FUNDAMENTALS OF NEURAL NETWORKS

ARCHITECTURES, ALGORITHMS, AND APPLICATIONS

Laurene Fausett

Florida Institute of Technology

Prentice Hall, Upper Saddle River, New Jersey 07458

2.3 PERCEPTRON

Perceptrons had perhaps the most far-reaching impact of any of the early neural nets. The perceptron learning rule is a more powerful learning rule than the Hebb rule. Under suitable assumptions, its iterative learning procedure can be proved to converge to the correct weights, i.e., the weights that allow the net to produce the correct output value for each of the training input patterns. Not too surprisingly, one of the necessary assumptions is that such weights exist.

A number of different types of perceptrons are described in Rosenblatt (1962) and in Minsky and Papert (1969, 1988). Although some perceptrons were self-organizing, most were trained. Typically, the original perceptrons had three layers of neurons—sensory units, associator units, and a response unit—forming an approximate model of a retina. One particular simple perceptron [Block, 1962] used binary activations for the sensory and associator units and an activation of $+1$, 0, or -1 for the response unit. The sensory units were connected to the associator units by connections with fixed weights having values of $+1$, 0, or -1, assigned at random.

The activation function for each associator unit was the binary step function with an arbitrary, but fixed, threshold. Thus, the signal sent from the associator units to the output unit was a binary (0 or 1) signal. The output of the perceptron is $y = f(y\_in)$, where the activation function is

$$f(y\_in) = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \leq y\_in \leq \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

The weights from the associator units to the response (or output) unit were adjusted by the perceptron learning rule. For each training input, the net would calculate the response of the output unit. Then the net would determine whether an error occurred for this pattern (by comparing the calculated output with the target value). The net did not distinguish between an error in which the calculated output was zero and the target -1, as opposed to an error in which the calculated output was $+1$ and the target -1. In either of these cases, the sign of the error denotes that the weights should be changed in the direction indicated by the target value. However, only the weights on the connections from units that sent a non-zero signal to the output unit would be adjusted (since only these signals contributed to the error). If an error occurred for a particular training input pattern, the weights would be changed according to the formula

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i,$$

where the target value t is $+1$ or -1 and α is the learning rate. If an error did not occur, the weights would not be changed.

Training would continue until no error occurred. The perceptron learning rule convergence theorem states that if weights exist to allow the net to respond correctly to all training patterns, then the rule's procedure for adjusting the weights will find values such that the net does respond correctly to all training patterns (i.e., the net solves the problem or learns the classification). Moreover, the net will find these weights in a finite number of training steps. We will consider a proof of this theorem in Section 2.3.4, since it helps clarify which aspects of the many variations on perceptron learning are significant.

2.3.1 Architecture

Simple perceptron for pattern classification

The output from the associator units in the original simple perceptron was a binary vector; that vector is treated as the input signal to the output unit in the sections that follow. As the proof of the perceptron learning rule convergence theorem given in Section 2.3.4 illustrates, the assumption of a binary input is not necessary. Since only the weights from the associator units to the output unit could be adjusted, we limit our consideration to the single-layer portion of the net, shown in Figure 2.14. Thus, the associator units function like input units, and the architecture is as given in the figure.

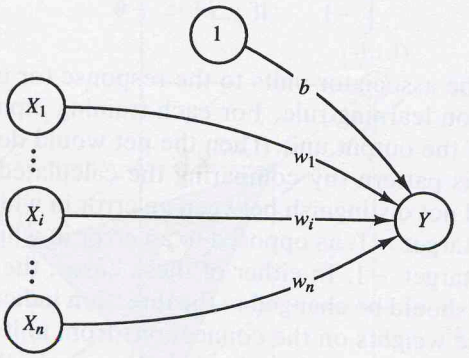

Figure 2.14 Perceptron to perform single classification.

The goal of the net is to classify each input pattern as belonging, or not belonging, to a particular class. Belonging is signified by the output unit giving a response of $+1$; not belonging is indicated by a response of -1. The net is trained to perform this classification by the iterative technique described earlier and given in the algorithm that follows.

2.3.2 Algorithm

The algorithm given here is suitable for either binary or bipolar input vectors (n-tuples), with a bipolar target, fixed θ, and adjustable bias. The threshold θ does not play the same role as in the step function illustrated in Section 2.1.2; thus, both a bias and a threshold are needed. The role of the threshold is discussed following the presentation of the algorithm. The algorithm is not particularly sensitive to the initial values of the weights or the value of the learning rate.

Step 0. Initialize weights and bias.
(For simplicity, set weights and bias to zero.)
Set learning rate α ($0 < \alpha \leq 1$).
(For simplicity, α can be set to 1.)

Step 1. While stopping condition is false, do Steps 2–6.

