FUNDAMENTALS OF NEURAL NETWORKS

ARCHITECTURES, ALGORITHMS, AND APPLICATIONS

Laurene Fausett

Florida Institute of Technology

Prentice Hall, Upper Saddle River, New Jersey 07458

CHAPTER 2

Simple Neural Nets for Pattern Classification

2.1 GENERAL DISCUSSION

One of the simplest tasks that neural nets can be trained to perform is pattern classification. In pattern classification problems, each input vector (pattern) belongs, or does not belong, to a particular class or category. For a neural net approach, we assume we have a set of training patterns for which the correct classification is known. In the simplest case, we consider the question of membership in a single class. The output unit represents membership in the class with a response of 1; a response of -1 (or 0 if binary representation is used) indicates that the pattern is not a member of the class. For the single-layer nets described in this chapter, extension to the more general case in which each pattern may or may not belong to any of several classes is immediate. In such case, there is an output unit for each class. Pattern classification is one type of pattern recognition; the associative recall of patterns (discussed in Chapter 3) is another.

Pattern classification problems arise in many areas. In 1963, Donald Specht (a student of Widrow) used neural networks to detect heart abnormalities with EKG types of data as input (46 measurements). The output was "normal" or "abnormal," with an "on" response signifying normal [Specht, 1967; Caudill & Butler, 1990]. In the early 1960s, Minsky and Papert used neural nets to classify input patterns as convex or not convex and connected or not connected [Minsky & Papert, 1988]. There are many other examples of pattern classification problems

being solved by neural networks, both the simple nets described in this chapter, other early nets not discussed here, and multilayer nets (especially the backpropagation nets described in Chapter 6).

In this chapter, we shall discuss three methods of training a simple single-layer neural net for pattern classification: the Hebb rule, the perceptron learning rule, and the delta rule (used by Widrow in his ADALINE neural net). First, however, we discuss some issues that are common to all single-layer nets that perform pattern classification. We conclude the chapter with some comparisons of the nets discussed and an extension to a multilayer net, MADALINE.

Many real-world examples need more sophisticated architecture and training rules because the conditions for a single-layer net to be adequate (see Section 2.1.3) are not met. However, if the conditions are met approximately, the results may be sufficiently accurate. Also, insight can be gained from the more simple nets, since the meaning of the weights may be easier to interpret.

2.1.1 Architecture

The basic architecture of the simplest possible neural networks that perform pattern classification consists of a layer of input units (as many units as the patterns to be classified have components) and a single output unit. Most neural nets we shall consider in this chapter use the single-layer architecture shown in Figure 2.1. This allows classification of vectors, which are n-tuples, but considers membership in only one category.

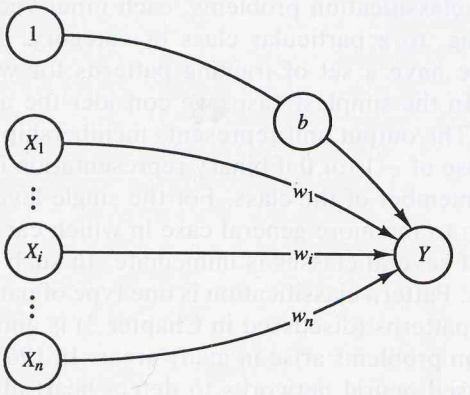

Figure 2.1 Single-layer net for pattern classification.

An example of a net that classifies the input into several categories is considered in Section 2.3.3. This net is a simple extension of the nets that perform

a single classification. The MADALINE net considered in Section 2.4.5 is a multilayer extension of the single-layer ADALINE net.

2.1.2 Biases and Thresholds

A bias acts exactly as a weight on a connection from a unit whose activation is always 1. Increasing the bias increases the net input to the unit. If a bias is included, the activation function is typically taken to be

$$f(\text{net}) = \begin{cases} 1 & \text{if net} \geq 0; \\ -1 & \text{if net} < 0; \end{cases}$$

where

$$\text{net} = b + \sum_i x_i w_i.$$

Some authors do not use a bias weight, but instead use a fixed threshold θ for the activation function. In that case,

$$f(\text{net}) = \begin{cases} 1 & \text{if net} \geq \theta; \\ -1 & \text{if net} < \theta; \end{cases}$$

where

$$\text{net} = \sum_i x_i w_i.$$

However, as the next example will demonstrate, this is essentially equivalent to the use of an adjustable bias.

