CHAPTER 10

Neural Networks Fundamentals

10.1 INTRODUCTION

The artificial neural networks, which we describe in this course, are all variations on the parallel distributed processing (PDP) idea. The architecture of each network is based on very similar building blocks, which perform the processing. In this chapter we first discuss these processing units and discuss different network topologies. Learning strategies as a basis for an adaptive system will be presented in the last section.

10.2 BIOLOGICAL NEURAL NETWORK

The term 'neural network' comes from the intended analogy with the functioning of the human brain adopting simplified models of 'biological neural network'. The human brain consists of nearly 1011 neurons (nerve cells) of different types. In a typical neuron, one can find nucleus with which the connections with other neurons are made through a network of fibres called dendrites. Extending out from the nucleus is the axon, which transmits, by means of complex chemical process, electric potentials to the neurons, with which the axon is connected to (Fig. 10.1). When signals, received by neuron, become equal or surpass their threshold values, it 'triggers' sending an electric signal of constant level and duration through axon. In this way, the message is transferred from one neuron to the other.

In the neural network, the neurons or the processing units may have several input paths corresponding to the dendrites. The units combine usually by a simple summation, that is, the weighted values of these paths (Fig. 10.2). The weighted value is passed to the neuron, where it is modified by threshold function such as sigmoid function. The modified value is directly presented to the next neuron.

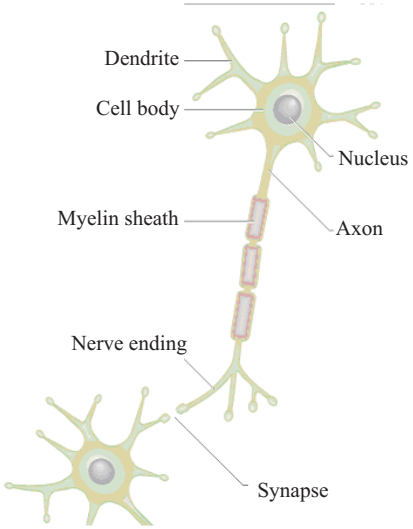

Fig. 10.1 Schematic representation of biological neuron network.

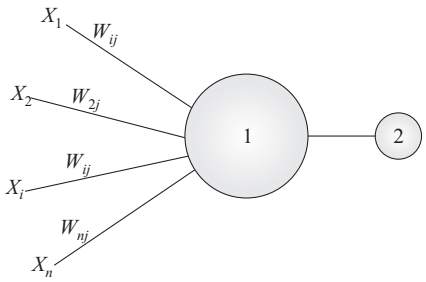

Fig. 10.2 Schematic representation of mathematical neuron network.

10.3 A FRAMEWORK FOR DISTRIBUTED REPRESENTATION

An artificial network consists of a pool of simple processing units, which communicate by sending signals to each other over a large number of weighted connections. A set of major aspects of a parallel distributed model can be distinguished as:

- a set of processing units (‘neurons’, ‘cells’);
- a state of activation y_k for every unit, which is equivalent to the output of the unit;
- connections between the units. Generally each connection is defined by a weight *wjk* which determines the effect which the signal of unit *j* has on unit *k*;
- a propagation rule, which determines the effective input s_k of a unit from its external inputs;
- an activation function F_k, which determines the new level of activation based on the effective input *sk*(*t*) and the current activation $y_k(t)$ (i.e., the update);’

- an external input (aka bias, offset) θ_k for each unit;
- a method for information gathering (the learning rule);
- an environment within which the system must operate, providing input signals and if necessary error signals.

Figure 10.3 illustrates these basics, some of which will be discussed in the next sections.

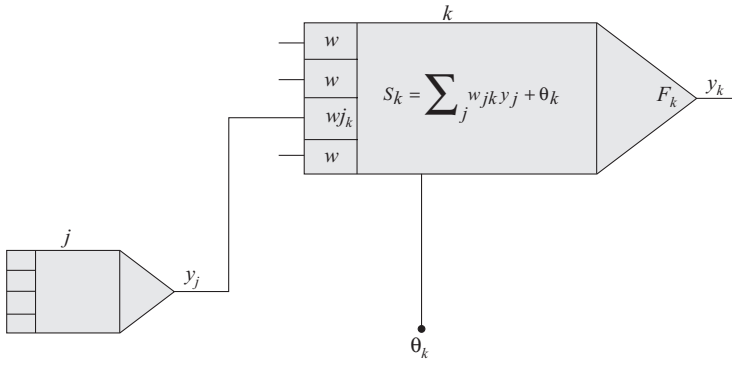

Fig. 10.3 The basic components of an artificial neural network. The propagation rule used here is the 'standard' weighted summation.

