



Extreme Machine Learning: Feed Forward Networks

Akshay Sharma*

Dronacharya College of Engg.(CSE,
Gurgaon, India

Babita Sandooja

Dronacharya College of Engg.(CSE)
Gurgaon, India

Deepika Yadav

Dronacharya College of Engg.(CSE)
Gurgaon, India

Abstract—The class of adaptive systems known as Artificial Neural Networks (ANN) was motivated by the amazing parallel processing capabilities of biological brains (especially the human brain). The main driving force was to re-create these abilities by constructing artificial models of the biological neuron. This paper is focused to feed-forward networks. The field has become so vast that a complete and clear-cut description of all the approaches is an enormous undertaking. The power of biological neural structures stems from the enormous number of highly interconnected simple units. The simplicity comes from the fact that, once the complex electro-chemical processes are abstracted, the resulting computation turns out to be conceptually very simple. These artificial neurons have nowadays little in common with their biological counterpart in the ANN paradigm. Rather, they are primarily used as computational devices, clearly intended to problem solving: optimization, function approximation, classification, time-series prediction and others. In practice few elements are connected and their connectivity is low.

Keywords—Adaptive systems, ANN paradigm, feed-forward networks, Optimization, parallel processing

I. Introduction

The answer to the theoretical question: “Can a machine be built capable of doing what the brain does?” is yes, provided you specify in a finite and unambiguous way what the brain does. -- Warren S. McCulloch The class of adaptive systems known as Artificial Neural Networks (ANN) was motivated by the amazing parallel processing capabilities of biological brains (especially the human brain). The main driving force was to re-create these abilities by constructing artificial models of the biological neuron. The power of biological neural structures stems from the enormous number of highly interconnected simple units. The simplicity comes from the fact that, once the complex electro-chemical processes are abstracted, the resulting computation turns out to be conceptually very simple [1]. The feed-forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network [2].

A feed-forward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit in a layer is connected with all the units in the previous layer. These connections are not all equal; each connection may have a different strength or weight where connections between the units do not form a directed cycle. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. . This is different from recurrent neural networks. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feed-forward neural networks [3].

As shown in Fig. 1 we see an example of a 2-layered network with, from top to bottom: an output layer with 5 units, a *hidden* layer with 4 units, respectively. The network has 3 input units.

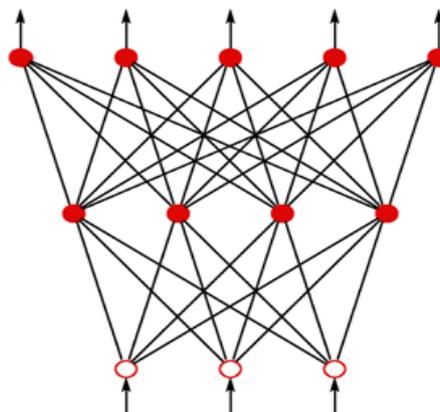


Fig. 1 Feed forward Network

The 3 inputs are shown as circles and these do not belong to any layer of the network (although the inputs sometimes are considered as a virtual layer with layer number 0). Any layer that is not an output layer is a *hidden* layer. This network therefore has 1 hidden layer and 1 output layer. The figure also shows all the connections between the units in different layers. A layer only connects to the previous layer.

The operation of this network can be divided into two phases:

1. The learning phase
2. The classification phase

These artificial neurons have nowadays little in common with their biological counterpart in the ANN paradigm. Rather, they are primarily used as computational devices, clearly intended to problem solving: optimization, function approximation, classification, time-series prediction and others. In practice few elements are connected and their connectivity is low. This paper is focused to supervised feed-forward networks. The field has become so vast that a complete and clear-cut description of all the approaches is an enormous undertaking.

II. Learning of Feed-Forward Networks

During the learning phase the weights in the Feed-Forward Network will be modified. All weights are modified in such a way that when a pattern is presented, the output unit with the correct category, hopefully, will have the largest output value. The Feed-Forward Network uses a *supervised* learning algorithm: besides the input pattern, the neural net also needs to know to what category the pattern belongs. Learning proceeds as follows: a pattern is presented at the inputs. The pattern will be transformed in its passage through the layers of the network until it reaches the output layer. The units in the output layer all belong to a different category. The outputs of the network as they are now are compared with the outputs as they ideally would have been if this pattern were correctly classified: in the latter case the unit with the correct category would have had the largest output value and the output values of the other output units would have been very small. On the basis of this comparison all the connection weights are modified a little bit to guarantee that, the next time this same pattern is presented at the inputs, the value of the output unit that corresponds with the correct category is a little bit higher than it is now and that, at the same time, the output values of all the other incorrect outputs are a little bit lower than they are now. (The differences between the actual outputs and the idealized outputs are propagated back from the top layer to lower layers to be used at these layers to modify connection weights. This is why the term *back propagation network* is also often used to describe this type of neural network [3].

