Artificial Neural Networks: tutorial

Chapter · January 2015

CITATIONS
0

READS
2,548

1 author:

Crescenzio Gallo
Università degli studi di Foggia
115 PUBLICATIONS **190** CITATIONS

SEE PROFILE

Encyclopedia of Information Science and Technology, Third Edition

Mehdi Khosrow-Pour
Information Resources Management Association, USA

A volume in the

Information Science REFERENCE
An Imprint of IGI Global

Artificial Neural Networks Tutorial

N

Crescenzio Gallo
Department of Clinical and Experimental Medicine, University of Foggia, Italy

INTRODUCTION

The purpose of this chapter is to introduce a powerful class of mathematical models: the artificial neural networks. This is a very general term that includes many different systems and various types of approaches, both from statistics and computer science. Our aim is not to examine them all (it would be a very long discussion), but to understand the basic functionality and the possible implementations of this powerful tool. We initially introduce neural networks, by analogy with the human brain. The analogy is not very detailed, but it serves to introduce the concept of parallel and distributed computing. Then we analyze in detail a widely applied type of artificial neural network: the feed-forward network with error back-propagation algorithm. We illustrate the architecture of the models, the main learning methods and data representation, showing how to build a typical artificial neural network.

BACKGROUND

Artificial neural networks (Bandy, 1997; Haykin, 1999) are information processing structures providing the (often unknown) connection between input and output data (Honkela, Duch, & Girolami, 2011) by artificially simulating the physiological structure and functioning of human brain structures.

The natural neural network, instead, consists of a very large number of nerve cells (about ten billion in humans), said neurons, and linked together in a complex network. The intelligent behavior is the result of extensive interaction between interconnected units. The input of a neuron is composed of the output signals of the neurons connected to it. When the contribution of these inputs exceeds a certain threshold, the neuron - through a suitable transfer function - generates a bioelectric signal, which propagates through the synaptic weights to other neurons.

DOI: 10.4018/978-1-4666-5888-2.ch626

Significant features of this network, which artificial neural models intend to simulate, are:

- The parallel processing, due to the fact that neurons process simultaneously the information;
- The twofold function of the neuron, that acts simultaneously as memory and signal processor;
- The distributed nature of the data representation, i.e. knowledge is distributed throughout the network, not circumscribed or predetermined;
- The network's ability to learn from experience.

This last but fundamental capacity enables neural networks to self-organize, adapt to new incoming information and extract the input-output connections from known examples that are the basis of their organization. An artificial neural network captures this attitude in an appropriate "learning" stage.

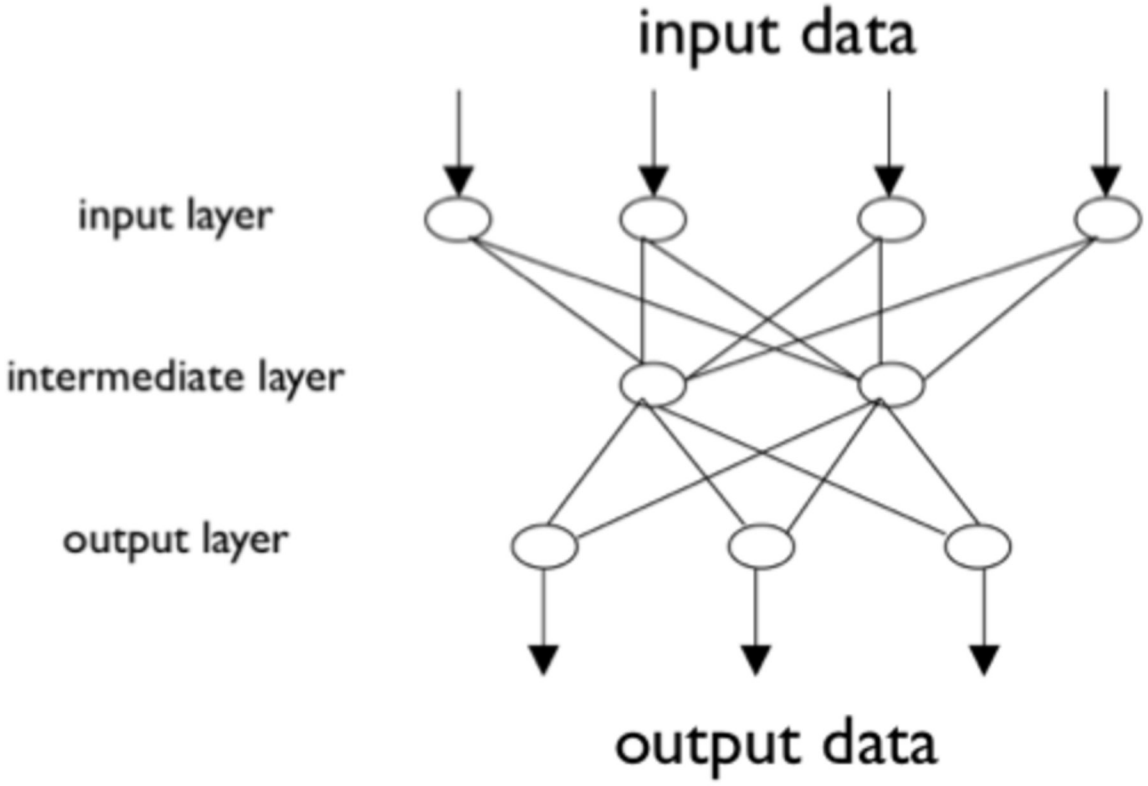
Despite the great success achieved by artificial neural networks, it is however better to remain aware of the limits of this technology due to the necessary reduction of the real system to be examined.

