# FUNDAMENTALS OF NEURAL NETWORKS

## ARCHITECTURES, ALGORITHMS, AND APPLICATIONS

Laurene Fausett

Florida Institute of Technology

**Prentice Hall, Upper Saddle River, New Jersey 07458**

## 1.2 WHAT IS A NEURAL NET?

### 1.2.1 Artificial Neural Networks

An *artificial neural network* is an information-processing system that has certain performance characteristics in common with biological neural networks. Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions that:

1. Information processing occurs at many simple elements called neurons.
2. Signals are passed between neurons over connection links.
3. Each connection link has an associated weight, which, in a typical neural net, multiplies the signal transmitted.
4. Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

A neural network is characterized by (1) its pattern of connections between the neurons (called its *architecture*), (2) its method of determining the weights on the connections (called its *training*, or *learning, algorithm*), and (3) its *activation function*.

Since *what* distinguishes (artificial) neural networks from other approaches to information processing provides an introduction to both *how* and *when* to use neural networks, let us consider the defining characteristics of neural networks further.

A neural net consists of a large number of simple processing elements called *neurons, units, cells,* or *nodes*. Each neuron is connected to other neurons by means of directed communication links, each with an associated weight. The weights represent information being used by the net to solve a problem. Neural nets can be applied to a wide variety of problems, such as storing and recalling data or patterns, classifying patterns, performing general mappings from input patterns to output patterns, grouping similar patterns, or finding solutions to constrained optimization problems.

Each neuron has an internal state, called its *activation* or *activity level,* which is a function of the inputs it has received. Typically, a neuron sends its activation as a signal to several other neurons. It is important to note that a neuron can send only one signal at a time, although that signal is broadcast to several other neurons.

For example, consider a neuron $Y$, illustrated in Figure 1.1, that receives inputs from neurons $X_1$, $X_2$, and $X_3$. The activations (output signals) of these neurons are $x_1$, $x_2$, and $x_3$, respectively. The weights on the connections from $X_1$, $X_2$, and $X_3$ to neuron $Y$ are $w_1$, $w_2$, and $w_3$, respectively. The net input, $y\_in$, to neuron $Y$ is the sum of the weighted signals from neurons $X_1$, $X_2$, and $X_3$, i.e.,

$$y\_in = w_1x_1 + w_2x_2 + w_3x_3.$$

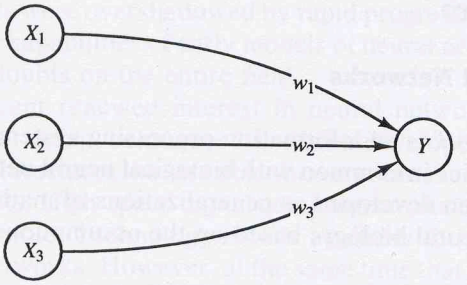



**Figure 1.1** A simple (artificial) neuron.

The activation $y$ of neuron $Y$ is given by some function of its net input, $y = f(y\_in)$, e.g., the logistic sigmoid function (an $S$-shaped curve)

$$f(x) = \frac{1}{1 + \exp(-x)},$$

or any of a number of other activation functions. Several common activation functions are illustrated in Section 1.4.3.

Now suppose further that neuron $Y$ is connected to neurons $Z_1$ and $Z_2$, with weights $v_1$ and $v_2$, respectively, as shown in Figure 1.2. Neuron $Y$ sends its signal $y$ to each of these units. However, in general, the values received by neurons $Z_1$ and $Z_2$ will be different, because each signal is scaled by the appropriate weight, $v_1$ or $v_2$. In a typical net, the activations $z_1$ and $z_2$ of neurons $Z_1$ and $Z_2$ would depend on inputs from several or even many neurons, not just one, as shown in this simple example.