Example 2.1 The role of a bias or a threshold

In this example and in the next section, we consider the separation of the input space into regions where the response of the net is positive and regions where the response is negative. To facilitate a graphical display of the relationships, we illustrate the ideas for an input with two components while the output is a scalar (i.e., it has only one component). The architecture of these examples is given in Figure 2.2.

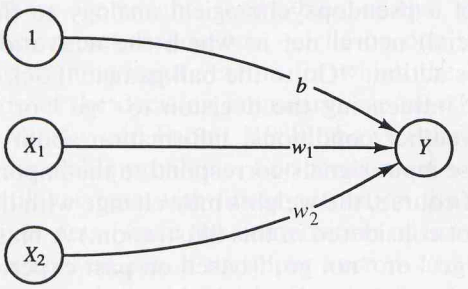

Figure 2.2 Single-layer neural network for logic functions.

The boundary between the values of x_1 and x_2 for which the net gives a positive response and the values for which it gives a negative response is the separating line

$$b + x_1 w_1 + x_2 w_2 = 0,$$

or (assuming that $w_2 \neq 0$),

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}.$$

The requirement for a positive response from the output unit is that the net input it receives, namely, $b + x_1 w_1 + x_2 w_2$, be greater than 0. During training, values of w_1, w_2, and b are determined so that the net will have the correct response for the training data.

If one thinks in terms of a threshold, the requirement for a positive response from the output unit is that the net input it receives, namely, $x_1 w_1 + x_2 w_2$, be greater than the threshold. This gives the equation of the line separating positive from negative output as

$$x_1 w_1 + x_2 w_2 = \theta,$$

or (assuming that $w_2 \neq 0$),

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}.$$

During training, values of w_1 and w_2 are determined so that the net will have the correct response for the training data. In this case, the separating line cannot pass through the origin, but a line can be found that passes arbitrarily close to the origin.

The form of the separating line found by using an adjustable bias and the form obtained by using a fixed threshold illustrate that there is no advantage to including both a bias and a nonzero threshold for a neuron that uses the step function as its activation function. On the other hand, including neither a bias nor a threshold is equivalent to requiring the separating line (or plane or hyperplane for inputs with more components) to pass through the origin. This may or may not be appropriate for a particular problem.

As an illustration of a pseudopsychological analogy to the use of a bias, consider a simple (artificial) neural net in which the activation of the neuron corresponds to a person's action, "Go to the ball game." Each input signal corresponds to some factor influencing the decision to "go" or "not go" (other possible activities, the weather conditions, information about who is pitching, etc.). The weights on these input signals correspond to the importance the person places on each factor. (Of course, the weights may change with time, but methods for modifying them are not considered in this illustration.) A bias could represent a general inclination to "go" or "not go," based on past experiences. Thus, the bias would be modifiable, but the signal to it would not correspond to information about the specific game in question or activities competing for the person's time.

The threshold for this "decision neuron" indicates the total net input necessary to cause the person to "go," i.e., for the decision neuron to fire. The threshold would be different for different people; however, for the sake of this simple example, it should be thought of as a quantity that remains fixed for each individual. Since it is the relative values of the weights, rather than their actual magnitudes, that determine the response of the neuron, the model can cover all possibilities using either the fixed threshold or the adjustable bias.

2.1.3 Linear Separability

For each of the nets in this chapter, the intent is to train the net (i.e., adaptively determine its weights) so that it will respond with the desired classification when presented with an input pattern that it was trained on or when presented with one that is sufficiently similar to one of the training patterns. Before discussing the particular nets (which is to say, the particular styles of training), it is useful to discuss some issues common to all of the nets. For a particular output unit, the desired response is a "yes" if the input pattern is a member of its class and a "no" if it is not. A "yes" response is represented by an output signal of 1, a "no" by an output signal of -1 (for bipolar signals). Since we want one of two responses, the activation (or transfer or output) function is taken to be a step function. The value of the function is 1 if the net input is positive and -1 if the net input is negative. Since the net input to the output unit is

$$y\_in = b + \sum_i x_i w_i,$$

it is easy to see that the boundary between the region where $y\_in > 0$ and the region where $y\_in < 0$, which we call the *decision boundary*, is determined by the relation

$$b + \sum_i x_i w_i = 0.$$

Depending on the number of input units in the network, this equation represents a line, a plane, or a hyperplane.