10.3.1 Processing Units

Each unit performs a relatively simple job: receive input from neighbors or external sources and use this to compute an output signal, which is propagated to other units. Apart from this processing, a second task is the adjustment of the weights. The system is inherently parallel in the sense that many units can carry out their computations at the same time.

Within neural systems it is useful to distinguish three types of units: input units (indicated by an index *i*) which receive data from outside the neural network, output units (indicated by an index *o*) which send data out of the neural network, and hidden units (indicated by an index *h*) whose input and output signals remain within the neural network.

During operation, units can be updated either synchronously or asynchronously. With synchronous updating, all units update their activation simultaneously; with asynchronous updating, each unit has a (usually fixed) probability of updating its activation at a time *t*, and usually only one unit will be able to do this at a time. In some cases the latter model has some advantages.

10.3.2 Connections between Units

In most cases we assume that each unit provides an additive contribution to the input of the unit with which it is connected. The total input to unit *k* is simply the weighted sum of the separate outputs from each of the connected units plus a bias or offset term θ_k:

$$s_k(t) = \sum_j w_{jk}(t)\, y_j(t) + \theta_k(t) \qquad ...(10.1)$$

The contribution for positive w_{jk} is considered as an excitation and for negative w_{jk} as inhibition. In some cases more complex rules for combining inputs are used, in which a distinction is made between excitatory and inhibitory inputs. We call units with a propagation rule (10.1) sigma units.

A different propagation rule, introduced by Feldman and Ballard, is known as the propagation rule for the sigma-pi unit is given by

$$s_k(t) = \sum_j w_{jk}(t) \prod_m y_{jm}(t) + \theta_k(t) \qquad ...(10.2)$$

Often, the y_{jm} are weighted before multiplication. Although these units are not frequently used, they have their value for gating of input, as well as implementation of lookup tables.

10.3.3 Activation and Output Rules

We also need a rule, which gives the effect of the total input on the activation of the unit. We need a function F_k which takes the total input $s_k(t)$ and the current activation $y_k(t)$ and produces a new value of the activation of the unit k:

$$y_k(t+1) = F_k(y_k(t), s_k(t)). \qquad ...(10.3)$$

Often, the activation function is a non-decreasing function of the total input of the unit:

$$y_k(t+1) = F_k(s_k(t)) = F_k\left(\sum_j w_{jk}(t)\, y_j(t) + \theta_k(t)\right) \qquad ...(10.4)$$

although activation functions are not restricted to non-decreasing functions. Generally, some sort of threshold function is used: a hard limiting threshold function (a sgn function), or a linear or semi-linear function, or a smoothly limiting threshold (see Fig. 10.4). For this smoothly limiting function often a sigmoid (S-shaped) function like

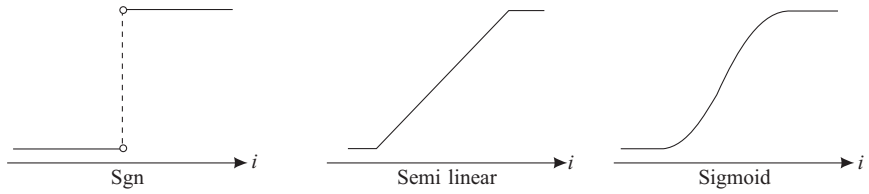

Fig. 10.4 Various activation functions for a unit.

$$y_k = F(s_k) = \frac{1}{1 + e^{-s_k}} \qquad ...(10.5)$$

is used. In some applications a hyperbolic tangent is used, yielding output values in the range [–1; +1].

In some cases, the output of a unit can be a stochastic function of the total input of the unit. In that case the activation is not deterministically determined by the neuron input, but the neuron input determines the probability p that a neuron get a high activation value:

$$p(y_k \leftarrow 1) = \frac{1}{1 + e^{-s_k/T}} \qquad ...(10.6)$$

in which T (temperature) is a parameter which determines the slope of the probability function.

In all networks we consider that the output of a neuron is to be identical to its activation level.

10.4 NETWORK TOPOLOGIES

In the previous section we discussed the properties of the basic processing unit in an artificial neural network. This section focuses on the pattern of connections between the units and the propagation of data.

As for this pattern of connections, the main distinction we can make is between:

- Feed-forward networks, where the data flow from input to output units is strictly feed-forward. The data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers.
- Recurrent networks that do contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the network will evolve to a stable state in which these activations do not change anymore. In other applications, the change of the activation values of the output neurons are significant, such that the dynamical behavior constitutes the output of the network.

