An Overview of Artificial Neural Networks

Introduction

In recent years neural computing has emerged as a practical technology, with successful applications in many fields as diverse as finance, medicine, engineering, geology, physics and biology. The excitement stems from the fact that these networks are attempts to model the capabilities of the human brain. From a statistical perspective neural networks are interesting because of their potential use in prediction and classification problems.

Artificial neural networks (ANNs) are non-linear data driven self adaptive approach as opposed to the traditional model based methods. They are powerful tools for modelling, especially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. After training, ANNs can be used to predict the outcome of new independent input data. ANNs imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. Thus they are ideally suited for the modeling of agricultural data which are known to be complex and often non-linear.

A very important feature of these networks is their adaptive nature, where “learning by example” replaces “programming” in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available.

These networks are “neural” in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. In fact majority of the network are more closely related to traditional mathematical and/or statistical models such as non-parametric pattern classifiers, clustering algorithms, nonlinear filters, and statistical regression models than they are to neurobiology models.

Neural networks (NNs) have been used for a wide variety of applications where statistical methods are traditionally employed. They have been used in classification problems, such as identifying underwater sonar currents, recognizing speech, and predicting the secondary structure of globular proteins. In time-series applications, NNs have been used in predicting stock market performance. As statisticians or users of statistics, these problems are normally solved through classical statistical methods, such as discriminant analysis, logistic regression, Bayes analysis, multiple regression, and ARIMA time-series models. It is, therefore, time to recognize neural networks as a powerful tool for data analysis.

Characteristics of neural networks

- The NNs exhibit mapping capabilities, that is, they can map input patterns to their associated output patterns.
- The NNs learn by examples. Thus, NN architectures can be ‘trained’ with known examples of a problem before they are tested for their ‘inference’ capability on

unknown instances of the problem. They can, therefore, identify new objects previously untrained.

- The NNs possess the capability to generalize. Thus, they can predict new outcomes from past trends.
- The NNs are robust systems and are fault tolerant. They can, therefore, recall full patterns from incomplete, partial or noisy patterns.
- The NNs can process information in parallel, at high speed, and in a distributed manner.

Basics of artificial neural networks

The terminology of artificial neural networks has developed from a biological model of the brain. A neural network consists of a set of connected cells: The neurons. The neurons receive impulses from either input cells or other neurons and perform some kind of transformation of the input and transmit the outcome to other neurons or to output cells. The neural networks are built from layers of neurons connected so that one layer receives input from the preceding layer of neurons and passes the output on to the subsequent layer.

A neuron is a real function of the input vector$(y_1, \ldots, y_k)$. The output is obtained as $f(x_j) = f\left(\alpha_j + \sum_{i=1}^{k} w_{ij} y_j\right)$, where f is a function, typically the sigmoid (logistic or tangent hyperbolic) function. A graphical presentation of neuron is given in Figure 1. Mathematically a Multi-Layer Perceptron network is a function consisting of compositions of weighted sums of the functions corresponding to the neurons.

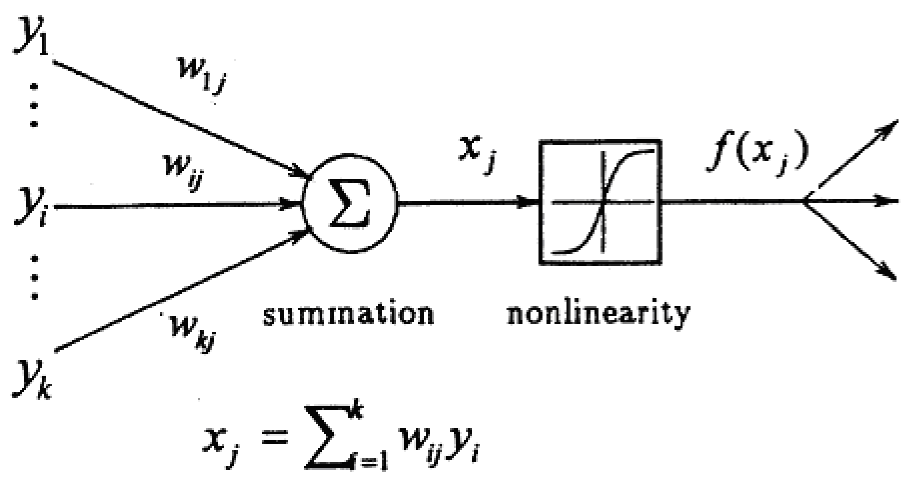

Figure 1: A single neuron

Neural networks architectures

An ANN is defined as a data processing system consisting of a large number of simple highly inter connected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain. There are several types of architecture of NNs. However, the two most widely used NNs are discussed below:

Feed forward networks

In a feed forward network, information flows in one direction along connecting pathways, from the input layer via the hidden layers to the final output layer. There is no feedback (loops) i.e., the output of any layer does not affect that same or preceding layer.