Step 2. For each training pair s:t, do Steps 3–5.

Step 3. Set activations of input units:

$$x_i = s_i.$$

Step 4. Compute response of output unit:

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

$$y = \begin{cases} 1 & \text{if } y\_in > \theta \\ 0 & \text{if } -\theta \leq y\_in \leq \theta \\ -1 & \text{if } y\_in < -\theta \end{cases}$$

Step 5. Update weights and bias if an error occurred for this pattern.

If $y \neq t$,

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i,$$

$$b(\text{new}) = b(\text{old}) + \alpha t.$$

else

$$w_i(\text{new}) = w_i(\text{old}),$$

$$b(\text{new}) = b(\text{old}).$$

Step 6. Test stopping condition:
If no weights changed in Step 2, stop; else, continue.

Note that only weights connecting active input units ($x_i \neq 0$) are updated. Also, weights are updated only for patterns that do not produce the correct value of y. This means that as more training patterns produce the correct response, less learning occurs. This is in contrast to the training of the ADALINE units described in Section 2.4, in which learning is based on the difference between y_in and t.

The threshold on the activation function for the response unit is a fixed, non-negative value θ. The form of the activation function for the output unit (response unit) is such that there is an "undecided" band (of fixed width determined by θ) separating the region of positive response from that of negative response. Thus, the previous analysis of the interchangeability of bias and threshold does not apply, because changing θ would change the width of the band, not just the position.

Note that instead of one separating line, we have a line separating the region of positive response from the region of zero response, namely, the line bounding the inequality

$$w_1x_1 + w_2x_2 + b > \theta,$$

and a line separating the region of zero response from the region of negative response, namely, the line bounding the inequality

$$w_1x_1 + w_2x_2 + b < -\theta.$$

2.3.3 Application

Logic functions

Example 2.11 A Perceptron for the AND function: binary inputs, bipolar targets

Let us consider again the AND function with binary input and bipolar target, now using the perceptron learning rule. The training data are as given in Example 2.6 for the Hebb rule. An adjustable bias is included, since it is necessary if a single-layer net is to be able to solve this problem. For simplicity, we take $\alpha = 1$ and set the initial weights and bias to 0, as indicated. However, to illustrate the role of the threshold, we take $\theta = .2$.

The weight change is $\Delta \mathbf{w} = t(x_1, x_2, 1)$ if an error has occurred and zero otherwise. Presenting the first input, we have:

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	1)							$(w_1$	w_2	$b)$
									(0	0	0)
(1	1	1)	0	0	1	(1	1	1)	(1	1	1)

The separating lines become

$$x_1 + x_2 + 1 = .2$$

and

$$x_1 + x_2 + 1 = -.2.$$

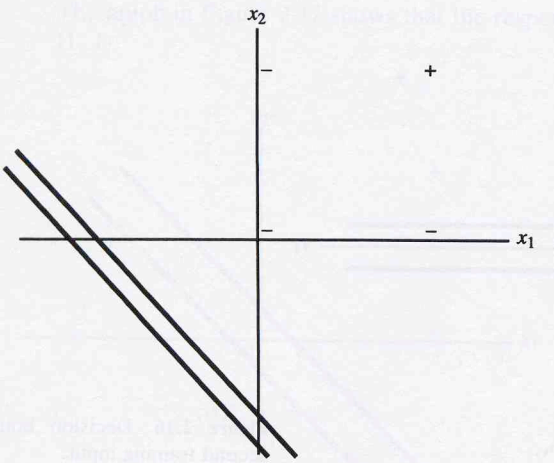

Figure 2.15 Decision boundary for logic function AND after first training input.

The graph in Figure 2.15 shows that the response of the net will now be correct for the first input pattern.

Presenting the second input yields the following:

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	1)							$(w_1$	w_2	$b)$
									(1	1	1)
(1	0	1)	2	1	−1	(−1	0	−1)	(0	1	0)

The separating lines become

$$x_2 = .2$$

and

$$x_2 = -.2$$

The graph in Figure 2.16 shows that the response of the net will now (still) be correct for the first input point.

For the third input, we have:

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
$(x_1$	x_2	1)							$(w_1$	w_2	$b)$
									(0	1	0)
(0	1	1)	1	1	−1	(0	−1	−1)	(0	0	−1)

Since the components of the input patterns are nonnegative and the components of the weight vector are nonpositive, the response of the net will be negative (or zero).