Although the neural network in Figure 1.2 is very simple, the presence of a hidden unit, together with a nonlinear activation function, gives it the ability to solve many more problems than can be solved by a net with only input and output units. On the other hand, it is more difficult to train (i.e., find optimal values for the weights) a net with hidden units. The arrangement of the units (the architecture

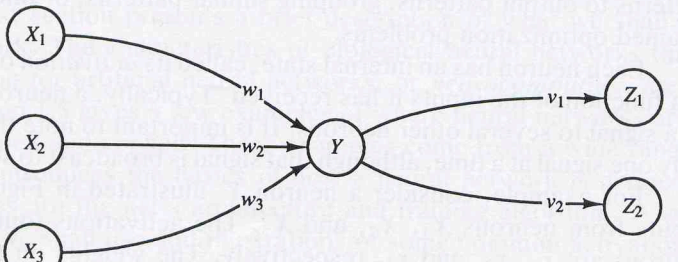



**Figure 1.2** A very simple neural network.

of the net) and the method of training the net are discussed further in Section 1.4. A detailed consideration of these ideas for specific nets, together with simple examples of an application of each net, is the focus of the following chapters.

### 1.2.2 Biological Neural Networks

The extent to which a neural network models a particular biological neural system varies. For some researchers, this is a primary concern; for others, the ability of the net to perform useful tasks (such as approximate or represent a function) is more important than the biological plausibility of the net. Although our interest lies almost exclusively in the computational capabilities of neural networks, we shall present a brief discussion of some features of biological neurons that may help to clarify the most important characteristics of artificial neural networks. In addition to being the original inspiration for artificial nets, biological neural systems suggest features that have distinct computational advantages.

There is a close analogy between the structure of a biological neuron (i.e., a brain or nerve cell) and the processing element (or artificial neuron) presented in the rest of this book. In fact, the structure of an individual neuron varies much less from species to species than does the organization of the system of which the neuron is an element.

A biological neuron has three types of components that are of particular interest in understanding an artificial neuron: its *dendrites, soma,* and *axon*. The many dendrites receive signals from other neurons. The signals are electric impulses that are transmitted across a synaptic gap by means of a chemical process. The action of the chemical transmitter modifies the incoming signal (typically, by scaling the frequency of the signals that are received) in a manner similar to the action of the weights in an artificial neural network.

The soma, or cell body, sums the incoming signals. When sufficient input is received, the cell fires; that is, it transmits a signal over its axon to other cells. It is often supposed that a cell either fires or doesn't at any instant of time, so that transmitted signals can be treated as binary. However, the frequency of firing varies and can be viewed as a signal of either greater or lesser magnitude. This corresponds to looking at discrete time steps and summing all activity (signals received or signals sent) at a particular point in time.

The transmission of the signal from a particular neuron is accomplished by an action potential resulting from differential concentrations of ions on either side of the neuron's axon sheath (the brain's "white matter"). The ions most directly involved are potassium, sodium, and chloride.

A generic biological neuron is illustrated in Figure 1.3, together with axons from two other neurons (from which the illustrated neuron could receive signals) and dendrites for two other neurons (to which the original neuron would send signals). Several key features of the processing elements of artificial neural networks are suggested by the properties of biological neurons, viz., that:

We also illustrate several typical activation functions and conclude the section with a summary of the notation we shall use throughout the rest of the text.

### 1.4.1 Typical Architectures

Often, it is convenient to visualize neurons as arranged in layers. Typically, neurons in the same layer behave in the same manner. Key factors in determining the behavior of a neuron are its activation function and the pattern of weighted connections over which it sends and receives signals. Within each layer, neurons usually have the same activation function and the same pattern of connections to other neurons. To be more specific, in many neural networks, the neurons within a layer are either fully interconnected or not interconnected at all. If any neuron in a layer (for instance, the layer of hidden units) is connected to a neuron in another layer (say, the output layer), then each hidden unit is connected to every output neuron.

The arrangement of neurons into layers and the connection patterns within and between layers is called the *net architecture*. Many neural nets have an input layer in which the activation of each unit is equal to an external input signal. The net illustrated in Figure 1.2 consists of input units, output units, and one hidden unit (a unit that is neither an input unit nor an output unit).

Neural nets are often classified as single layer or multilayer. In determining the number of layers, the input units are not counted as a layer, because they perform no computation. Equivalently, the number of layers in the net can be defined to be the number of layers of weighted interconnect links between the slabs of neurons. This view is motivated by the fact that the weights in a net contain extremely important information. The net shown in Figure 1.2 has two layers of weights.