If you perform the procedure above once for every pattern and category pair in your data set you have performed 1 epoch of learning. The hope is that eventually, probably after many epochs, the neural net will remember these pattern-category pairs. You even hope that the neural net when the learning phase has terminated, will be able to *generalize* and has learned to classify correctly any unknown pattern presented to it. Because real-life data many times contains noise as well as partly contradictory information these hopes can only be partly fulfilled. For learning you need to select 3 different objects together: a Feed-Forward Network (the *classifier*), a Pattern (the *inputs*) and categories (the *correct outputs*) [4].

How long will the learning phase take? In general this question is hard to answer. It depends on the size of the neural network, the number of patterns to be learned, the number of epochs, the tolerance of the minimiser and the speed of your computer, how much computing time the learning phase may take.

III. Single-Layer Perceptron

The simplest kind of neural network is a *single-layer* network (called Perceptron), which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights. In this way as shown in Fig. 2, it can be considered the simplest kind of feed-forward network. The Perceptron is a single-layer artificial network with only one neuron. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). The neuron unit calculates the linear combination of its real-valued or Boolean inputs and passes it through a threshold activation function, according to Eq. 1.

$$o = \text{Threshold} \left(\sum_{i=0}^d w_i x_i \right) \quad (1)$$

where x_i are the components of the input $\mathbf{x}_e = (x_{e1}, x_{e2}, \dots, x_{ed})$ from the set $\{(\mathbf{x}_e, y_e)\}_{e=1}^N$

Threshold is the activation function defined as follows: $\text{Threshold}(s) = 1$ if $s > 0$ and -1 otherwise

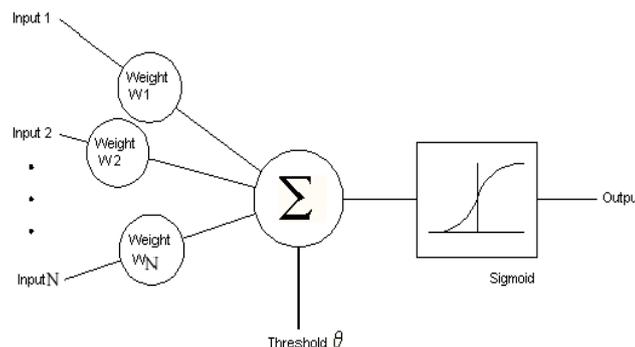


Fig.2 Perceptron

Neurons with this kind of activation function are also called artificial neurons or linear threshold units. In the literature the term Perceptron often refers to networks consisting of just one of these units. A similar neuron was described by Warren McCulloch and Walter Pitts in the 1940s. The Perceptron is sometimes referred to a threshold logic unit (TLU) since it discriminates the data depending on whether the sum is greater than the threshold value $\sum_{i=1}^d w_i x_i > -w_0$ or the sum is less than the threshold value $\sum_{i=1}^d w_i x_i < -w_0$. In the above formulation we imagine that the threshold value w_0 is the weight of an additional connection held constantly to $x_0 = 1$ [5].

The Perceptron is strictly equivalent to a linear discriminant, shown in Fig. 3, and it is often used as a device that decides whether an input pattern belongs to one of two classes.

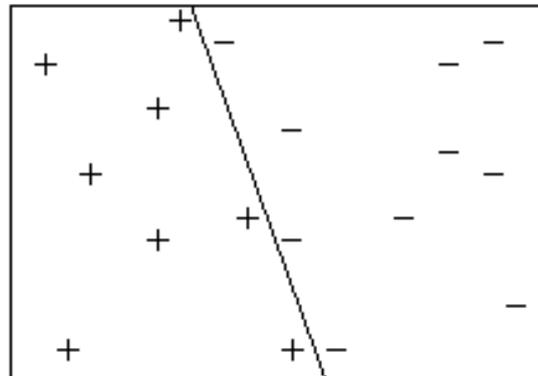


Fig. 3 Linear Discriminant

Networks of such threshold units can represent a rich variety of functions while single units alone cannot. For example, every Boolean function can be presented by some network of interconnected units. The Perceptron can represent most of the primitive Boolean functions in Fig. 4: AND, OR, NAND and NOR but cannot represent XOR [6].