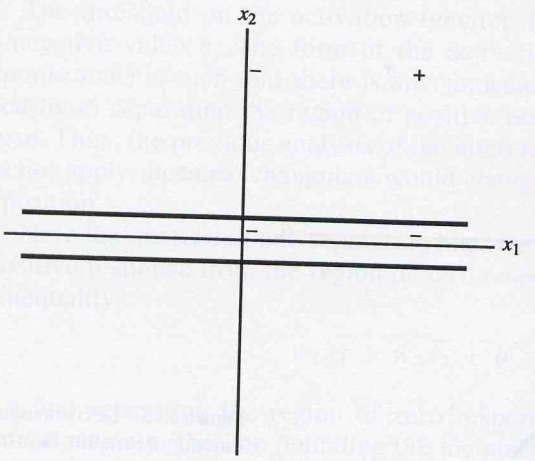

Figure 2.16 Decision boundary after second training input.

To complete the first epoch of training, we present the fourth training pattern:

INPUT	NET	OUT	TARGET	WEIGHT CHANGES	WEIGHTS
$(x_1 \quad x_2 \quad 1)$					$(w_1 \quad w_2 \quad b)$
					(0 0 −1)
(0 0 1)	−1	−1	−1	(0 0 0)	(0 0 −1)

The response for all of the input patterns is negative for the weights derived; but since the response for input pattern (1, 1) is not correct, we are not finished.

The second epoch of training yields the following weight updates for the first input:

INPUT	NET	OUT	TARGET	WEIGHT CHANGES	WEIGHTS
$(x_1 \quad x_2 \quad 1)$					$(w_1 \quad w_2 \quad b)$
					(0 0 −1)
(1 1 1)	−1	−1	1	(1 1 1)	(1 1 0)

The separating lines become

$$x_1 + x_2 = .2$$

and

$$x_1 + x_2 = -.2.$$

The graph in Figure 2.17 shows that the response of the net will now be correct for (1, 1).

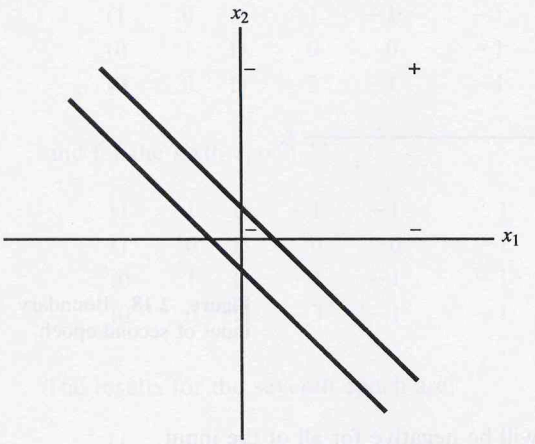

Figure 2.17 Boundary after first training input of second epoch.

For the second input in the second epoch, we have:

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
(x_1	x_2	1)							(w_1	w_2	b)
									(1	1	0)
(1	0	1)	1	1	−1	(−1	0	−1)	(0	1	−1)

The separating lines become

$$x_2 - 1 = .2$$

and

$$x_2 - 1 = -.2.$$

The graph in Figure 2.18 shows that the response of the net will now be correct (negative) for the input points (1, 0) and (0, 0); the response for input points (0, 1) and (1, 1) will be 0, since the net input would be 0, which is between −.2 and .2 ($\theta = .2$).

In the second epoch, the third input yields:

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
(x_1	x_2	1)							(w_1	w_2	b)
									(0	1	−1)
(0	1	1)	0	0	−1	(0	−1	−1)	(0	0	−2)

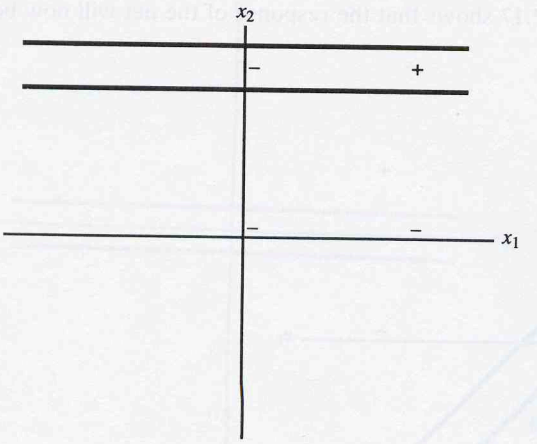

Figure 2.18 Boundary after second input of second epoch.

Again, the response will be negative for all of the input.