STRUCTURE OF A NEURAL NETWORK

Artificial neural networks are composed of elementary computational units called neurons (McCulloch & Pitts, 1943) combined according to different architectures. For example, they can be arranged in layers (multi-layer network), or they may have a connection topology. Layered networks consist of:

- Input layer, made of n neurons (one for each network input);
- Hidden layer, composed of one or more hidden (or intermediate) layers consisting of m neurons;

Figure 1. A basic artificial neuron



- Output layer, consisting of p neurons (one for each network output).

The connection mode allows distinguishing between two types of architectures:

- The feedback architecture, with connections between neurons of the same or previous layer;
- The feedforward architecture (Hornik, Stinchcombe, & White, 1989), without feedback connections (signals go only to the next layer's neurons).

Each neuron (Figure 1) receives n input signals x_i (with connection weights w_i) which sum to an "activation" value y. A suitable transfer (or activation) function F transforms it into the output F(y).

The operational capability of a network (namely, his knowledge) is contained in the connection weights, which assume their values thanks to the training phase.

ARCHITECTURES AND MODELS OF NEURAL NETWORKS

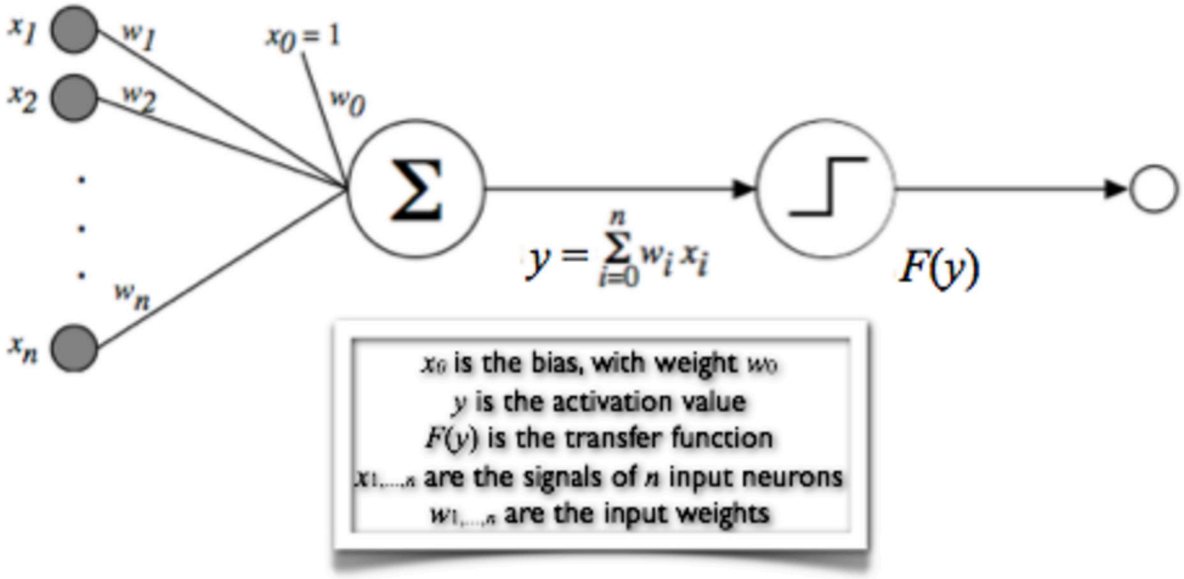
The different configuration possibilities are endless, so the choice of the optimal configuration must be primarily a function of the application target.