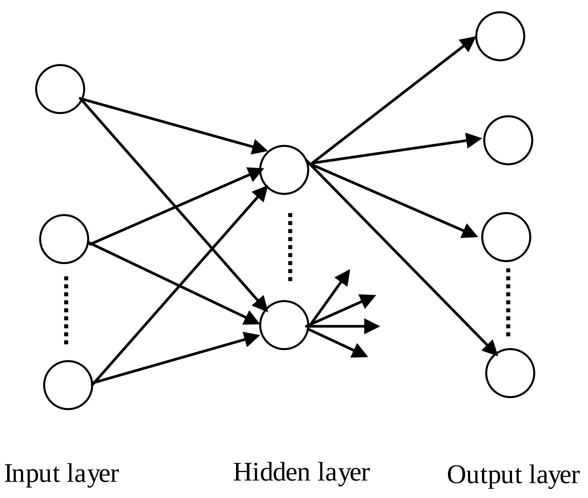

Figure 2: A multi-layer feed forward neural network

Recurrent networks

These networks differ from feed forward network architectures in the sense that there is at least one feedback loop. Thus, in these networks, for example, there could exist one layer with feedback connections as shown in figure below. There could also be neurons with self-feedback links, i.e. the output of a neuron is fed back into itself as input.

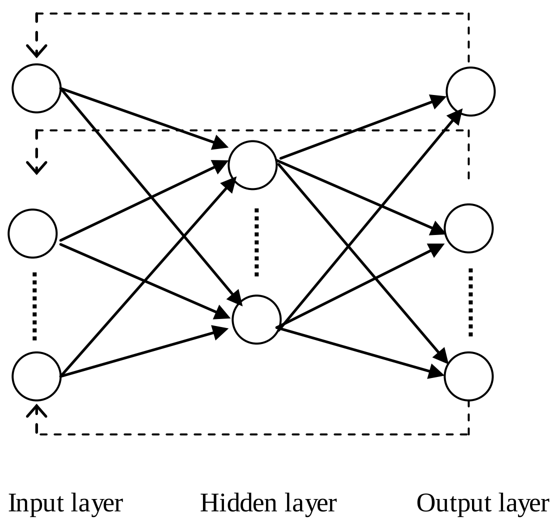

Figure 3: A recurrent neural network

Learning/Training methods

Learning methods in neural networks can be broadly classified into three basic types: supervised, unsupervised and reinforced.

Supervised learning

In this, every input pattern that is used to train the network is associated with an output pattern, which is the target or the desired pattern. A teacher is assumed to be present during the learning process, when a comparison is made between the network's computed output and the correct expected output, to determine the error. The error can then be used to change network parameters, which result in an improvement in performance.

Unsupervised learning

In this learning method, the target output is not presented to the network. It is as if there is no teacher to present the desired patterns and hence, the system learns of its own by discovering and adapting to structural features in the input patterns.

Reinforced learning

In this method, a teacher though available, does not present the expected answer but only indicates if the computed output is correct or incorrect. The information provided helps the network in its learning process. A reward is given for a correct answer computed and a penalty for a wrong answer. But, reinforced learning is not one of the popular forms of learning.

Types of neural networks

The most important class of neural networks for real world problems solving includes

- Multilayer Perceptron
- Radial Basis Function Networks
- Kohonen Self Organizing Feature Maps

Multilayer Perceptrons

The most popular form of neural network architecture is the multilayer perceptron (MLP). A multilayer perceptron:

- has any number of inputs.
- has one or more hidden layers with any number of units.
- uses linear combination functions in the input layers.
- uses generally sigmoid activation functions in the hidden layers.
- has any number of outputs with any activation function.
- has connections between the input layer and the first hidden layer, between the hidden layers, and between the last hidden layer and the output layer.

Given enough data, enough hidden units, and enough training time, an MLP with just one hidden layer can learn to approximate virtually any function to any degree of accuracy. (A statistical analogy is approximating a function with *n*th order polynomials.) For this reason MLPs are known as universal approximators and can be used when you have little prior knowledge of the relationship between inputs and targets. Although one hidden layer is always sufficient provided you have enough data, there are situations where a network with two or more hidden layers may require fewer hidden units and weights than a network with one hidden layer, so using extra hidden layers sometimes can improve generalization.