To complete the second epoch of training, we present the fourth training pattern:

INPUT			NET	OUT	TARGET	WEIGHTS CHANGE			WEIGHTS		
(x_1	x_2	1)							(w_1	w_2	b)
									(0	0	−2)
(0	0	1)	−2	−1	−1	(0	0	0)	(0	0	−2)

The results for the third epoch are:

INPUT			NET	OUT	TARGET	WEIGHT CHANGES			WEIGHTS		
(x_1	x_2	1)							(w_1	w_2	b)
									(0	0	−2)
(1	1	1)	−2	−1	1	(1	1	1)	(1	1	−1)
(1	0	1)	0	0	−1	(−1	0	−1)	(0	1	−2)
(0	1	1)	−1	−1	−1	(0	0	0)	(0	1	−2)
(0	0	1)	−2	−1	−1	(0	0	0)	(0	1	−2)

The results for the fourth epoch are:

(1	1	1)	−1	−1	1	(1	1	1)	(1	2	−1)
(1	0	1)	0	0	−1	(−1	0	−1)	(0	2	−2)
(0	1	1)	0	0	−1	(0	−1	−1)	(0	1	−3)
(0	0	1)	−3	−1	−1	(0	0	0)	(0	1	−3)

For the fifth epoch, we have

(1	1	1)	−2	−1	1	(1	1	1)	(1	2	−2)
(1	0	1)	−1	−1	−1	(0	0	0)	(1	2	−2)
(0	1	1)	0	0	−1	(0	−1	−1)	(1	1	−3)
(0	0	1)	−3	−1	−1	(0	0	0)	(1	1	−3)

and for the sixth epoch,

(1	1	1)	−1	−1	1	(1	1	1)	(2	2	−2)
(1	0	1)	0	0	−1	(−1	0	−1)	(1	2	−3)
(0	1	1)	−1	−1	−1	(0	0	0)	(1	2	−3)
(0	0	1)	−3	−1	−1	(0	0	0)	(1	2	−3)

The results for the seventh epoch are:

(1	1	1)	0	0	1	(1	1	1)	(2	3	−2)
(1	0	1)	0	0	−1	(−1	0	−1)	(1	3	−3)
(0	1	1)	0	0	−1	(0	−1	−1)	(1	2	−4)
(0	0	1)	−4	−1	−1	(0	0	0)	(1	2	−4)

The eighth epoch yields

(1	1	1)	−1	−1	1	(1	1	1)	(2	3	−3)
(1	0	1)	−1	−1	−1	(0	0	0)	(2	3	−3)
(0	1	1)	0	0	−1	(0	−1	−1)	(2	2	−4)
(0	0	1)	−4	−1	−1	(0	0	0)	(2	2	−4)

and the ninth

(1	1	1)	0	0	1	(1	1	1)	(3	3	−3)
(1	0	1)	0	0	−1	(−1	0	−1)	(2	3	−4)
(0	1	1)	−1	−1	−1	(0	0	0)	(2	3	−4)
(0	0	1)	−4	−1	−1	(0	0	0)	(2	3	−4)

Finally, the results for the tenth epoch are:

(1	1	1)	1	1	1	(0	0	0)	(2	3	−4)
(1	0	1)	−2	−1	−1	(0	0	0)	(2	3	−4)
(0	1	1)	−1	−1	−1	(0	0	0)	(2	3	−4)
(0	0	1)	−4	−1	−1	(0	0	0)	(2	3	−4)

Thus, the positive response is given by all points such that

$$2x_1 + 3x_2 - 4 > .2,$$

with boundary line

$$x_2 = -\frac{2}{3}x_1 + \frac{7}{5},$$

and the negative response is given by all points such that

$$2x_1 + 3x_2 - 4 < -.2,$$

with boundary line

$$x_2 = -\frac{2}{3}x_1 + \frac{19}{15}$$

(see Figure 2.19.)

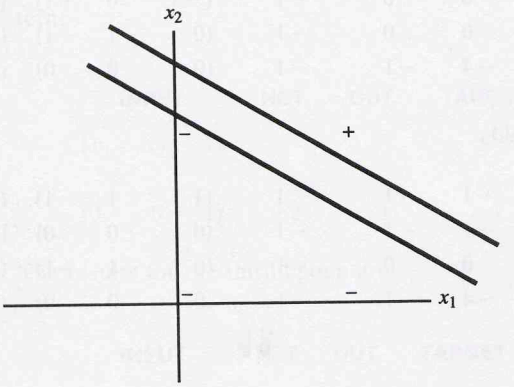

Figure 2.19 Final decision boundaries for AND function in perceptron learning.

Since the proof of the perceptron learning rule convergence theorem (Section 2.3.4) shows that binary input is not required, and in previous examples bipolar input was often preferable, we consider again the previous example, but with bipolar inputs, an adjustable bias, and $\theta = 0$. This variation provides the most direct comparison with Widrow-Hoff learning (an ADALINE net), which we consider in the next section. Note that it is not necessary to modify the training set so that all patterns are mapped to $+1$ (as is done in the proof of the perceptron learning rule convergence theorem); the weight adjustment is $t\mathbf{x}$ whenever the response of the net to input vector $\mathbf{x}$ is incorrect. The target value is still bipolar.