Perceptron

The simplest network (Rosenblatt, 1958; Minsky & Papert, 1969) is constituted by a single neuron, with n inputs and a single output. The basic learning algorithm of perceptron analyzes the input configuration (pattern) and — weighting variables through the synapses — decides which output is associated with the configuration.

This type of architecture presents the major limitation of being able to solve only linearly separable problems.

Figure 2. Structure of a feedforward multi-layer neural network



Multi Layer Perceptron (MLP) Networks

The neural network with an input layer, one or more intermediate layers of neurons and an output layer is called Multi Layer Perceptron or MLP (Hornik, Stinchcombe, & White, 1989). This network (see Figure 2) is of feed-forward type and uses, in most cases, the backpropagation learning algorithm. It calculates the weights between layers starting from random values and making small gradual and progressive changes after the network's output errors until the learning algorithm converges to an acceptable error approximation.

There are many other types of more complex structures, but these lie beyond our purposes. Indeed, the feedforward supervised backpropagation architecture is the most popular and is widely used for the capacity that this set of models has to generalize the results to a large number of problems.

Applications of Neural Networks

Neural network applications (Wong, Sumudu, Mendis, & Bouzerdoum, 2011) can be grouped into three major areas:

- Classification;
- Time series forecasting;
- Function approximation.

Function Approximation and Time Series Prediction

In the case of function approximation (or regression), networks are applied to all situations lacking a precise functional form describing input-output relations. The particular (useful) area of time series prediction aims to predict future value(s) through past available data periods.

Consider, for example, forecasting an exchange rate: it can only affect the exact value rather than the tendency of the period. In the first case the neural network's output is used for implementing the trading system. In the second case, it will suffice to know that a market is rising or falling to be able to compare it with the expected dynamics in other markets.

In any case, in each of the possible applications, from an operational viewpoint it is necessary to divide the series into two parts: one, made from the so-called in-sample observations, acts as a base for training (training set); the other, made from the out-of-sample observations, has the purpose to verify its validity (validation set). The network can be trained to provide a forecast for broader horizons, using its own short-term forecasts as input to long-term forecasts.

For the efficiency of this type of application the assessment of particular technical aspects is important, such as:

- **Choice of the Input Variables:** It must be made considering that the network is unable to provide any explanatory function, so it could use non-significant variables; in fact, the relationships between variables change with time and consequently significant input today may not be true in the future;
- **Optimal Learning Level:** It is necessary to take into account that a too short training process does not allow the network to capture the relationships between variables, while a too long training could make the network unable to generalize (overfitting);
- **Choice of Time Horizon for the Forecast:** This is an important factor in that very short forecast horizons increase the number of correct predictions; in contrast, indications of long forecasting time horizons are on average less correct, but the correct ones result in an higher average profit.

Classification and Discrimination

Neural networks can also be used to classify data (Haykin, 1999; Honkela, Duch, & Girolami, 2011). Unlike regression problems, where the goal is to produce the output corresponding to a given input, classification problems require to label each data point as belonging to one of n given classes. Neural networks can be trained to provide a discriminant function separating the classes. In a typical classification problem with two classes a straight line can be used as discriminating; the classes are so called linearly separable. These problems can be learned without hidden units but, occasionally, a nonlinear function is required to ensure the separation of the classes: this can only be solved by a neural network with one or more hidden layers.

Typical classification applications are, for example, credit ratings, trust decisions or biological risk assessment. In these cases the network has the task of assigning the input to a corresponding output among a number of predefined categories.