Radial Basis Function Networks

Radial basis functions (RBF) networks are also feedforward, but have only *one* hidden layer. A RBF network:

- has any number of inputs.
- typically has only one hidden layer with any number of units.
- uses radial combination functions in the hidden layer, based on the squared Euclidean distance between the input vector and the weight vector.
- typically uses exponential or softmax activation functions in the hidden layer, in which case the network is a Gaussian RBF network.
- has any number of outputs with any activation function.
- has connections between the input layer and the hidden layer, and between the hidden layer and the output layer.

MLPs are said to be distributed-processing networks because the effect of a hidden unit can be distributed over the entire input space. On the other hand, Gaussian RBF networks are said to be local-processing networks because the effect of a hidden unit is usually concentrated in a local area centered at the weight vector.

Kohonen Neural Network

Self Organizing Feature Map (SOFM, or Kohonen) networks are used quite differently to the other networks. Whereas all the other networks are designed for supervised learning tasks, SOFM networks are designed primarily for unsupervised learning (Patterson, 1996).

The principal goal of the SOFM is to transform an incoming signal pattern of arbitrary dimension into a one-or two dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion. Figure 4 shows the schematic diagram of a two-dimensional lattice of neurons commonly used as the discrete map. Each neuron in the lattice is fully connected to all the source nodes in the input layer. This network represents a feedforward structure with a single computational layer consisting of neurons arranged in rows and columns. A one-dimensional lattice is a special case of the configuration depicted in Fig. 4: in this special case the computational layer consists simply of a single column or row of neurons.

Each input pattern presented to the network typically consists of a localized region or "spot" of activity against a quiet background. The location and nature of such a spot usually varies from one realization of the input pattern to another. All the neurons in the network should therefore be exposed to a sufficient number of different realizations of the input pattern to ensure that the self-organization process has a chance to mature properly.

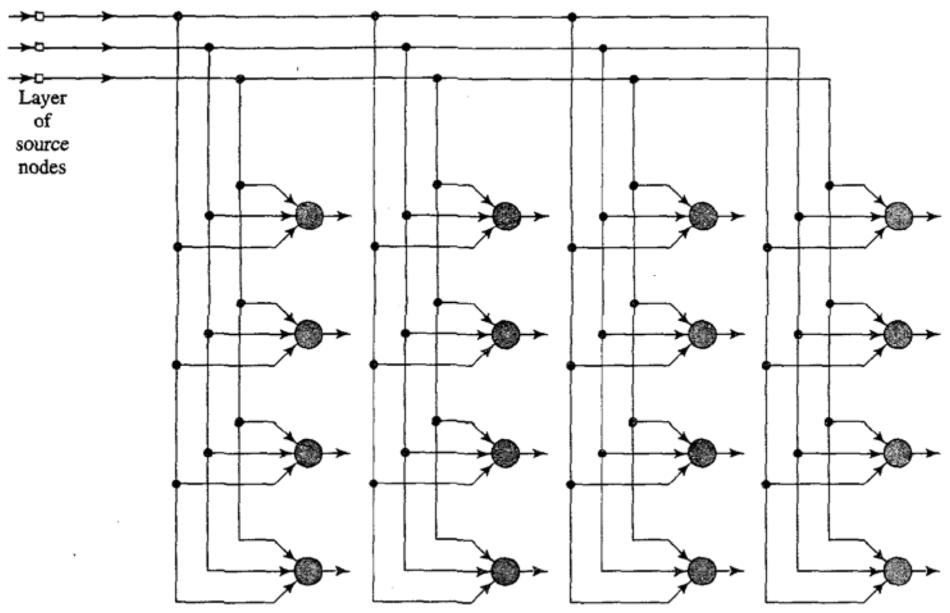

Fig. 4: Two-dimensional lattice of neurons

The algorithm responsible for the formation of the self-organizing map proceeds first by initialing the synaptic weights in the network. This can be done by assigning them small values picked from a random number generator, in so doing, no prior order is imposed on the feature map. Once the network has been properly initialized, there are three essential processes involved in the formation of the self-organizing map, as summarized here:

- *Competition*: For each input pattern, the neurons in the network compute their respective values of a discriminant function. This discriminant function provides the basis for competition among the neurons. The particular neuron with the largest value of discriminant function is declared winner of the competition.
- *Cooperation*: The winning neuron determines the spatial location of a topological neighborhood of excited neurons, thereby providing the basis for cooperation among such neighboring neurons.
- *Synaptic Adaptation*: This last mechanism enables the excited neurons to increase their individual values of the discriminant function in relation to the input pattern through suitable adjustments applied to their synaptic weights. The adjustments made are such that the response of winning neuron to the subsequent application of similar input pattern in enhanced.

