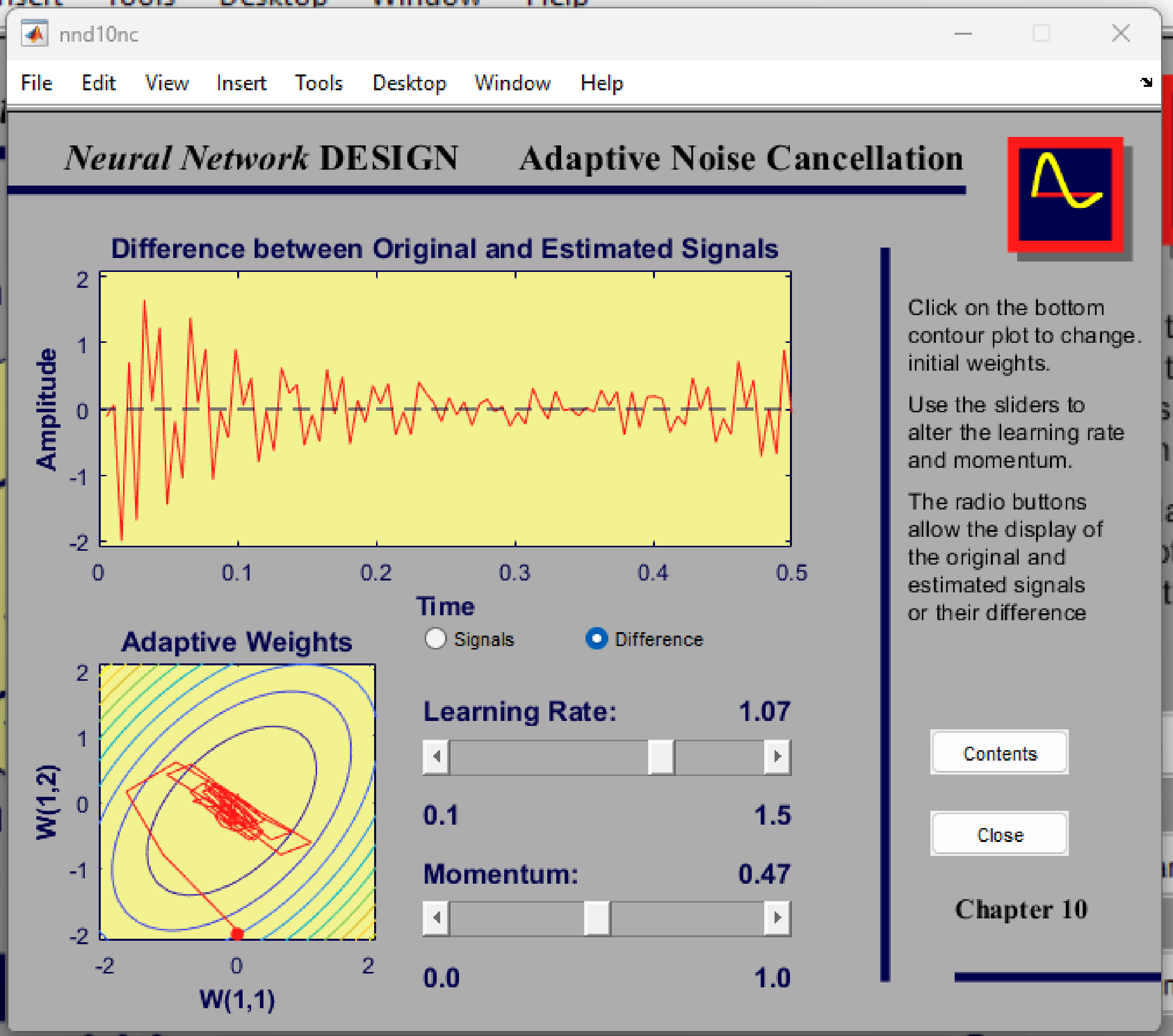

nnd10nc
File
Edit
View
Insert
Tools
Desktop
Window
Help
Neural Network DESIGN
Adaptive Noise Cancellation
Difference between Original and Estimated Signals
Amplitude
2
1
0
-1
-2
0
0.1
0.2
0.3
0.4
0.5
Time
Signals
Difference
Adaptive Weights
W(1,2)
W(1,1)
-2
0
2
Learning Rate:
1.07
0.1
1.5
Momentum:
0.47
0.0
1.0
Click on the bottom contour plot to change. initial weights.
Use the sliders to alter the learning rate and momentum.
The radio buttons allow the display of the original and estimated signals or their difference
Contents
Close
Chapter 10

Solved Problems

P4.1 Solve the three simple classification problems shown in Figure P4.1 by drawing a decision boundary. Find weight and bias values that result in single-neuron perceptrons with the chosen decision boundaries.

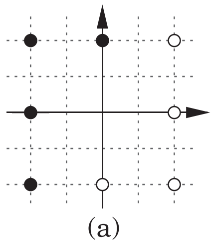
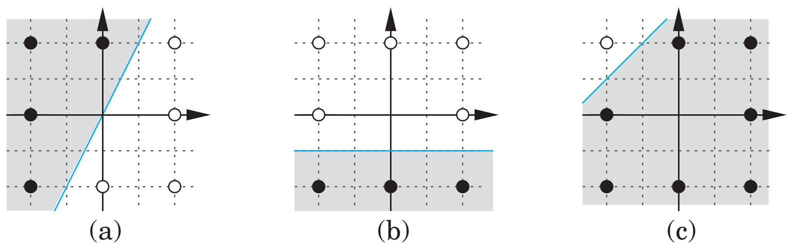
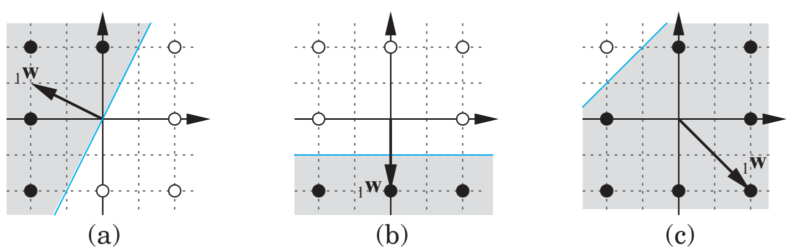


Figure P4.1 Simple Classification Problems

First we draw a line between each set of dark and light data points.



