TRAINING ARTIFICIAL NEURAL NETWORKS USING TAGUCHI METHODS

Geva Dror, Chris MacLeod, Grant Maxwell

The Robert Gordon University, Aberdeen

Abstract

This paper shows how the process optimization methods known as Taguchi methods may be applied to the training of Artificial Neural Networks. A comparison is made between the efficiency of training using Taguchi methods and the efficiency of conventional training methods; attention is drawn to the advantages of Taguchi methods. Further, it is shown that Taguchi methods offer potential benefits in evaluating network behaviour such as the ability to examine interaction of weights and neurons within a network.

Introduction

Taguchi methods were developed as a process optimization technique by Dr Genuchi Taguchi during the 1950's [1]. The methods are based on the statistical analysis of data and offer a simple means of analysis and optimization of complex systems.

Although the methods are normally used in process engineering, they are equally applicable to other systems [2]. One system to which they lend themselves is the training of Neural Networks.

A detailed description of the operation of Taguchi methods is beyond the scope of this paper, but a simple example is given below to illustrate their application in a process environment. A further example is then developed to show how this technique can be applied to the training of Neural Networks.

Application of Taguchi Methods in a Process Environment

Consider the process of mixing concrete; we have a choice of different mixtures of sand, cement and water and we do not know which to choose. We decide to try two different levels of each, as listed below:

C1 = 1Kg of cement
C2 = 1.5Kg of cement

S1 = 500g of sand
S2 = 750g of sand

W1 = 1 litre of water
W2 = 2 litres of water

We can try every combination of sand, cement and water and test each different combination to see which is the hardest. If we do this, there will be a total of eight combinations.

Taguchi experiments reduce the number of experiments required to find the best levels for each factor. The method works by calculating the statistical properties of orthogonal arrays.

We can draw up a table for the cement mixing example, with 3 factors (cement, sand and water)

and 2 levels (for each) in an orthogonal array.

Trial Number	Factors C	S	W
Y1	1	1	1
Y2	1	2	2
Y3	2	1	2
Y4	2	2	1

where 1 and 2 are the levels of each factor. For example, in trial 1, we make our mixture with all the ingredients at level 1.

A full set of experiments for this process would require eight different experiments (= 2^3) as opposed to the four which are needed for the Taguchi version of the experiment. The saving involved in using the Taguchi method becomes more significant as the number of levels or factors increases [1].

To analyze the results, we must have a way of finding which experiment produced the best answer. In our example, we would have to measure the hardness of the cement. Assume that a lower result indicates a harder cement. (In Neural Network terms, we would find the error associated with each experiment. The lower the error is, the better the result.)

So having undertaken the experiments and obtained the results, we can now calculate the best levels to use with each factor. Let us assume, for example, that the results obtained are as shown below:

experiment number	result (hardness)
Y1	11
Y2	20
Y3	5
Y4	7

We can find the effect of each level in each factor by averaging the results which contain that level and that factor.

C1 = (Y1 + Y2) / 2 = (11+20) / 2 = 15.5
C2 = (Y3 + Y4) / 2 = (5+7) / 2 = 6

S1 = (Y1 + Y3) / 2 = (11+5) / 2 = 8
S2 = (Y2 + Y4) / 2 = (20+7) / 2 = 13.5

W1 = (Y1 + Y4) / 2 = (11+7) / 2 = 9
W2 = (Y2 + Y3) / 2 = (20+5) / 2 = 12.5

From the above we can see that the best combination of factors is W2, S1, W1. These are the factors which produce the lowest results and hence the hardest mixture.

If we were to try a larger experiment with more factors (sand, cement, water and hardener) and several different levels of each factor, the number of trials required to exhaustively test all combinations would become very large. In general, the total number of experiments would be:

$N = L^m$

where N = total number of experiments

L = number of levels for each factor
m = number of factors.

If, for example we had an experiment requiring 21 factors at 4 different levels, we would require 4^{21} experiments (= 4.3×10^{12}) to investigate all the possible combinations. Using Taguchi methods, however, we only require 64 experiments. Such an array is called a L64 array for this reason.

Application of Taguchi methods to the training of artificial neural networks

By making the elements of the orthogonal array the weights of the neural network, we can run the Taguchi method and find the optimum combination of weights for our network.

To illustrate this process, let us again take a very simple example. Consider a network with three weights A, B and C. This network is shown in figure 1.

If we decide to take the levels of the weights as 0.2 (level 1) and 0.8 (level 2), we can encode this system into an orthogonal array as shown below.

trial	A	B	C
Y1	1 (0.2)	1 (0.2)	1 (0.2)
Y2	1 (0.2)	2 (0.8)	2 (0.8)
Y3	2 (0.8)	1 (0.2)	2 (0.8)
Y4	2 (0.8)	2 (0.8)	1 (0.2)

In this system there are four trials. We can apply an input vector and get some results [3], which are the network errors as shown below.

trial	Result (error)
Y1	1.2754
Y2	1.4355
Y3	0.9826
Y4	1.1095

The method of measuring the error is not important in this example. We want to select the weights which will result in the lowest error [1].