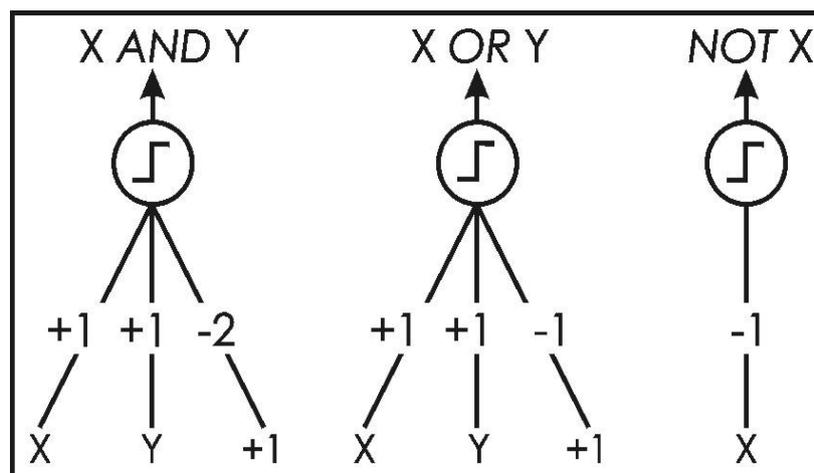


Fig. 4 McCulloch-Pitts networks

Most perceptrons have outputs of 1 or -1 with a threshold of 0 and there is some evidence that such networks can be trained more quickly than networks created from nodes with different activation and deactivation values. Perceptrons can be trained by a simple learning algorithm that is usually called the *delta rule*. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent. Single-unit perceptrons are only capable of learning linearly separable patterns; in 1969 in a famous monograph entitled *Perceptrons*, Marvin Minsky and Seymour Papert showed that it was impossible for a single-layer Perceptron network to learn an XOR function. It is often believed that they also conjectured (incorrectly) that a similar result would hold for a multi-layer Perceptron network. However, this is not true, as both Minsky and Papert already knew that multi-layer Perceptrons were capable of producing an XOR Function.

Although a single threshold unit is quite limited in its computational power, it has been shown that networks of parallel threshold units can approximate any continuous function from a compact interval of the real numbers into the interval $[-1,1]$. This very recent result can be found in Peter Auer, Harald Burgsteiner and Wolfgang Maass "A learning rule for very simple universal approximators consisting of a single layer of Perceptron".

A multi-layer neural network can compute a continuous output instead of a step function. A common choice is the so-called logistic function, according to Eq. 2.

$$y = \frac{1}{1 + e^{-x}} \quad (2)$$

(In general form, $f(X)$ is in place of x , where $f(X)$ is an analytic function in set of x 's.) With this choice, the single-layer network is identical to the logistic regression model, widely used in statistical modeling. The logistic function, y , is also known as the sigmoid function. It has a continuous derivative, y' , which allows it to be used in back propagation. According to Eq. 3, this function, is also preferred because its derivative is easily calculated:

$$y' = y(1 - y) \quad (3)$$

IV. MULTI-LAYER FEED-FORWARD NEURAL NETWORKS

Multi-Layer Feed-Forward neural networks, trained with a back-propagation learning algorithm, are the most popular neural networks. They are applied to a wide variety of chemistry related problems [7]. A Multi-Layer Feed-Forward neural network consists of neurons that are ordered into layers shown in Fig. 5. The first layer is called the input layer, the last layer is called the output layer, and the layers between are hidden layers.