To use a neural network for classification, you need to build an equivalent function approximation problem by assigning a target value to each class. For a binary problem (two classes) you can use a neural network with a single output y, and a binary target value: 0 for one class, 1 for the other. You can therefore interpret the output of the network as an estimate of the probability that a given sample belongs to one of two classes. In these cases an activation function is often used which saturates the two target values, i.e. the logistic function:

$$f(x) = 1/(1+e^{-x}) \quad (1)$$

TRAINING AN ARTIFICIAL NEURAL NETWORK

The neural network is not programmed directly but it is explicitly trained through a learning algorithm for solving a given task, a process that leads to "learning through experience". The learning algorithm helps to define the specific configuration of a neural network and, therefore, conditions and determines the ability of the network itself to provide correct answers to specific problems.

We distinguish at least three types of learning: unsupervised, supervised and by reinforcement. In the first case, the network is trained only on the basis of a set of inputs, without providing the corresponding outputs. For supervised learning it is necessary to identify, instead, a set of examples consisting of appropriate inputs and corresponding outputs to be presented to the network so that it "learns" from them. Learning by reinforcement is used in cases where it is not possible to specify input-output patterns. Reinforcement is provided to the system, which interprets it as a positive/negative signal about its behavior and adjusts settings accordingly.

The dataset used for network learning constitutes the so-called learning or training set.

BACKPROPAGATION ALGORITHM

The backpropagation algorithm is used in supervised learning. It allows modifying connection weights in such a way that it minimizes a certain error function

E. This function depends on the i-th network's output (vector) net_i given the i-th input (vector) x_i and the i-th (vector of) desired output y_i. The training set is thus a set of N input/output pairs. The error function to be minimized can be written as:

$$E(w) = (1/2)\ \Sigma i_{\Sigma} k\ _{(}netik_{-y}ik_{)2} \qquad (2)$$

where index k represents the value corresponding to the k-th output neuron. E(w) is a function depending on the weights (which, in general, vary over time). To minimize it you can use the gradient-descent algorithm, which starts from a generic data point and calculates the gradient ∇, which gives the direction in which to move in order to reach the maximum increase (or decrease, if you consider $-\nabla$). Once defined the direction, you move of a preset η distance and find a new point on which to recalculate the gradient. The algorithm continues iteratively until the gradient is zero. The backpropagation algorithm can be divided into two steps.

- **Forward Step:** The input data to the network is propagated to the next level and so on. Then the network error E(w) is computed.
- **Backward Step:** The error made by the network is propagated backwards, and the weights are updated appropriately.

The logical steps for training a neural network with supervised learning are the following.

Create a set of input patterns and the associated set of (desired) output patterns.

Initialize the weights of the neural network to random values.

Learning cycle (this cycle ends only when the error is generally less than a predefined threshold or after a specified number of iterations).

1. Feedfoward phase (from the input to the output layer):
   a. extract an input pattern at random from those available;
   b. compute the value of all next neurons;
   c. subtract from the result the neuron's threshold activation value (if this value has not already been simulated with the addition of a fixed unit input value, the bias);
   d. filter the neuron's output applying a logistic function to make this value the input of the next neuron.
2. Compare the network result with the corresponding output pattern and derive the current network error.
3. Backpropagation phase (from the output to the input layer):
   a. calculate the correction to the weights according to the chosen minimum locating rule;
   b. apply the correction to the weights of the layer.

In order to have a good training, the set of input patterns must be completely examined (epoch). So, after randomly choosing a pattern, in the extraction of the next the old one should not be considered. In this way, at each step only inputs not yet processed participate to training, thus making a complete processing of all the training set.

BUILDING A NEURAL NETWORK

Network Parameters

The construction of a neural network (Bandy, 1997; Haykin, 1999) necessarily involves some steps requiring you to set the appropriate parameters:

Database Splitting

In order to get an optimal training and network implementation, it is necessary to divide the dataset into subsets, which determine the learning environment: training and validation set. The latter is in turn divided into test and generalization set.

In essence, the network learns by trying to recognize the dynamics of the training set, checks how it fits on the test set and then applies itself to a set of data (generalization) never observed before.

There are no universally valid rules for the subdivision of the dataset: the solutions adopted are (60% – 20% – 20%) and (60% – 30% – 10%) respectively for the training, test and generalization set.

Number of Hidden Layers and Neurons

There are numerous empirical studies that use a single hidden layer, as it is sufficient to approximate non-linear functions with a high degree of accuracy. However, this approach requires a high number of neurons, so limiting the learning process. Sometimes networks with two hidden layers are more effective, especially for the prediction of high-frequency data.