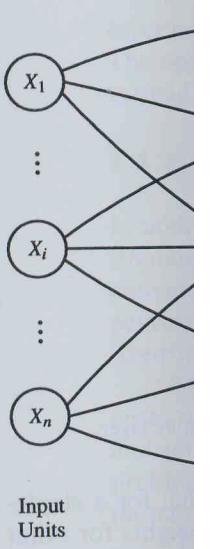
The single-layer and multilayer nets illustrated in Figures 1.4 and 1.5 are examples of *feedforward* nets—nets in which the signals flow from the input units to the output units, in a forward direction. The fully interconnected competitive net in Figure 1.6 is an example of a *recurrent* net, in which there are closed-loop signal paths from a unit back to itself.

#### Single-Layer Net

A single-layer net has one layer of connection weights. Often, the units can be distinguished as input units, which receive signals from the outside world, and output units, from which the response of the net can be read. In the typical single-layer net shown in Figure 1.4, the input units are fully connected to output units but are not connected to other input units, and the output units are not connected to other output units. By contrast, the Hopfield net architecture, shown in Figure 3.7, is an example of a single-layer net in which all units function as both input and output units.

For pattern classification, each output unit corresponds to a particular cat-





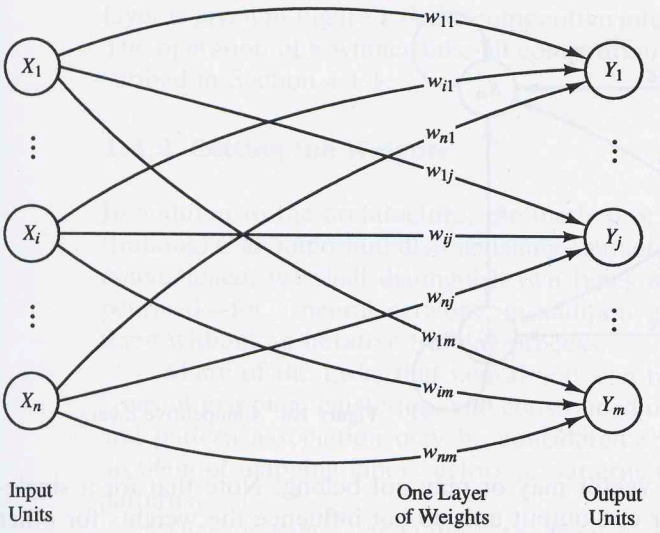



**Figure 1.4** A single-layer neural net.

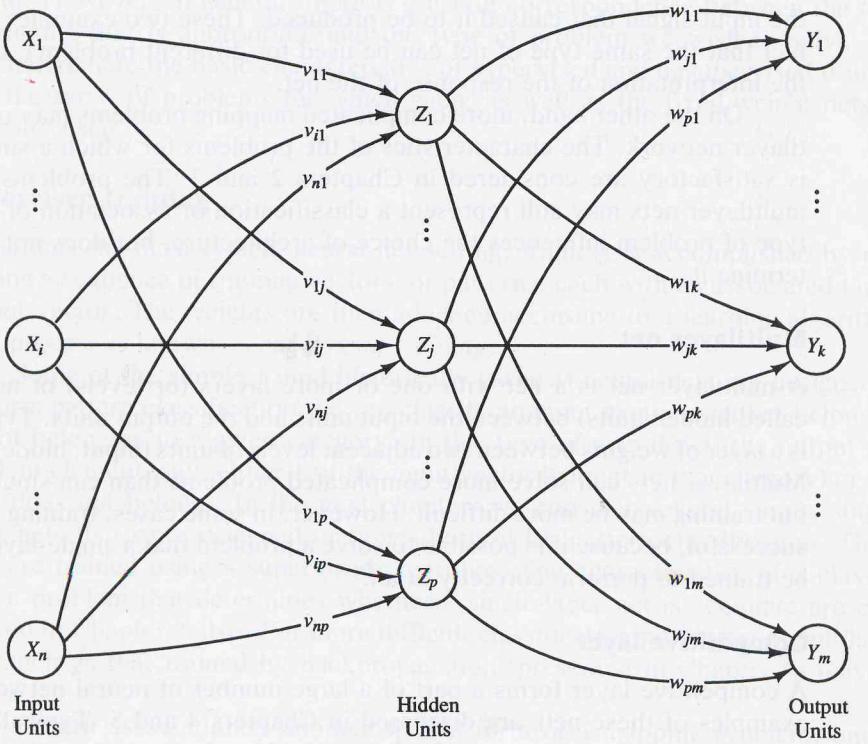



**Figure 1.5** A multilayer neural net.

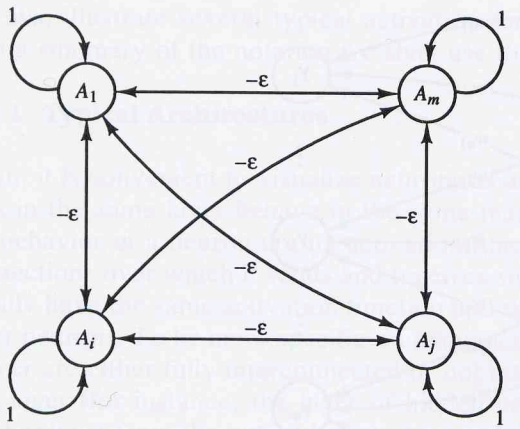



**Figure 1.6** Competitive layer.

egory to which an input vector may or may not belong. Note that for a single-layer net, the weights for one output unit do not influence the weights for other output units. For pattern association, the same architecture can be used, but now the overall pattern of output signals gives the response pattern associated with the input signal that caused it to be produced. These two examples illustrate the fact that the same type of net can be used for different problems, depending on the interpretation of the response of the net.

On the other hand, more complicated mapping problems may require a multilayer network. The characteristics of the problems for which a single-layer net is satisfactory are considered in Chapters 2 and 3. The problems that require multilayer nets may still represent a classification or association of patterns; the type of problem influences the choice of architecture, but does not uniquely determine it.

### Multilayer net

A multilayer net is a net with one or more layers (or levels) of nodes (the so-called hidden units) between the input units and the output units. Typically, there is a layer of weights between two adjacent levels of units (input, hidden, or output). Multilayer nets can solve more complicated problems than can single-layer nets, but training may be more difficult. However, in some cases, training may be more successful, because it is possible to solve a problem that a single-layer net cannot be trained to perform correctly at all.