If there are weights (and a bias) so that all of the training input vectors for which the correct response is $+1$ lie on one side of the decision boundary and all of the training input vectors for which the correct response is -1 lie on the other side of the decision boundary, we say that the problem is "linearly separable." Minsky and Papert [1988] showed that a single-layer net can learn only linearly separable problems. Furthermore, it is easy to extend this result to show that multilayer nets with linear activation functions are no more powerful than single-layer nets (since the composition of linear functions is linear).

It is convenient, if the input vectors are ordered pairs (or at most ordered triples), to graph the input training vectors and indicate the desired response by the appropriate symbol ("+" or "−"). The analysis also extends easily to nets

with more input units; however, the graphical display is not as convenient. The region where y is positive is separated from the region where it is negative by the line

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}.$$

These two regions are often called *decision regions* for the net. Notice in the following examples that there are many different lines that will serve to separate the input points that have different target values. However, for any particular line, there are also many choices of w_1, w_2, and b that give exactly the same line. The choice of the sign for b determines which side of the separating line corresponds to a $+1$ response and which side to a -1 response.

There are four different bipolar input patterns we can use to train a net with two input units. However, there are two possible responses for each input pattern, so there are 2^4 different functions that we might be able to train a very simple net to perform. Several of these functions are familiar from elementary logic, and we will use them for illustrations, for convenience. The first question we consider is, For this very simple net, do weights exist so that the net will have the desired output for each of the training input vectors?

Example 2.2 Response regions for the AND function

The AND function (for bipolar inputs and target) is defined as follows:

INPUT (x_1, x_2)	OUTPUT (t)
(1, 1)	+1
(1, −1)	−1
(−1, 1)	−1
(−1, −1)	−1

The desired responses can be illustrated as shown in Figure 2.3. One possible decision boundary for this function is shown in Figure 2.4.

An example of weights that would give the decision boundary illustrated in the figure, namely, the separating line

$$x_2 = -x_1 + 1,$$

is

$$b = -1,$$
$$w_1 = 1,$$
$$w_2 = 1.$$

The choice of sign for b is determined by the requirement that

$$b + x_1 w_1 + x_2 w_2 < 0$$

where $x_1 = 0$ and $x_2 = 0$. (Any point that is not on the decision boundary can be used to determine which side of the boundary is positive and which is negative; the origin is particularly convenient to use when it is not on the boundary.)

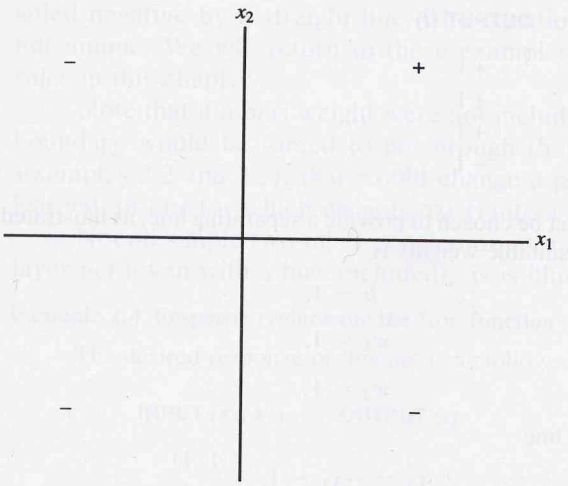

Figure 2.3 Desired response for the logic function AND (for bipolar inputs).