This choice, in addition to being suggested by a specific theory, is supported by the experience that shows how, on the other hand, more than two hidden layers do not produce significant improvements in the results obtained by the network.

It should also be noted that an excessive number of neurons can generate overlearning, i.e. the neurons are not able to generate a reliable prediction because they reduce the contribution of the inputs; in these cases it is as if the network had "retained" the correct answers, without being able to generalize. On the contrary, a too low number of neurons reduce the network's learning potential.

The formulas proposed in literature are very different, and in some cases contradictory:

$$h = 2 \cdot n + 1 \quad (3)$$

$$h = 2 \cdot n \quad (4)$$

$$h = n \quad (5)$$

$$h = (n+m)/2 + t^{1/2} \quad (6)$$

where

h = number of hidden neurons;
n = number of input neurons;
m = number of output neurons;
t = number of observations contained in the training set.

The empirical results show that none of these rules appears generalizable to every problem, although a not meaningless number of positive results seem to prefer (5).

Connection Mechanisms

There are several connection modes between the different layers.

- **Standard Connections:** Provide direct connections, without feedback, between input and output that pass through one or more hidden layers.
- **Jump Connections:** The net can assign connective weights even between neurons not in adjacent layers.
- **Multiple Connections:** Provide for the possibility that neurons assigned to the hidden layers can return to the input variables with iterative processes, in order to precisely quantify the connection's weight.

Activation Function

We can distinguish several types of different functional forms (see Table 1) that affect the binding of the neurons (linear, sinusoidal, Gaussian, etc.). You can define a different activation function for each layer.

There is no rule theoretically acceptable to define the activation function of the various layers. Although many studies present different functions for the individual layers, some do have the same function for the input, hidden and output layers.

The linear function is typically used for the output layer. Its use is less effective, however, in the hidden layers, especially if these are characterized by a high number of neurons.

The logistic and symmetric logistic functions have the characteristic to vary, respectively, in the intervals [0, 1] and [−1, 1]. Some problems have dynamic characteristics that are grasped in a more precise measure by the symmetric function, especially in the input and hidden layers. Most of the empirical literature shows the use of this function in the hidden layer, although without a strong theoretical motivation.

The hyperbolic tangent function allows adapting the network in a reliable manner in the hidden layers (particularly in networks with three layers), especially in the case in which the analyst has chosen a logistic or linear function for output.

Table 1. Activation functions

Function	Formula
Linear	$f(x) = x$
Logistic (sigmoid)	$f(x) = 1/(1+e^{-x})$
Logistic symmetric	$f(x) = (1+e^{-x})/2$
Hyperbolic tangent	$f(x) = (e^{x}-e^{-x})/(e^{x}+e^{-x})$
Corrected tangent	$f(x) = \tanh(c \bullet x)$
Sinusoidal	$f(x) = \sin(x)$
Gaussian	$f(x) = e^{-x\wedge 2}$
Inverse Gaussian	$f(x) = 1-e^{-x\wedge 2}$

The sinusoidal function is generally used in research and it is suggested to normalize input and output in the range [−1, 1].

The Gaussian function lends itself to the identification of specific dynamic processes caught with two hidden parallel layers architectures, with a tangent function in the second layer.

Learning Rules

After defining the initial characteristics of the neural network, you must define the criteria for stopping learning. If you want to build a neural network with forecasting purposes, it is preferable to assess its learning on the test set; if, instead, you want to adequately describe the studied phenomenon it is preferable to assess learning on the training set. Further, learning parameters are related to network's error indicators (average error, maximum error, number of times with no error improvement).

Updating the Connection Weights

A further problem is the choice of the learning rule: in particular, it is necessary to decide the rate at which the network changes weights compared to error. Consider Figure 3. On the left you can see progress both in learning and oscillation, with a frequency that achieves the desired result more quickly; on the right hand the network learns by changing its "path" with a lower rate and reaches the target at a higher time.

The learning rate η is normally initially set to the value:

$$\eta = [\max(x) - \min(x)] / N \quad (7)$$

where x represents the training set with N input records. Through the compensation for the input values x and relative number N this initial value of η should be fine for many classification problems, regardless of the number of training samples and their values' range. However, although highest values of η may accelerate the learning process, this could induce oscillations that can slow convergence.

