mohiuddin, and A Jain. Parsimonious network design and feature selection through node pruning. In *12th International Conference on Pattern Recognition*, pages 622–624, 1994.

[8] K. Messer, J. Kittler, and M. Kraaijveld. Selecting features for neural networks to aid an iconic search through an image database. In *IEE 6th International Conference on Image Processing and Its Applications*, pages 428–432, 1997.

[9] M Mozer and P Smolensky. Skeletonization: a technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems*, volume 1, pages 107–115, 1989.

[10] P Pudil, J Novovicova, and J Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.

[11] R Reed. Pruning algorithms - a survey. *IEEE Transactions on Neural Netwroks*, 4(5):740–747, September 1991.

[12] D Ruck, S Rogers, and M Kabrisky. Feature selection using a multilayer perceptron. *Neural Network Computation*, 20:40–48, 1990.

[13] R Setiono. A penalty function approach for pruning feed-forward neural networks. *Neural Computation*, 9:185–204, 1997.

[14] R Setiono and H Liu. Neural network feature selector. *IEEE Transactions on Neural Networks*, 8(3):654–661, May 1997.