Classical examples of feed-forward networks are the Perceptron and Adaline, which will be discussed in the next chapter. Examples of recurrent networks have been presented by Anderson, Kohonen, and Hopfield and will be discussed in subsequent chapters.

10.5 TRAINING OF ARTIFICIAL NEURAL NETWORKS

A neural network has to be configured such that the application of a set of inputs produces (either 'direct' or via a relaxation process) the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a priori knowledge. Another way is to 'train' the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule.

10.5.1 Paradigms of Learning

We can categorize the learning situations in two distinct sorts. These are:

- Supervised learning or Associative learning in which the network is trained by providing it with input and matching output patterns. These input-output pairs can be provided by an external teacher, or by the system, which contains the network (self-supervised).
- Unsupervised learning or Self-organization in which an (output) unit is trained to respond to clusters of pattern within the input. In this paradigm the system is supposed to discover

statistically salient features of the input population. Unlike the supervised learning paradigm, there is no a priori set of categories into which the patterns are to be classified rather the system must develop its own representation of the input stimuli.

10.5.2 Modifying Patterns of Connectivity

Both learning paradigms discussed above result in an adjustment of the weights of the connections between units, according to some modification rule. Virtually all learning rules for models of this type can be considered as a variant of the Hebbian learning rule. The basic idea is that if two units j and k are active simultaneously, their interconnection must be strengthened. If j receives input from k, the simplest version of Hebbian learning prescribes to modify the weight w_{jk} with

$$\Delta w_{jk} = \gamma y_j y_k \qquad ...(10.7)$$

where γ is a positive constant of proportionality representing the learning rate. Another common rule uses not the actual activation of unit k but the difference between the actual and desired activation for adjusting the weights:

$$\Delta w_{jk} = \gamma y_j (d_k - y_k) \qquad ...(10.8)$$

in which d_k is the desired activation provided by a teacher. This is often called the Widrow-Hoff rule or the delta rule, and will be discussed in the next chapter.

Many variants (often very exotic ones) have been published the last few years. In the next chapters some of these update rules will be discussed.

10.6 NOTATION AND TERMINOLOGY

10.6.1 Notation

We use the following notation in our formulae. Note that not all symbols are meaningful for all networks, and that in some cases subscripts or superscripts may be left out (e.g., p is often not necessary) or added (e.g., vectors can, contrariwise to the notation below, have indices) where necessary. Vectors are indicated with a bold non-slanted font:

j, k,…. the unit j, k, …;

i an input unit;

h a hidden unit;

o an output unit;

x^p the pth input pattern vector;

x_j^p the jth element of the pth input pattern vector;

s^p the input to a set of neurons when input pattern vector p is clamped (i.e., presented to the network); often: the input of the network by clamping input pattern vector p;

d^p the desired output of the network when input pattern vector p was input to the network;

d_j^p the jth element of the desired output of the network when input pattern vector p was input to the network;

y^p the activation values of the network when input pattern vector p was input to the network;
y_j^p the activation values of element j of the network when input pattern vector p was input to the network;
W the matrix of connection weights;
w_j the weights of the connections which feed into unit j;
w_{jk} the weight of the connection from unit j to unit k;
F_j the activation function associated with unit j;
γ_{jk} the learning rate associated with weight w_{jk};
θ the biases to the units;
θ_j the bias input to unit j;
U_j the threshold of unit j in F_j;
E_p the error in the output of the network when input pattern vector p is input;
ε the energy of the network.

10.6.2 Terminology

Output vs. activation of a unit. Since there is no need to do otherwise, we consider the output and the activation value of a unit to be one and the same thing. That is, the output of each neuron equals its activation value.

Bias, offset, threshold: These terms all refer to a constant (i.e., independent of the network input but adapted by the learning rule) term which is input to a unit. They may be used interchangeably, although the latter two terms are often envisaged as a property of the activation function. Furthermore, this external input is usually implemented (and can be written) as a weight from a unit with activation value 1.

Number of layers: In a feed-forward network, the inputs perform no computation and their layer is therefore not counted. Thus a network with one input layer, one hidden layer, and one output layer is referred to as a network with two layers. This convention is widely though not yet universally used.

Representation vs. learning: When using a neural network one has to distinguish two issues which influence the performance of the system. The first one is the representational power of the network, the second one is the learning algorithm.

The representational power of a neural network refers to the ability of a neural network to represent a desired function. Because a neural network is built from a set of standard functions, in most cases the network will only approximate the desired function, and even for an optimal set of weights the approximation error is not zero.

The second issue is the learning algorithm. Given that there exist a set of optimal weights in the network, is there a procedure to (iteratively) find this set of weights?