Alternatively, it is possible to make use of momentum, i.e. the proportion with which the variation of the last weight reached by the neural network is added to the new weight. In this way, the network can learn even at a higher rate but it is not likely to swing since it fetches a high proportion of the last weight loss.

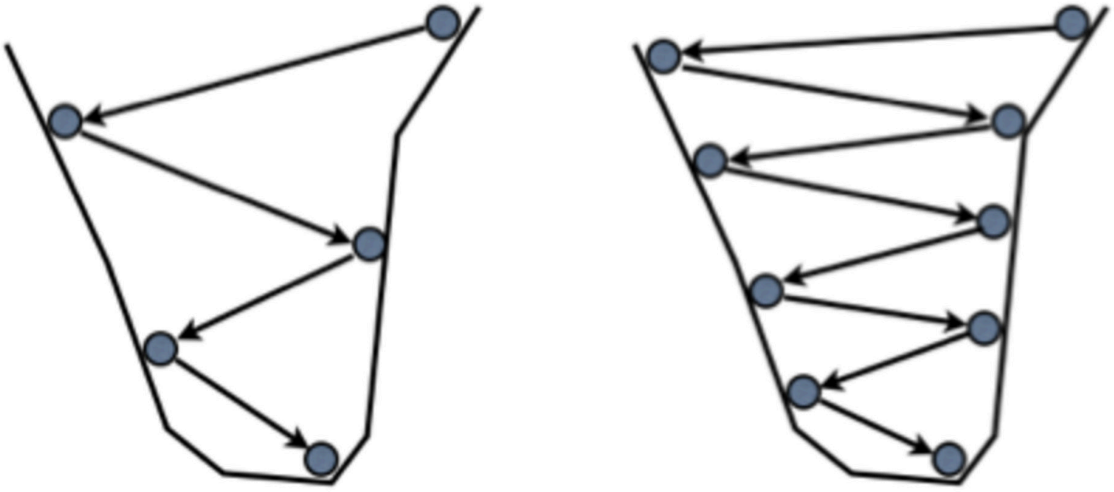
The analyst must then decide the level of the initial connection weights between neurons and assess whether high noise levels characterize the observations. In this case, you should keep up the value of momentum. The network's learning process passes naturally from the progressive identification of the most suitable values for these parameters. It therefore requires a long series of attempts that can generate significantly different results. The suggestion is to avoid radical changes in the parameters.

Error Indicators

Learning parameters are generally related to the network's error indicators:

- Mean error;
- Maximum error;

Figure 3. Network processing with different learning rates



- Number of epochs with no error improvement.

By setting a value for these parameters, the network will stop when it reaches the desired value. It is usually easier to fix a large number of epochs (which are influenced by the number of observations of the training set) that the network analyzes without improving the error. It can be assumed that a value between 10,000 and 50,000 epochs is safe enough to block a network that with great difficulty can learn more than it has done up to that point. Of course, the choice also depends on the speed with which the network reaches these values.

An item for the acceptance of a network is convergence. If errors are modest but the oscillation has led to a high divergence, it is advisable to check the adequacy of the parameters (in particular, learning rate and momentum).

Once the correct temporal dynamics of the error has been verified, at least in graphical terms, you must measure it quantitatively. Among the various error indicators developed in statistics, we point out:

1. Determination Index (R^2)
2. Mean Absolute Error (MAE)
3. Mean Absolute Percentage Error (MAPE)
4. Mean Square Error (MSE)
5. Root Mean Square Error (RMSE)

These indicators measure, in various ways, the spread between the original and the estimated output from the network. Only if the input neurons, activation functions and parameters described above were perfectly able to identify the original phenomenon, the difference between actual and estimated output would be zero, thus optimizing the above mentioned error indicators.

Time Series Prediction

Once the neural network has been built properly, you need to verify its forecasting "goodness". It is possible that a model is able to describe well the training and test set, but then it appears entirely inadequate as regards its generalization, i.e. — in the forecasting case — prediction.

You have, therefore, to test the neural network on generalization set with the same techniques already described. First, you should measure the above-described error indicators on the series never observed by the network. If these prove to be significantly worse and, however, not acceptable on the basis of the original objectives, the network will be further tested until it reaches an acceptable forecasting ability.