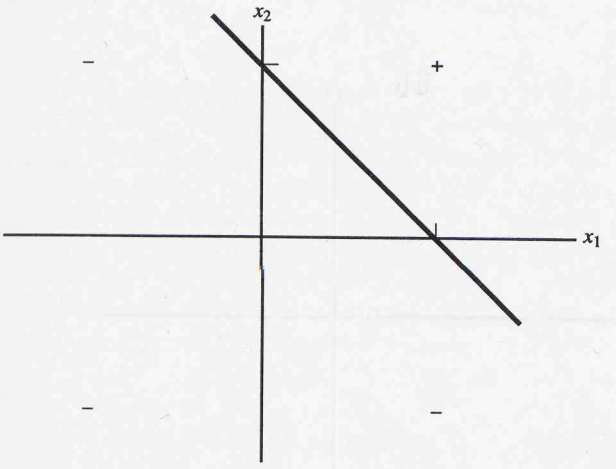

Figure 2.4 The logic function AND, showing a possible decision boundary.

Example 2.3 Response regions for the OR function

The logical OR function (for bipolar inputs and target) is defined as follows:

INPUT (x_1, x_2)	OUTPUT (t)
(1, 1)	+1
(1, −1)	+1
(−1, 1)	+1
(−1, −1)	−1

The weights must be chosen to provide a separating line, as illustrated in Figure 2.5. One example of suitable weights is

$$b = 1,$$
$$w_1 = 1,$$
$$w_2 = 1,$$

giving the separating line

$$x_2 = -x_1 - 1.$$

The choice of sign for b is determined by the requirement that

$$b + x_1 w_1 + x_2 w_2 > 0$$

where $x_1 = 0$ and $x_2 = 0$.

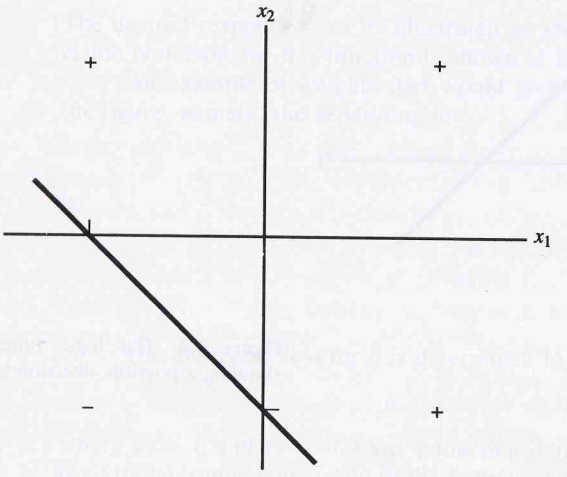

Figure 2.5 The logic function OR, showing a possible decision boundary.

The preceding two mappings (which can each be solved by a single-layer neural net) illustrate graphically the concept of linearly separable input. The input points to be classified positive can be separated from the input points to be classified negative by a straight line. The equations of the decision boundaries are not unique. We will return to these examples to illustrate each of the learning rules in this chapter.

Note that if a bias weight were not included in these examples, the decision boundary would be forced to go through the origin. In many cases (including Examples 2.2 and 2.3), that would change a problem that could be solved (i.e., learned, or one for which weights exist) into a problem that could not be solved.

Not all simple two-input, single-output mappings can be solved by a single-layer net (even with a bias included), as is illustrated in Example 2.4.

Example 2.4 Response regions for the XOR function

The desired response of this net is as follows:

INPUT (x_1, x_2)	OUTPUT (t)
(1, 1)	−1
(1, −1)	+1
(−1, 1)	+1
(−1, −1)	−1

It is easy to see that no single straight line can separate the points for which a positive response is desired from those for which a negative response is desired.

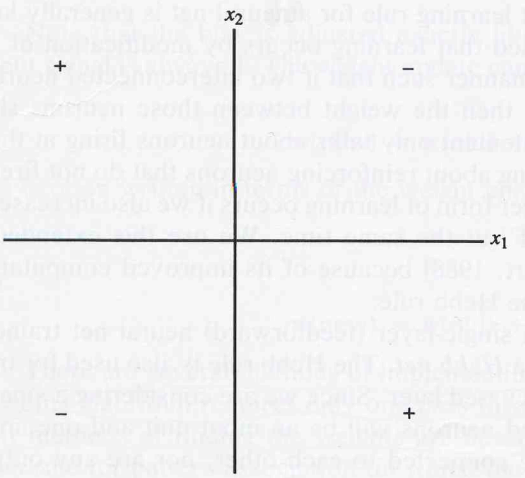

Figure 2.6 Desired response for the logic function XOR.