The processes of competition and cooperation are in accordance with two of the four principles of self-organization. As for the principle of self-amplification, it comes in a modified form of Hebbian learning in the adaptive process. The presence of redundancy in the input data is needed for learning since it provides knowledge.

One possible use of SOFM is in exploratory data analysis. A second possible use is in novelty detection. SOFM networks can learn to recognize clusters in the training data, and respond to it. If new data, unlike previous cases, is encountered, the network fails to recognize it and this indicates novelty.

Applications

Neural networks have proven to be effective mapping tools for a wide variety of problems and consequently they have been used extensively by practitioners in almost every application domain ranging from agriculture to zoology. Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting applications. A very small sample of applications has been indicated in the next paragraph.

Kaastra and Boyd (1996) developed neural network model for forecasting financial and economic time series. Kohzadi *et al.* (1996) used a feedforward neural network to compare ARIMA and neural network price forecasting performance. Dewolf *et al.* (1997, 2000) demonstrated the applicability of neural network technology for plant diseases forecasting. Zhang *et al.* (1998) provided the general summary of the work in ANN forecasting, providing the guidelines for neural network modeling, general paradigm of the ANNs especially those used for forecasting, modeling issue of ANNs in forecasting and relative performance of ANN over traditional statistical methods. Sanzogni *et al.* (2001) developed the models for predicting milk production from farm inputs using standard feed forward ANN. Kumar *et al.* (2002) studied utility of neural networks for estimation of daily grass reference crop evapotranspiration and compared the performance of ANNs with the conventional method used to estimate evapotranspiration. Pal *et al.* (2002) developed MLP based forecasting model for maximum and minimum temperatures for ground level at Dum Dum station, Kolkata on the basis of daily data on several variables, such as mean sea level pressure, vapour pressure, relative humidity, rainfall, and radiation for the period 1989-95. Gaudart *et al.* (2004) compared the performance of MLP and that of linear regression for epidemiological data with regard to quality of prediction and robustness to deviation from underlying assumptions of normality, homoscedasticity and independence of errors. The details of some of the application in the field of agriculture will be discussed in the class. In the context of economic data, Swanson and White (1997) investigate the performance of neural network models in forecasting nine quarterly seasonality adjusted US macro econometric time series, finding that they generally outperform traditional economic approaches even where there is no explicit non-linearity. Heravi *et al.* (2004) consider linear and ANN models for forecasting seasonally unadjusted monthly data on European industrial production series and conclude that linear models generally produce more accurate post-sample forecast than neural network models at horizons of up to a year, in terms of root mean square error.

The large number of parameters that must be selected to develop a neural network model for any application indicates that the design process still involves much trial and error. The next section provides a practical introductory guide for designing a neural network model.

Development of an ANN model

The various steps in developing a neural network model are:

A. Variable selection

The input variables important for modeling variable(s) under study are selected by suitable variable selection procedures.

B. Formation of training, testing and validation sets

The data set is divided into three distinct sets called training, testing and validation sets. The training set is the largest set and is used by neural network to learn patterns present in the data. The testing set is used to evaluate the generalization ability of a supposedly trained network. A final check on the performance of the trained network is made using validation set.

C. Neural network architecture

Neural network architecture defines its structure including number of hidden layers, number of hidden nodes and number of output nodes etc.

- Number of hidden layers: The hidden layer(s) provide the network with its ability to generalize. In theory, a neural network with one hidden layer with a sufficient number of hidden neurons is capable of approximating any continuous function. In practice, neural network with one and occasionally two hidden layers are widely used and have to perform very well.
- Number of hidden nodes: There is no magic formula for selecting the optimum number of hidden neurons. However, some thumb rules are available for calculating number of hidden neurons. A rough approximation can be obtained by the geometric pyramid rule proposed by Masters (1993). For a three layer network with n input and m output neurons, the hidden layer would have *sqrt(n*m)* neurons.
- Number of output nodes: Neural networks with multiple outputs, especially if these outputs are widely spaced, will produce inferior results as compared to a network with a single output.
- Activation function: Activation functions are mathematical formulae that determine the output of a processing node. Each unit takes its net input and applies an activation function to it. Non linear functions have been used as activation functions such as logistic, tanh etc. The purpose of the transfer function is to prevent output from reaching very large value which can 'paralyze' neural networks and thereby inhibit training. Transfer functions such as sigmoid are commonly used because they are nonlinear and continuously differentiable which are desirable for network learning.