CONCLUSION

A neural network is a sophisticated statistical system (operating in parallel) with good noise immunity (Liu, Zhang, & Polycarpou, 2011): if some unit of the system were to malfunction, the network as a whole would have some reductions but hardly would experience a crash.

The models produced by neural networks, although very efficient, cannot be explained in human symbolic language: the result must be accepted "as is" (black box). In other words — as opposed to an algorithmic system — a neural network is able to generate a valid result (or at least with a high likelihood of being acceptable) but it is not possible to explain how and why this result has been generated.

As with any modeling algorithm, neural networks are efficient only if the predictor variables are chosen with care and are not able to deal well with categorical variables (for example, the name of a city) with many different values. They require a training phase that fixes the weights of individual connections, and this may take some time if the number of samples and variables analyzed is very large. There are no theorems or models to define the optimal network, so the success of a network depends very much on the experience of its maker.

FUTURE RESEARCH DIRECTIONS

Neural networks are typically used in contexts where the data may be partially incorrect, or where there are no analytical models able to deal with the problem (Haykin, 1999). A typical use is in OCR software, in face recognition systems and, more generally, in systems that deal with the processing of data subject to errors or noise. They are also one of the most used tools in the analysis of Data Mining. Neural networks are a promising powerful mean for financial analysis (Gately, 1995) or weather forecasting (Zhang & Patuwo, 1998). In recent years their importance has greatly increased also in the field of bioinformatics (Herrero et al., 2001), in which they can be used to search for functional patterns and/or structural proteins and nucleic acids and analysis of data from EEG. If suitably given a long series of inputs (learning or training phase), the network is able to provide the more likely output. Recent studies (Panakkat, & Adeli, 2007) have shown a good potential of neural networks in seismology for the location of epicenters of earthquakes and prediction of their intensity.

REFERENCES

Bandy, H. (1997). Developing a Neural Network System. AAII Working Paper.

Gately, E. (1995). *Neural Networks for Financial Forecasting*. New York: John Wiley & Sons.

Haykin, S. (1999). *Neural Networks: a comprehensive foundation* (2nd ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Herrero, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. [Oxford University Press.]. *Bioinformatics (Oxford, England)*, *17*(2), 126–136. doi:10.1093/bioinformatics/17.2.126 PMID:11238068

Honkela, T., Odzis Aw Duch, W., & Girolami, M. (Eds.). (2011). Artificial Neural Networks and Machine Learning: ICANN 2011, part 1-2. Springer.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366. doi:10.1016/0893-6080(89)90020-8

Liu, D., Zhang, H., & Polycarpou, M. (2011). Advances in Neural Networks. ISNN 2011: 8th International Symposium on Neural Networks, Guilin, China, May 29–June 1, 2011, Proceedings, Springer.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*, 115–133. doi:10.1007/BF02478259

Minsky, M., & Papert, S. (1969). *Perceptron: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press.

Panakkat, A., & Adeli, H. (2007). Neural network models for earthquake magnitude prediction using multiple seismicity indicators. *International Journal of Neural Systems*, *17*(1), 13–33. doi:10.1142/S0129065707000890 PMID:17393560

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408. doi:10.1037/h0042519 PMID:13602029

Wai Wong, K., Sumudu, B., Mendis, U., & Bouzerdoum, A. (Eds.). (2011). *Neural Information Processing: Models and Applications*. Springer.

Zhang, G. P., & Patuwo, M. Y. H. (1998). Forecasting with artificial neural networks: The state of the art. *International Archives of Photogrammetry and Remote Sensing*, *1*(14), 35–62.

ADDITIONAL READING

Abraham, A. (2005). *Artificial Neural Networks*. John Wiley & Sons, Ltd.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. New York, Heidelberg: Springer.

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis, Forecasting and Control*. Oakland, CA, USA: Holden-Day.

Fletcher, R. (1987). *Practical Methods of Optimization*. Chippenham, Great Britain: John Wiley & Sons.

Freeman, J. A., & Skapura, D. M. (1991). *Neural networks: algorithms, applications, and programming techniques*. USA: Addison-Wesley.

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*. MA, USA: The MIT Press Cambridge.

