2.5.1 Feedforward Neural Networks

Feedforward neural networks (FF networks) are the most popular and most widely used models in many practical applications. They are known by many different names, such as "multi-layer perceptrons."

Figure 2.5 illustrates a one-hidden-layer FF network with inputs $x_1,\ldots,x_n$ and output $\hat{y}$. Each arrow in the figure symbolizes a parameter in the network. The network is divided into *layers*. The input layer consists of just the inputs to the network. Then follows a *hidden layer,* which consists of any number of *neurons*, or *hidden units* placed in parallel. Each neuron performs a weighted summation of the inputs, which then passes a nonlinear *activation function* σ, also called the *neuron* function.

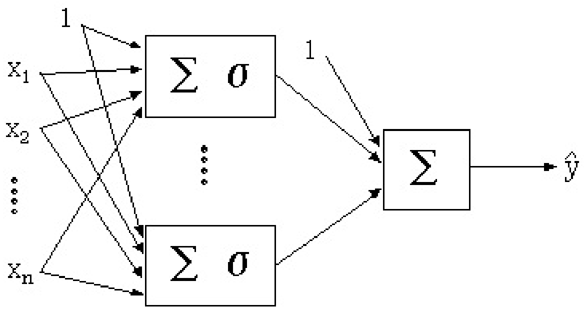


Figure 2.5. A feedforward network with one hidden layer and one output.

Mathematically the functionality of a hidden neuron is described by

$$\sigma\left(\sum_{j=1}^{n} w_j\, x_j + b_j\right)$$

where the weights $\{w_j, b_j\}$ are symbolized with the arrows feeding into the neuron.

The network output is formed by another weighted summation of the outputs of the neurons in the hidden layer. This summation on the output is called the *output layer*. In Figure 2.5 there is only one output in the output layer since it is a single-output problem. Generally, the number of output neurons equals the number of outputs of the approximation problem.

The neurons in the hidden layer of the network in Figure 2.5 are similar in structure to those of the perceptron, with the exception that their activation functions can be any differential function. The output of this network is given by

$$\hat{y}(\theta) = g(\theta, x) = \sum_{i=1}^{nh} w_i^2\, \sigma\left(\sum_{j=1}^{n} w_{i,j}^1\, x_j + b_{j,i}^1\right) + b^2 \qquad (8)$$

where n is the number of inputs and nh is the number of neurons in the hidden layer. The variables $\{w_{i,j}^1, b_{j,i}^1, w_i^2, b^2\}$ are the parameters of the network model that are represented collectively by the parameter vector θ. In general, the neural network model will be represented by the compact notation $g(\theta, x)$ whenever the exact structure of the neural network is not necessary in the context of a discussion.

Some small function approximation examples using an FF network can be found in Section 5.2.

Note that the size of the input and output layers are defined by the number of inputs and outputs of the network and, therefore, only the number of hidden neurons has to be specified when the network is defined. The network in Figure 2.5 is sometimes referred to as a three-layer network, counting input, hidden, and output layers. However, since no processing takes place in the input layer, it is also sometimes called a two-layer network. To avoid confusion this network is called a *one-hidden-layer FF network* throughout this documentation.

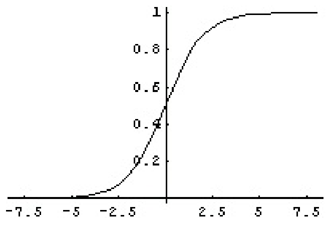
In training the network, its parameters are adjusted incrementally until the training data satisfy the desired mapping as well as possible; that is, until $\hat{y}(\theta)$ matches the desired output y as closely as possible up to a maximum number of iterations. The training process is described in Section 2.5.3, Training Feedforward and Radial Basis Function Networks.

The nonlinear activation function in the neuron is usually chosen to be a smooth step function. The default is the standard sigmoid

$$\text{Sigmoid}[x] = \frac{1}{1 + e^{-x}} \qquad (9)$$

that looks like this.

```
In[1]:= << NeuralNetworks`
        Plot[Sigmoid[x], {x, -8, 8}]
```



The FF network in Figure 2.5 is just one possible architecture of an FF network. You can modify the architecture in various ways by changing the options. For example, you can change the activation function to any differentiable function you want. This is illustrated in Section 13.3.2, The Neuron Function in a Feedforward Network.

Multilayer Networks

The package supports FF neural networks with any number of hidden layers and any number of neurons (hidden neurons) in each layer. In Figure 2.6 a multi-output FF network with two hidden layers is shown.

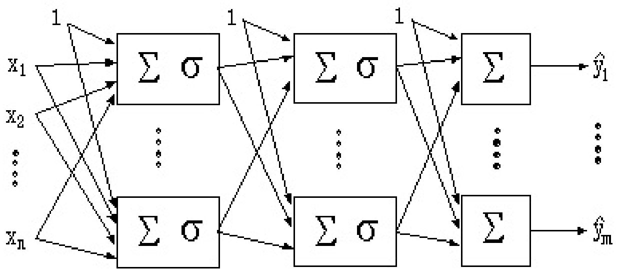


Figure 2.6. A multi-output feedforward network with two hidden layers.

The number of layers and the number of hidden neurons in each hidden layer are user design parameters. The general rule is to choose these design parameters so that the best possible model with as few parameters as possible is obtained. This is, of course, not a very useful rule, and in practice you have to experiment with different designs and compare the results, to find the most suitable neural network model for the problem at hand.

For many practical applications, one or two hidden layers will suffice. The recommendation is to start with a linear model; that is, neural networks with no hidden layers, and then go over to networks with one hidden layer but with no more than five to ten neurons. As a last step you should try two hidden layers.

The output neurons in the FF networks in Figures 2.5 and 2.6 are linear; that is, they do not have any nonlinear activation function after the weighted sum. This is normally the best choice if you have a general function, a time series or a dynamical system that you want to model. However, if you are using the FF network for classification, then it is generally advantageous to use nonlinear output neurons. You can do this by using the option `OutputNonlinearity` when using the built-in functions provided with the *Neural Networks* package, as illustrated in the examples offered in Section 5.3, Classification with Feedforward Networks and Section 12.1, Classification of Paper Quality.