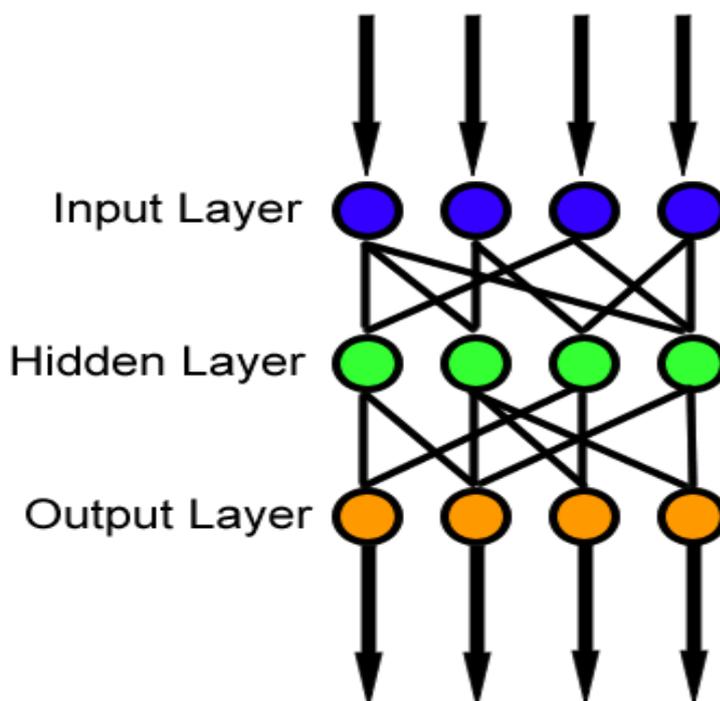


Fig. 5 Typical feed forward neural network composed of three layers

For the formal description of the neurons we can use the so-called mapping function Γ , that assigns for each neuron i a subset $\Gamma(i) \subset V$ which consists of all ancestors of the given neuron. A subset $\Gamma^{-1}(i) \subset V$ then consists of all predecessors of the given neuron i . Each neuron in a particular layer is connected with all neurons in the next layer. The connection between the i_{th} and j_{th} neuron is characterised by the weight coefficient w_{ij} and the i_{th} neuron by the threshold coefficient ϑ_i . The weight coefficient reflects the degree of importance of the given connection in the neural network. The output value (activity) of the i_{th} neuron x_i is determined by Equation 4. It holds that:

$$x_i = f(\xi_i) \quad (4)$$

$$\xi_i = \vartheta_i + \sum_{j \in \Gamma^{-1}(i)} w_{ij} x_j$$

where ξ_i is the potential of the i th neuron and function $f(\xi_i)$ is the so-called transfer function (the summation in Equation is carried out over all neurons j transferring the signal to the i_{th} neuron). The threshold coefficient can be understood as a weight coefficient of the connection with formally added neuron j , where $x_j = 1$ (so-called bias).

The supervised adaptation process varies the threshold coefficients ϑ_i and weight coefficients w_{ij} to minimise the sum of the squared differences between the computed and required output values.

A two-layer neural network capable of calculating XOR, shown in Fig 6. The numbers within the neurons represent each neuron's explicit threshold (which can be factored out so that all neurons have the same threshold. The numbers that annotate arrows represent the weight of the inputs. This net assumes that if the threshold is not reached, zero (not -1) is output. Note that the bottom layer of inputs is not always considered a real neural network layer

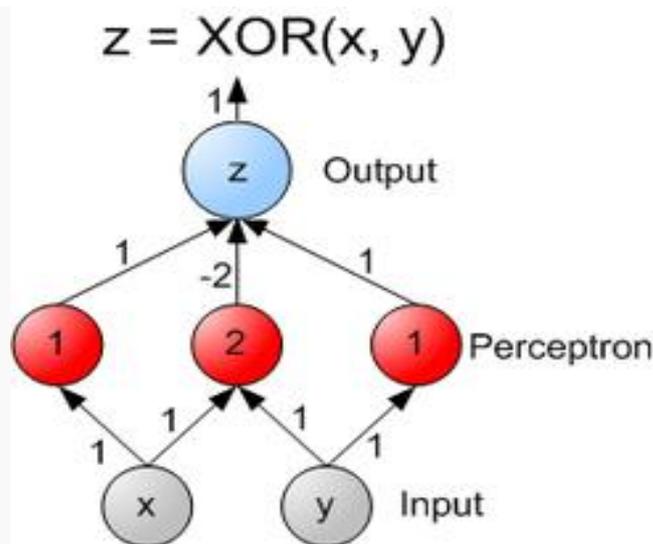


Fig. 6 two-layer neural network capable of calculating XOR

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a sigmoid function as an activation function. The universal approximation theorem for neural networks states that every continuous function that maps intervals of real numbers to some output interval of real numbers can be approximated arbitrarily closely by a multi-layer perceptron with just one hidden layer. This result holds only for restricted classes of activation functions, e.g. for the sigmoidal functions.