Heaton, J. (2008). *Introduction to Neural Networks with Java* (2nd ed.). Chesterfield, MO, USA: Heaton Research Inc.

Hecht-Nielsen, R. (1989). *Neurocomputing*. Reading, MA, USA: Addison-Wesley.

Herz, J., Krough, A., & Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Reading, MA, USA: Addison-Wesley.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences of the United States of America*, *79*, 2554–2558. doi:10.1073/pnas.79.8.2554 PMID:6953413

Hopfield, J. J. (1984). Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons. *Proceedings of the National Academy of Sciences of the United States of America*, *81*, 3088–3092. doi:10.1073/pnas.81.10.3088 PMID:6587342

Johansson, R. (1993). *System Modeling and Identification*. Englewood Cliffs, NJ, USA: Prentice Hall.

Kohonen, T. (1995). *Self-Organizing Maps*. Berlin, Germany: Springer-Verlag. doi:10.1007/978-3-642-97610-0

Lek, S., Delacoste, M., Baran, P., Dimopoulos, I., Lauga, J., & Aulagnier, S. (1996). Application of neural networks to modelling non-linear relationships in ecology. *Ecological Modelling*, *90*, 39–52. doi:10.1016/0304-3800(95)00142-5

Murata, N., Yoshizawa, S., & Amari, S.-I. (1994). Network information criterion-determining the number of hidden units for an artificial neural network model. *IEEE Transactions on Neural Networks*, *5*(6), 865–872. doi:10.1109/72.329683 PMID:18267861

Paruelo, M., & Tomasel, F. (1997). Prediction of functional characteristics of ecosystems: a comparison of artificial neural network and regression models. *Ecological Modelling*, *98*, 173–186. doi:10.1016/S0304-3800(96)01913-8

Sjöberg, J. et al. (1995). Non-Linear Black-Box Modeling in System Identification: A Unified Overview. *Automatica*, *31*(12), 1691–1724. doi:10.1016/0005-1098(95)00120-8

Sjöberg, J., & Ljung, L. (1995). Overtraining, Regularization, and Searching for Minimum with Application to Neural Nets. *International Journal of Control*, *62*(6), 1391–1407. doi:10.1080/00207179508921605

Weigend, A. S., & Gershenfeld, N. A. (1994). Time Series Prediction: Forecasting the Future and Understanding the Past. In Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis held in Santa Fe, New Mexico, May 14–17, 1992. Addison-Wesley, Reading, MA, USA.

Yao, X. (1993). Evolutionary Artificial Neural Networks. *International Journal of Neural Systems*, *4*(3), 203–222. doi:10.1142/S0129065793000171 PMID:8293227

Zhou, Z.-H., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence* **137**(1–2):239–263, ISSN 0004-3702, http://dx.doi.org/10.1016/S0004-3702(02)00190-X.

KEY TERMS AND DEFINTIONS

Artificial Intelligence: The term "artificial intelligence" generally refers to the ability of a computer to perform functions and reasoning typical of the human mind. It covers the theory and techniques for the development of algorithms that allow computers to show an ability and/or intelligent activity, at least in specific domains.

Artificial Neural Network (ANN): An artificial neural network defines a mathematical model for the simulation of a network of biological neurons (e.g. human nervous system). It simulates different aspects related to the behavior and capacity of the human brain, such as: intelligent information processing; distributed processing; high level of parallelism; faculty of learning, generalization and adaptation; high tolerance to inaccurate (or wrong) information.

Learning: The acquisition or modification of knowledge (new or existing), behaviors, skills, values, or preferences and may involve the synthesis of different types of information.

Mathematical Model: A mathematical model is a model built using the language and tools of mathematics. A mathematical model is often constructed with the aim to provide predictions on the future 'state' of a phenomenon or a system.

Neuron: The neuron is the unit cell, which constitutes the nervous tissue, contributing to the formation (together with the fabric of the neuroglia and the vascular tissue) of the nervous system.

Perceptron: A type of binary classifier that maps its inputs (a vector of real type) to an output value (a scalar real type). The perceptron may be considered as the simplest model of feed-forward neural network, as the inputs directly feeding the output units through weighted connections.

Synapse: The synapse (or synaptic junction) is a highly specialized structure that enables communication of nerve cells together (neurons) or with other cells (muscle cells, sensory or endocrine glands).