The next step is to find the weights and biases. The weight vectors must be orthogonal to the decision boundaries, and pointing in the direction of points to be classified as 1 (the dark points). The weight vectors can have any length we like.



Here is one set of choices for the weight vectors:

$$\text{(a) } {}_1\mathbf{w}^T = \begin{bmatrix} -2 & 1 \end{bmatrix}, \quad \text{(b) } {}_1\mathbf{w}^T = \begin{bmatrix} 0 & -2 \end{bmatrix}, \quad \text{(c) } {}_1\mathbf{w}^T = \begin{bmatrix} 2 & -2 \end{bmatrix}.$$

Now we find the bias values for each perceptron by picking a point on the decision boundary and satisfying Eq. (4.15).

$$\begin{aligned} {}_1\mathbf{w}^T\mathbf{p} + b &= 0 \\ b &= -{}_1\mathbf{w}^T\mathbf{p} \end{aligned}$$

This gives us the following three biases:

$$\text{(a) } b = -\begin{bmatrix} -2 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0\text{, (b) } b = -\begin{bmatrix} 0 & -2 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix} = -2\text{, (c) } b = -\begin{bmatrix} 2 & -2 \end{bmatrix}\begin{bmatrix} -2 \\ 1 \end{bmatrix} = 6$$

We can now check our solution against the original points. Here we test the first network on the input vector $\mathbf{p} = \begin{bmatrix} -2 & 2 \end{bmatrix}^T$.

