Feed-Forward Artificial Neural Network Basics

Lluís A. Belanche Muñoz
Universitat Politècnica de Catalunya, Spain

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

INTRODUCTION

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.

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 chapter 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; we refer the reader to (Fiesler & Beale, 1997) for a comprehensive exposition.

BACKGROUND

Artificial Neural Networks (Bishop, 1995), (Haykin, 1994), (Hertz, Krogh & Palmer, 1991), (Hecht-Nielsen, 1990) are information processing structures without global or shared memory, where each of the computing elements operates only when all its incoming information is available, a kind of data-flow architectures. Each element is a simple processor with internal and adjustable parameters. The interest in ANN is primarily related to the finding of satisfactory solutions for problems cast as function approximation tasks and for which there is scarce or null knowledge about the process itself, but a (limited) access to examples of response. They have been widely and most fruitfully used in a variety of applications—see (Fiesler & Beale, 1997) for a comprehensive review—especially after the boosting works of (Hopfield, 1982), (Rumelhart, Hinton & Williams, 1986) and (Fukushima, 1980).

The most general form for an ANN is a *labelled directed graph*, where each of the nodes (called *units* or *neurons*) has a certain computing ability and is connected to and from other nodes in the network via labelled edges. The edge label is a real number expressing the strength with which the two involved units are connected. These labels are called *weights*. The *architecture* of a network refers to the number of units, their arrangement and connectivity.

In its basic form, the computation of a unit i is expressed as a function F_i of its input (the *transfer* function), parameterized with its weight vector or local information. The whole system is thus a collection of interconnected elements, and the transfer function performed by a single one (i.e., the *neuron model*) is the most important fixed characteristic of the system.

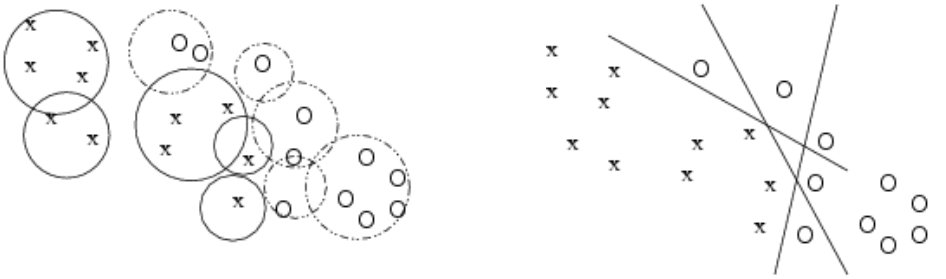
There are two basic types of neuron models in the literature used in practice. Both express the overall computation of the unit as the composition of two functions, as is classically done since the earlier model proposal of McCulloch & Pitts (1943):

$$F_i(\boldsymbol{x}) = \{g(h(\boldsymbol{x},\boldsymbol{w}_i)),\ \boldsymbol{w}_i \in R^n\},\quad \boldsymbol{x}\in R^n \qquad (1)$$

where $\boldsymbol{w}_i$ is the weight vector of neuron i, $h:R^n\times R^n\rightarrow R$ is called the *net input* or *aggregation* function, and $g:R\rightarrow R$ is called the *activation* function. All neuron parameters are included in its weight vector.

The choice $h(\boldsymbol{x},\boldsymbol{w}_i)=\boldsymbol{x}\cdot\boldsymbol{w}_i+\theta$, where $\theta\in R$ is an offset term that may be included in the weight vector, leads to one of the most widely used neuron models. When

Figure 1. A classification problem. Left: Separation by spherical RBF units (R-neurons). Right: Separation by straight lines (P-neurons) in the MLP.



neurons of this type are arranged in a feed-forward architecture, the obtained neural network is called MultiLayer Perceptron (MLP) (Rumelhart, Hinton & Williams, 1986). Usually, a smooth non-linear and monotonic function is used as *activation*. Among them, the sigmoids are a preferred choice.

The choice $h(\boldsymbol{x},\boldsymbol{w}_i)= ||\boldsymbol{x}-\boldsymbol{w}_i||/\theta$ (or other distance measure), with $\theta>0\in R$ a smoothing term, plus an activation g with a monotonically decreasing response from the origin, leads to the wide family of localized Radial Basis Function networks (RBF) (Poggio & Girosi, 1989). Localized means that the units give a significant response only in a neighbourhood of their centre $\boldsymbol{w}_i$. A Gaussian $g(z)=exp(-z^2/2)$ is a preferred choice for the activation function.

The previous choices can be extended to take into account extra correlations between input variables. The inner product (containing no cross-product terms) can be generalized to a real quadratic form (an homogeneous polynomial of second degree with real coefficients) or even further to higher degrees, leading to the so-called higher-order units (or Σ–Π units). A higher-order unit of degree k includes all possible cross-products of at most k input variables, each with its own weight. Conversely, basic Euclidean distances can be generalized to completely weighted distance measures, where all the (quadratic) cross-products are included. These full expressions are not commonly used because of the high numbers of free parameters they involve.

These two basic neuron models have traditionally been regarded as completely separated, both from a mathematical and a conceptual point of view. To a certain degree, this is true: the local vs. global approximation approaches to a function that they carry out make them apparently quite opposite methods (see Fig. 1). Mathematically, under certain conditions, they can be shown to be related (Dorffner, 1995). These conditions (basically, that *both* input and weight vectors are normalized to unit norm) are difficult to fulfil in practice.

A *layer* is defined as a collection of independent units (not connected with one another) sharing the same input, and of the same functional form (same F_i but different $\boldsymbol{w}_i$). Multilayer feed-forward networks take the form of directed acyclic graphs obtained by concatenation of a number of layers. All the layers but the last (called the output layer) are labelled as *hidden*. This kind of networks (shown in Fig. 2) compute a parameterized function $F_{\underline{\boldsymbol{w}}}(\boldsymbol{x})$ of their input vector $\boldsymbol{x}$ by evaluating the layers in order, giving as final outcome the output of the last layer. The vector $\underline{\boldsymbol{w}}$ represents the collection of all the weights (free parameters) in the network. For simplicity, we are not considering connections between non-adjacent layers (*skip-layer connections*) and assume otherwise total connectivity. The set of input variables is *not* counted as a layer.

Output neurons take the form of a scalar product (a linear combination), eventually followed by an activation function g. For example, assuming a single output neuron, a one-hidden-layer neural network with h hidden units computes a function $F:R^n\rightarrow R$ of the form:

$$F_{\underline{\boldsymbol{w}}}(\boldsymbol{x})=g\left(\sum_{i=1}^{h} c_i F_i(\boldsymbol{x}) - \theta\right) \qquad (2)$$