Multi-layer networks use a variety of learning techniques, the most popular being back-propagation. Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, one would say that the network has learned a certain target function. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. For this, the derivative of the error function with respect to the network weights is calculated, and the weights are then changed such that the error decreases (thus going downhill on the surface of the error function). For this reason, back-propagation can only be applied on networks with differentiable activation functions. In general, the problem of teaching a network to perform well, even on samples that were not used as training samples, is a quite subtle issue that requires additional techniques. This is especially important for cases where only very limited numbers of training samples are available [8]. The danger is that the network over fits the training data and fails to capture the true statistical process generating the data. Computational learning theory is concerned with training classifiers on a limited amount of data. In the context of neural networks a simple heuristic, called early stopping, often ensures that the network will generalize well to examples not in the training set. Other typical problems of the back-propagation algorithm are the speed of convergence and the possibility of ending up in a local minimum of the error function. Today there are practical solutions that make back-propagation in multi-layer perceptrons the solution of choice for many machine learning tasks.

V. ADVANTAGES AND DISADVANTAGES OF FEED-FORWARD NEURAL NETWORKS

A. *The application of MLF neural networks offers the following useful properties and capabilities:*

- 1) *Learning:* ANNs are able to adapt without assistance of the user.
- 2) *Nonlinearity:* A neuron is a non-linear device. Consequently, a neural network is itself non-linear. Nonlinearity is very important property, particularly, if the relationship between input and output is inherently non-linear.
- 3) *Input-output mapping:* In supervised training, each example consists of a unique input signal and the corresponding desired response. An example picked from the training set is presented to the network, and the weight coefficients are modified so as to minimise the difference between the desired output and the actual response of the network. The training of the network is repeated for many examples in the training set until the network reaches the stable state. Thus the network learns from the examples by constructing an input-output mapping for the problem [9].
- 4) *Robustness:* MLF neural networks are very robust, i.e. their performance degrades gracefully in the presence of increasing amounts of noise.

B. *Disadvantages*

However, there are some problems and disadvantages of ANNs too. For some problems approximation via sigmoidal functions ANN are slowly converging - a reflection of the fact that no physical in-sight is used in the construction of the

approximating mapping of parameters on the result. The big problem is the fact, that ANNs cannot explain their prediction, the processes taking place during the training of a network are not well interpretable and this area is still under development [9], [10]. The number of weights in an ANN is usually quite large and time for training the ANN is too high.

VI. Conclusion

Although the long-term goal of the neural-network community remains the design of autonomous machine intelligence, the main modern application of artificial neural networks is in the field of pattern recognition. In the sub-field of data classification, neural-network methods have been found to be useful alternatives to statistical techniques such as those which involve regression analysis or probability density estimation. The potential utility of neural networks in the classification of multisource satellite-imagery databases has been recognized for well over a decade, and today neural networks are an established tool in the field of remote sensing.

Acknowledgments

We thank our guide for his timely help, giving outstanding ideas and encouragement to finish this research work successfully.

References

- [1] <http://www.igi-global.com/chapter/learning-feed-forward-artificial-neural/10365>
- [2] Auer, Peter; Harald Burgsteiner; Wolfgang Maass (2008). "A learning rule for very simple universal approximators consisting of a single layer of perceptrons". *Neural Networks* 21(5): 795. doi:10.1016/j.neunet.2007.12.036. PMID 18249524.
- [3] http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1__What_is_a_feedforward_network
- [4] <http://www.emilstefanov.net/Projects/NeuralNetworks.aspx>
- [5] http://www.google.co.in/search?q=learning+of+feed+forward+network&biw=1280&bih=650&source=lnms&tbm=isch&sa=X&ei=PagPUuT7KsWCrgeXiIHYCw&ved=0CAcQ_AUoAQ
- [6] http://www.webpages.ttu.edu/dleverin/neural_network/neural_networks.html
- [7] http://ich.vscht.cz/~svozil/pubs/nn_intro.pdf
- [8] S. Haykin, *Neural Networks - A Comprehensive Foundation*, Macmillan, 1994.
- [9] R. Andrews, J. Diedrich, A.B. Tickle, *A survey and critiques for extracting rules from trained artificial neural networks*, Internal printing of the Neurocomputing Research centre, Queensland University of Technology, Brisbane, 1995.
- [10] J.W. Shavlik, *A Framework for Combining Symbolic and Neural Learning*, CS-TR-92-1123, November 1992, The University of Wisconsin.