The error results are analysed in the same way as in the previous example.

A1 = (Y1 + Y2)/2 = (1.2754 + 1.4255) / 2 = 1.3555
A2 = (Y3 + Y4)/2 = (0.9826 + 1.1095) / 2 = 1.0461

B1 = (Y1 + Y3)/2 = (1.2754 + 0.9826) / 2 = 1.1290
B2 = (Y2 + Y4)/2 = (1.4355 + 1.1095) / 2 = 1.2725
C1 = (Y1 + Y4)/2 = (1.2754 + 1.1095) / 2 = 1.1925
C2 = (Y2 + Y3)/2 = (1.4355 + 0.9826) / 2 = 1.2091

It may be clearly seen that the optimum combination of weights in this case is A2, B1 and C1.

This is a very simple example to illustrate the application of the method. However, it is possible to scale this up to any size of network and a large number of levels.

A L64 array with four levels, as mentioned previously, is shown in figure 2. This is capable of

handling a network of 21 weights. The analysis of the results follows the same procedure as illustrated above.

Results

To test the application of this method, a L64, four level matrix was set up. This matrix is shown in figure 2.

The network was tested on a simple character recognition problem. Sets of simple shapes were set up on 3 by 3 pixel grids; three of these are shown in figure 3 [4].

These were applied to the network and an output vector obtained. The network weights were then changed to the optimum values as demonstrated in the section above. The network may be either fine tuned or trained on a second character by using the optimized weights obtained from the first pass as the starting point from which to continue training in the second pass.

In a simple experiment the network was operated to recognise any one of three simple characters. The network had three output neurons each trained to give a low error corresponding to a different character input. The actual errors are shown below.

Neuron No	Character number. A	B	C
1	**0.05**	0.4	0.35
2	0.56	**0.13**	0.87
3	0.80	0.67	**0.21**

We can see from the results above that neuron 1 produces the lowest error with character number 1 which is the correct answer and likewise neurons 2 and 3 give the best results with the characters they are trained for.

On average in the first pass of the method the error reduction was 37.32%. When comparing this reduction with a back-propagation algorithm running on the same size of network (using the matlab neural networks toolbox), it was found that the Taguchi method was around 10 times faster [5].

Although this is a fairly small and simple system, the network is scalable to any size [1].

Other applications of Taguchi methods within neural networks

Taguchi methods also have other applications within neural networks. Two of the most important are as preprocessing for other algorithms and to examine interrelated dependency and interactions between different parts of a network. These are discussed below.

1. *Preprocessing*

By using a multi-level Taguchi matrix, it is possible to fine tune the weights of a network and reduce the error. However, the Taguchi method does not allow you to move the weights in the small increments which other training algorithms do. Its advantage is that it is a fast method of reducing error. It is possible to use a one shot application of the Taguchi method to reduce errors and then apply another fine tuning training method such as back-propagation or Boltzmann to reduce the error even further by producing small changes in the weight matrix [6].

2. *Interaction*

The Taguchi method allows users to analyse the system and calculate the interactions between different components (for example, weights or neurons). These ideas are called analysis of interaction and analysis of variance [1]. This would enable the investigator to calculate what would be the effect on the network as a whole of changing a single weight, or even to calculate the effect of changing one weight on another. This provides an effective means of examining the inner operation of the network through statistics. This is a facility which it is difficult to provide through any other method. The manner in which these techniques is applied can be found in any good text on the Taguchi method [1,2].

Conclusions

It has been demonstrated above that the Taguchi method is an effective way of training feed forward neural networks. In some circumstances, it is faster than other algorithms and in a general character recognition problem it is up to 10 times faster than a back- propagation algorithm.

The application of the Taguchi method is particularly beneficial as a means of performing 'coarse', rapid training as a first pass. This can then be followed by the application of another training algorithm to fine-tune the weights.

A further advantage of the Taguchi method is that information can be made available about the interactions between weights and neurons which is not readily available though other methods.

References

1. Ranjit, K R (1990): A primer on the Taguchi method: Van Nostrand Reinhold.

2. Lochner, H R et al (1990): Designing for quality. An introduction to the best of Taguchi and western methods of statistical method design: Chapman - Hall.

3. Hilton, G E (1992): How neural networks learn from experience: Scientific American Vol 267 No 3.

4. Wasserman, P D (1989): Neural Computing, theory and practice: Van Nostrand Reinhold.

5. Anon :The matlab neural networks toolbox manual: Mathworks.

6. McShane, J (1992): An introduction to neural nets. Hewlett-packard Journal Vol 43 No 1.