### Competitive layer

A competitive layer forms a part of a large number of neural networks. Several examples of these nets are discussed in Chapters 4 and 5. Typically, the interconnections between neurons in the competitive layer are not shown in the architecture diagrams for such nets. An example of the architecture for a competitive

layer is given in Figure 1.6; the competitive interconnections have weights of $-\epsilon$. The operation of a winner-take-all competition, MAXNET [Lippman, 1987], is described in Section 4.1.1.

### 1.4.2 Setting the Weights

In addition to the architecture, the method of setting the values of the weights (training) is an important distinguishing characteristic of different neural nets. For convenience, we shall distinguish two types of training—supervised and unsupervised—for a neural network; in addition, there are nets whose weights are fixed without an iterative training process.

Many of the tasks that neural nets can be trained to perform fall into the areas of mapping, clustering, and constrained optimization. Pattern classification and pattern association may be considered special forms of the more general problem of mapping input vectors or patterns to the specified output vectors or patterns.

There is some ambiguity in the labeling of training methods as supervised or unsupervised, and some authors find a third category, self-supervised training, useful. However, in general, there is a useful correspondence between the type of training that is appropriate and the type of problem we wish to solve. We summarize here the basic characteristics of supervised and unsupervised training and the types of problems for which each, as well as the fixed-weight nets, is typically used.

#### Supervised training

In perhaps the most typical neural net setting, training is accomplished by presenting a sequence of training vectors, or patterns, each with an associated target output vector. The weights are then adjusted according to a learning algorithm. This process is known as *supervised training*.

Some of the simplest (and historically earliest) neural nets are designed to perform pattern classification, i.e., to classify an input vector as either belonging or not belonging to a given category. In this type of neural net, the output is a bivalent element, say, either 1 (if the input vector belongs to the category) or $-1$ (if it does not belong). In the next chapter, we consider several simple single-layer nets that were designed or typically used for pattern classification. These nets are trained using a supervised algorithm. The characteristics of a classification problem that determines whether a single-layer net is adequate are considered in Chapter 2 also. For more difficult classification problems, a multilayer net, such as that trained by backpropagation (presented in Chapter 6) may be better.

