Neural Networks for Prediction and Classification

Kate A. Smith
Monash University, Australia

INTRODUCTION

Neural networks are simple computational tools for examining data and developing models that help to identify interesting patterns or structures. The data used to develop these models is known as training data. Once a neural network has been exposed to the training data, and has learnt the patterns that exist in that data, it can be applied to new data thereby achieving a variety of outcomes. Neural networks can be used to:

- learn to *predict* future events based on the patterns that have been observed in the historical training data;
- learn to *classify* unseen data into pre-defined groups based on characteristics observed in the training data;
- learn to *cluster* the training data into natural groups based on the similarity of characteristics in the training data.

BACKGROUND

There are many different neural network models that have been developed over the last fifty years or so to achieve these tasks of prediction, classification, and clustering. Broadly speaking, these models can be grouped according to supervised learning algorithms (for prediction and classification), and unsupervised learning algorithms (for clustering). This paper focused on the former paradigm. We refer the interested reader to Haykin (1994) for a detailed account of many neural network models.

According to a recent study (Wong, Jiang, & Lam, 2000), over fifty percent of reported neural network business application studies utilise multilayered feedforward neural networks (MFNNs) with the backpropagation learning rule (Werbos, 1974; Rumelhart & McClelland, 1986). This type of neural network is popular because of its broad applicability to many problem domains of relevance to business and industry: principally prediction, classification, and modeling. MFNNs are appropriate for solving problems that involve learning the relationships between a set of inputs and known outputs. They are a supervised learning technique in the sense that they require a set of training data in order to learn the relationships. With supervised learning models, the training data contains matched information about input variables and observable outcomes. Models can be developed that learn the relationship between these data characteristics (inputs) and outcomes (outputs). Figure 1 shows the architecture of a 3-layer MFNN. The weights are updated using the backpropagation learning algorithm, as summarized in Table 1, where *f(.)* is a non-linear activation function such as $1/(1+exp(-\lambda(.)))$, λ is the learning rate, and d_k is the desired output of the k^{th} neuron.

The successful application of MFNNs to a wide range of areas of business, industry and society has been reported widely. We refer the interested reader to the excellent survey articles covering application domains of medicine (Weinstein, Myers, Casciari, Buolamwini, & Raghavan, 1994), communications (Ibnkahla, 2000), business (Smith & Gupta, 2002), finance (Refenes, 1995), and a variety of industrial applications (Fogelman, Soulié, & Gallinari, 1998).

MAIN THRUST

A number of significant issues will be discussed, and some guidelines for successful training of neural networks will be presented in this section.

Figure 1. Architecture of MFNN (note: not all weights are shown)

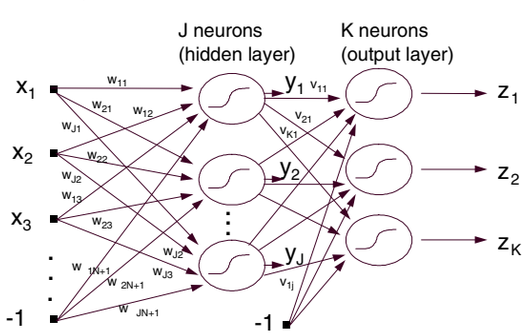

Table 1. Backpropagation learning algorithm for MFNNs

STEP 1:	Randomly select an input pattern **x** to present to the MFNN through the input layer
STEP 2:	Calculate the net inputs and outputs of the hidden layer neurons $net_j^h = \sum_{i=1}^{N+1} w_{ji} x_i \quad y_j = f(net_j^h)$
STEP 3:	Calculate the net inputs and outputs of the K output layer neurons $net_k^o = \sum_{j=1}^{J+1} v_{kj} y_j \quad z_k = f(net_k^o)$
STEP 4:	Update the weights in the output layer (for all k, j pairs) $v_{kj} \leftarrow v_{kj} + c\lambda(d_k - z_k) z_k (1 - z_k) y_j$
STEP 5:	Update the weights in the hidden layer (for all i, j pairs) $w_{ji} \leftarrow w_{ji} + c\lambda^2 y_j (1 - y_j) x_i (\sum_{k=1}^{K} (d_k - z_k) z_k (1 - z_k) v_{kj})$
STEP 6:	Update the error term $E \leftarrow E + \sum_{k=1}^{K} (d_k - z_k)^2$ and repeat from STEP 1 until all input patterns have been presented (one epoch).
STEP 7:	If E is below some predefined tolerance level (say 0.000001), then STOP. Otherwise, reset $E = 0$, and repeat from STEP 1 for another epoch.

Critical Issues for Neural Networks

Neural networks have not been without criticism, and there are a number of important limitations that need careful attention. At this stage it is still a trial and error process to obtain the *optimal model* to represent the relationships in a dataset. This requires appropriate selection of a number of parameters including learning rate, momentum rate (if an additional momentum term is added to steps 4 and 5 of the algorithm), choice of activation function, as well as the optimal neural architecture. If the architecture is too large, with too many hidden neurons, the MFNN will find it very easy to memorize the training data, but will not have generalized its knowledge to other datasets (out-of-sample or test sets). This problem is known as *overtraining*. Another limitation is the gradient descent nature of the backpropagation learning algorithm, which causes the network weights to become trapped in *local minima* of the error function being minimized. Finally, neural networks are considered inappropriate for certain problem domains where insight and *explanation* of the model is required. Some work has been done on extracting rules from trained neural networks (Andrews, Diederich, & Tickle, 1995), but other data mining techniques like rule induction may be better suited to such situations.

Guidelines for Successful Training

Successful prediction and classification with MFNNs requires careful attention to two main stages:

1. **Learning:** The developed model must represent an adequate fit of the training data. The data itself must contain the relationships that the neural network is trying to learn, and the neural network model must be able to derive the appropriate weights to represent these relationships.
2. **Generalization:** The developed model must also perform well when tested on new data to ensure that it has not simply memorized the training data characteristics. It is very easy for a neural network model to "overfit" the training data, especially for small data sets. The architecture needs to be kept small, and key validation procedures need to be adopted to ensure that the learning can be generalized.

Within each of these stages there are a number of important guidelines that can be adopted to ensure effective learning and successful generalization for prediction and classification problems (Remus & O'Connor, 2001).

To ensure successful learning:

- **Prepare the Data Prior to Learning the Neural Network Model:** A number of pre-processing steps may be necessary including cleansing the data; removing outliers; determining the correct level of summarisation; converting non-numeric data (Pyle, 1999).
- **Normalise, Scale, Deseasonalise and Detrend the Data Prior to Learning:** Time series data often needs to be deseasonalised and detrended to enable the neural network to learn the true patterns in the data (Zhang, Patuwo, & Hu, 1998).