D. Evaluation criteria

The most common error function minimized in neural networks is the sum of squared errors. Other error functions offered by different software include least absolute deviations, least fourth powers, asymmetric least squares and percentage differences.

E. Neural network training

Training a neural network to learn patterns in the data involves iteratively presenting it with examples of the correct known answers. The objective of training is to find the set of weights between the neurons that determine the global minimum of error function. This involves decision regarding the number of iteration i.e., when to stop training a neural network and the selection of learning rate (a constant of proportionality which determines the size of the weight adjustments made at each iteration) and momentum values (how past weight changes affect current weight changes).

Conclusion

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. A large number of claims have been made about the modeling capabilities of neural networks, some exaggerated and some justified. Hence, to best utilize ANNs for different problems, it is essential to understand the potential as well as limitations of neural networks. For some tasks, neural networks will never replace conventional methods, but for a growing list of applications, the neural architecture will provide either an alternative or a complement to these existing techniques. Finally, I would like to state that even though neural networks have a huge potential we will only get the best of them when they are integrated with Artificial Intelligence, Fuzzy Logic, Particle Swarm Optimization and related subjects.

Practical Exercise

The details of construction and training of a feed forward neural network will be illustrated through a simple logical problem known as exclusive OR (XOR) problem using Neural Network Toolbox of Matlab in the class. Time series forecasting application of neural network will be demonstrated using famous data of annual sunspot numbers.

References

Cheng, B. and Titterington, D. M. (1994). Neural networks: A review from a statistical perspective. Statistical Science, **9**, 2-54.

Dewolf, E.D., and Francl, L.J., (1997). Neural networks that distinguish in period of wheat tan spot in an outdoor environment. *Phytopathalogy*, **87,** 83-87.

Dewolf, E.D. and Francl, L.J. (2000) Neural network classification of tan spot and stagonespore blotch infection period in wheat field environment. *Phytopathalogy*, **20**, 108-113 .

Gaudart, J. Giusiano, B. and Huiart, L. (2004). Comparison of the performance of multi-layer perceptron and linear regression for epidemiological data. *Comput. Statist. & Data Anal.*, **44,** 547-70.

Girish K. Jha, Parimala Thulasiraman and Ruppa K. Thulasiram (2009). PSO based neural network for time series forecasting. In proceeding of the IEEE International Joint Conference on Neural Networks, USA, pp 1422-1427.

Hervai, S., Osborn, D. R., and Birchenhall. C. R. (2004). Linear versus neural network forecast for European industrial production series. *International Journal of Forecasting*, 20, 435-446.

Kaastra, I. and Boyd, M.(1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, **10**, 215-236.

Kohzadi, N., Boyd, S.M., Kermanshahi, B. and Kaastra, I. (1996). A comparision of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, **10**, 169-181.

Kumar, M., Raghuwanshi, N. S., Singh, R,. Wallender, W. W. and Pruitt, W. O. (2002). Estimating evapotranspiration using artificial neural network. *Journal of Irrigation and Drainage Engineering*, **128**, 224-233

Pal, S. Das, J. Sengupta, P. and Banerjee, S. K. (2002). Short term prediction of atmospheric temperature using neural networks. *Mausam*, **53,** 471-80

Patterson, D. (1996). Artificial Neural Networks. Singapore: Prentice Hall.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage ang organization in the brain. *Psychological review*, **65,** 386-408.

Rumelhart, D.E., Hinton, G.E and Williams, R.J. (1986). "Learning internal representation by error propagation", in Parallel distributed processing: Exploration in microstructure of cognition, Vol. (1) *(D.E. Rumelhart, J.L. McClelland and the PDP research gropus, edn.) Cambridge, MA: MIT Press*, 318-362.

Simon Haykin (2006). Neural Networks: A comprehensive foundation, Pearson Prentice Hall.

Saanzogni, Louis and Kerr, Don (2001) Milk production estimate using feed forward artificial neural networks. *Computer and Electronics in Agriculture*, **32**, 21-30.

Swanson, N. R., and White, H. (1997). Forecasting economic time series using adaptive versus non-adaptive and linear versus nonlinear economic models. *International Journal of Forecasting*, 13, 439–461.

Warner, B. and Misra, M. (1996). Understanding neural networks as statistical tools. *American Statistician*, **50**, 284-93.

Zhang, G., Patuwo, B. E. and Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, **14**, 35-62.

Zhang, G.P. (2007). Avoiding pitfalls in neural network research. IEEE transactions on systems, man and cybernetics-Part C: Applications and reviews, **37**, 3-16.