Pattern association is another special form of a mapping problem, one in which the desired output is not just a "yes" or "no," but rather a pattern. A neural net that is trained to associate a set of input vectors with a corresponding

set of output vectors is called an *associative memory*. If the desired output vector is the same as the input vector, the net is an *autoassociative memory*; if the output target vector is different from the input vector, the net is a *heteroassociative memory*. After training, an associative memory can recall a stored pattern when it is given an input vector that is sufficiently similar to a vector it has learned. Associative memory neural nets, both feedforward and recurrent, are discussed in Chapter 3.

Multilayer neural nets can be trained to perform a nonlinear mapping from an $n$-dimensional space of input vectors ($n$-tuples) to an $m$-dimensional output space—i.e., the output vectors are $m$-tuples.

The single-layer nets in Chapter 2 (pattern classification nets) and Chapter 3 (pattern association nets) use supervised training (the Hebb rule or the delta rule). Backpropagation (the generalized delta rule) is used to train the multilayer nets in Chapter 6. Other forms of supervised learning are used for some of the nets in Chapter 4 (learning vector quantization and counterpropagation) and Chapter 7. Each learning algorithm will be described in detail, along with a description of the net for which it is used.

### Unsupervised training

Self-organizing neural nets group similar input vectors together without the use of training data to specify what a typical member of each group looks like or to which group each vector belongs. A sequence of input vectors is provided, but no target vectors are specified. The net modifies the weights so that the most similar input vectors are assigned to the same output (or cluster) unit. The neural net will produce an exemplar (representative) vector for each cluster formed. Self-organizing nets are described in Chapters 4 (Kohonen self-organizing maps) and Chapter 5 (adaptive resonance theory).

Unsupervised learning is also used for other tasks, in addition to clustering. Examples are included in Chapter 7.

### Fixed-weight nets

Still other types of neural nets can solve constrained optimization problems. Such nets may work well for problems that can cause difficulty for traditional techniques, such as problems with conflicting constraints (i.e., not all constraints can be satisfied simultaneously). Often, in such cases, a nearly optimal solution (which the net can find) is satisfactory. When these nets are designed, the weights are set to represent the constraints and the quantity to be maximized or minimized. The Boltzmann machine (without learning) and the continuous Hopfield net (Chapter 7) can be used for constrained optimization problems.

Fixed weights are also used in contrast-enhancing nets (see Section 4.1).

### 1.4.3 Common Activation Functions

As mentioned before, the basic operation of an artificial neuron involves summing its weighted input signal and applying an output, or activation, function. For the input units, this function is the identity function (see Figure 1.7). Typically, the same activation function is used for all neurons in any particular layer of a neural net, although this is not required. In most cases, a nonlinear activation function is used. In order to achieve the advantages of multilayer nets, compared with the limited capabilities of single-layer nets, nonlinear functions are required (since the results of feeding a signal through two or more layers of linear processing elements—i.e., elements with linear activation functions—are no different from what can be obtained using a single layer).

(*i*) Identity function:

$$f(x) = x \qquad \text{for all } x.$$

Single-layer nets often use a step function to convert the net input, which is a continuously valued variable, to an output unit that is a binary (1 or 0) or bipolar (1 or $-1$) signal (see Figure 1.8). The use of a *threshold* in this regard is discussed in Section 2.1.2. The binary step function is also known as the threshold function or Heaviside function.

(*ii*) Binary step function (with threshold $\theta$):

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases}$$

Sigmoid functions (*S*-shaped curves) are useful activation functions. The logistic function and the hyperbolic tangent functions are the most common. They are especially advantageous for use in neural nets trained by backpropagation, because the simple relationship between the value of the function at a point and the value of the derivative at that point reduces the computational burden during training.

The logistic function, a sigmoid function with range from 0 to 1, is often

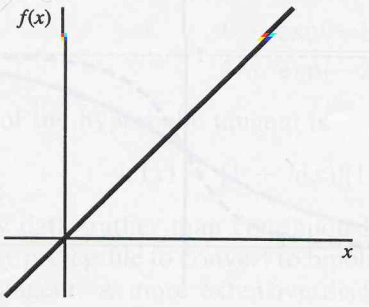



**Figure 1.7** Identity function.

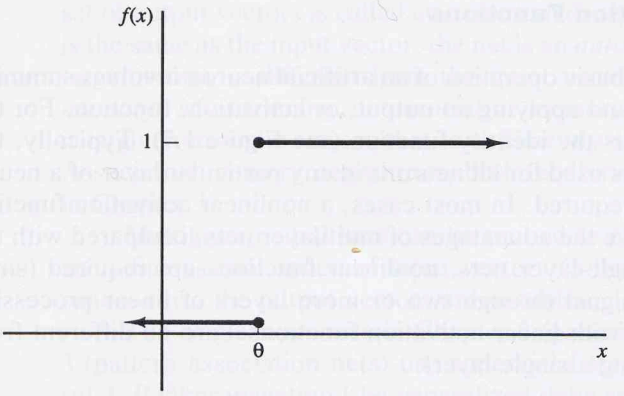



**Figure 1.8** Binary step function.

used as the activation function for neural nets in which the desired output values either are binary or are in the interval between 0 and 1. To emphasize the range of the function, we will call it the *binary sigmoid*; it is also called the *logistic sigmoid*. This function is illustrated in Figure 1.9 for two values of the *steepness parameter* $\sigma$.

(*iii*) Binary sigmoid:

$$f(x) = \frac{1}{1 + \exp(-\sigma x)}.$$

$$f'(x) = \sigma f(x)\,[1 - f(x)].$$

As is shown in Section 6.2.3, the logistic sigmoid function can be scaled to have any range of values that is appropriate for a given problem. The most common range is from $-1$ to 1; we call this sigmoid the *bipolar sigmoid*. It is illustrated in Figure 1.10 for $\sigma = 1$.

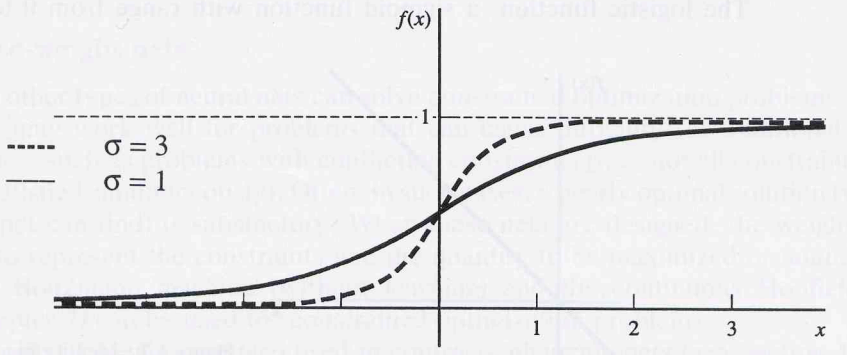



**Figure 1.9** Binary sigmoid. Steepness parameters $\sigma = 1$ and $\sigma = 3$.

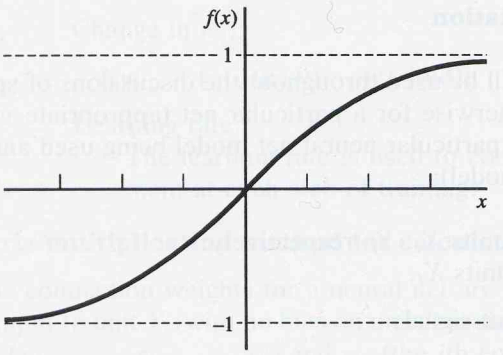



**Figure 1.10** Bipolar sigmoid.

(*iv*) Bipolar sigmoid:

$$g(x) = 2f(x) - 1 = \frac{2}{1 + \exp(-\sigma x)} - 1$$

$$= \frac{1 - \exp(-\sigma x)}{1 + \exp(-\sigma x)}.$$

$$g'(x) = \frac{\sigma}{2}[1 + g(x)][1 - g(x)].$$

The bipolar sigmoid is closely related to the hyperbolic tangent function, which is also often used as the activation function when the desired range of output values is between $-1$ and 1. We illustrate the correspondence between the two for $\sigma = 1$. We have

$$g(x) = \frac{1 - \exp(-x)}{1 + \exp(-x)}.$$

The hyperbolic tangent is

$$h(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

$$= \frac{1 - \exp(-2x)}{1 + \exp(-2x)}.$$

The derivative of the hyperbolic tangent is

$$h'(x) = [1 + h(x)][1 - h(x)].$$

For binary data (rather than continuously valued data in the range from 0 to 1), it is usually preferable to convert to bipolar form and use the bipolar sigmoid or hyperbolic tangent. A more extensive discussion of the choice of activation functions and different forms of sigmoid functions is given in Section 6.2.2.

### 1.4.4 Summary of Notation

The following notation will be used throughout the discussions of specific neural nets, unless indicated otherwise for a particular net (appropriate values for the parameter depend on the particular neural net model being used and will be discussed further for each model):

$x_i$, $y_j$ Activations of units $X_i$, $Y_j$, respectively:
For input units $X_i$,

$$x_i = \text{input signal};$$

for other units $Y_j$,

$$y_j = f(y\_in_j).$$

$w_{ij}$ Weight on connection from unit $X_i$ to unit $Y_j$:
Beware: Some authors use the opposite convention, with $w_{ij}$ denoting the weight from unit $Y_j$ to unit $X_i$.

$b_j$ Bias on unit $Y_j$:
A bias acts like a weight on a connection from a unit with a constant activation of 1 (see Figure 1.11).

$y\_in_j$ Net input to unit $Y_j$:

$$y\_in_j = b_j + \sum_i x_i w_{ij}$$

$W$ Weight matrix:

$$W = \{w_{ij}\}.$$

$\mathbf{w}_{\cdot j}$ Vector of weights:

$$\mathbf{w}_{\cdot j} = (w_{1j}, w_{2j}, \ldots, w_{nj})^T.$$

This is the $j$th column of the weight matrix.

$\| \mathbf{x} \|$ Norm or magnitude of vector $\mathbf{x}$.

$\theta_j$ Threshold for activation of neuron $Y_j$:
A step activation function sets the activation of a neuron to 1 whenever its net input is greater than the specified threshold value $\theta_j$; otherwise its activation is 0 (see Figure 1.8).

$\mathbf{s}$ Training input vector:

$$\mathbf{s} = (s_1, \ldots, s_i, \ldots, s_n).$$

$\mathbf{t}$ Training (or target) output vector:

$$\mathbf{t} = (t_1, \ldots, t_j, \ldots, t_m).$$

$\mathbf{x}$ Input vector (for the net to classify or respond to):

$$\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n).$$

$\Delta w_{ij}$ Change in $w_{ij}$:

$$\Delta w_{ij} = [w_{ij}\,(\text{new}) - w_{ij}\,(\text{old})].$$

$\alpha$ Learning rate:

The learning rate is used to control the amount of weight adjustment at each step of training.

**Matrix multiplication method for calculating net input**

If the connection weights for a neural net are stored in a matrix $\mathbf{W} = (w_{ij})$, the net input to unit $Y_j$ (with no bias on unit $j$) is simply the dot product of the vectors $\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)$ and $\mathbf{w}_{\cdot j}$ (the $j$th column of the weight matrix):

$$y\_in_j = \mathbf{x} \cdot \mathbf{w}_{\cdot j}$$

$$= \sum_{i=1}^{n} x_i w_{ij}\,.$$

**Bias**

A bias can be included by adding a component $x_0 = 1$ to the vector $\mathbf{x}$, i.e., $\mathbf{x} = (1, x_1, \ldots, x_i, \ldots, x_n)$. The bias is treated exactly like any other weight, i.e., $w_{0j} = b_j$. The net input to unit $Y_j$ is given by

$$y\_in_j = \mathbf{x} \cdot \mathbf{w}_{\cdot j}$$

$$= \sum_{i=0}^{n} x_i w_{ij}$$

$$= w_{0j} + \sum_{i=1}^{n} x_i w_{ij}$$

$$= b_j + \sum_{i=1}^{n} x_i w_{ij}\,.$$

The relation between a bias and a threshold is considered in Section 2.1.2.

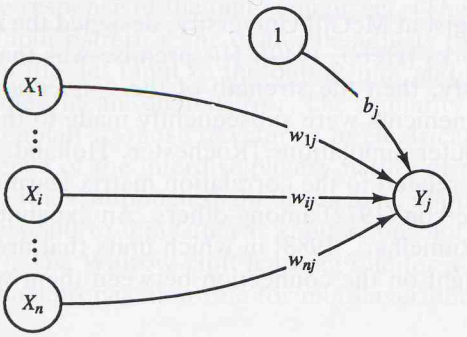



**Figure 1.11** Neuron with a bias.