$$\begin{aligned} a &= hardlim({}_1\mathbf{w}^T\mathbf{p} + b) \\ &= hardlim\left(\begin{bmatrix} -2 & 1 \end{bmatrix}\begin{bmatrix} -2 \\ 2 \end{bmatrix} + 0\right) \\ &= hardlim(6) \\ &= 1 \end{aligned}$$

P4.3 We have a classification problem with four classes of input vector. The four classes are

$$\text{class 1:}\left\{\mathbf{p}_1 = \begin{bmatrix}1\\1\end{bmatrix}, \mathbf{p}_2 = \begin{bmatrix}1\\2\end{bmatrix}\right\}, \text{class 2: }\left\{\mathbf{p}_3 = \begin{bmatrix}2\\-1\end{bmatrix}, \mathbf{p}_4 = \begin{bmatrix}2\\0\end{bmatrix}\right\},$$

$$\text{class 3:}\left\{\mathbf{p}_5 = \begin{bmatrix}-1\\2\end{bmatrix}, \mathbf{p}_6 = \begin{bmatrix}-2\\1\end{bmatrix}\right\}, \text{class 4: }\left\{\mathbf{p}_7 = \begin{bmatrix}-1\\-1\end{bmatrix}, \mathbf{p}_8 = \begin{bmatrix}-2\\-2\end{bmatrix}\right\}.$$

Design a perceptron network to solve this problem.

To solve a problem with four classes of input vector we will need a perceptron with at least two neurons, since an S-neuron perceptron can categorize 2^S classes. The two-neuron perceptron is shown in Figure P4.2.

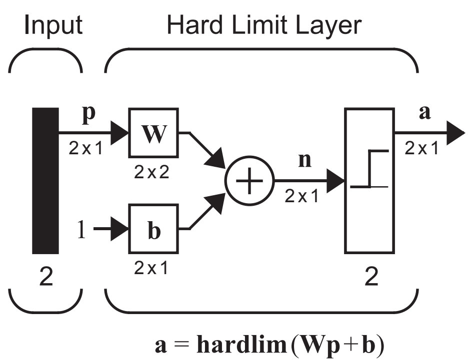


Figure P4.2 Two-Neuron Perceptron

Let's begin by displaying the input vectors, as in Figure P4.3. The light circles ○ indicate class 1 vectors, the light squares □ indicate class 2 vectors, the dark circles ● indicate class 3 vectors, and the dark squares ■ indicate class 4 vectors.

A two-neuron perceptron creates two decision boundaries. Therefore, to divide the input space into the four categories, we need to have one decision boundary divide the four classes into two sets of two. The remaining boundary must then isolate each class. Two such boundaries are illustrated in Figure P4.4. We now know that our patterns are linearly separable.

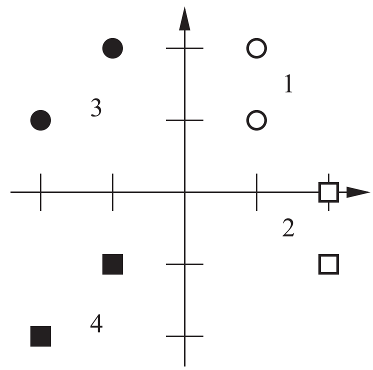


Figure P4.3 Input Vectors for Problem P4.3

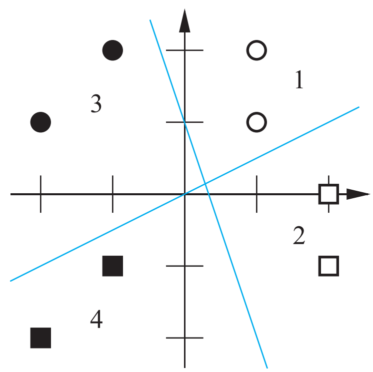


Figure P4.4 Tentative Decision Boundaries for Problem P4.3

The weight vectors should be orthogonal to the decision boundaries and should point toward the regions where the neuron outputs are 1. The next step is to decide which side of each boundary should produce a 1. One choice is illustrated in Figure P4.5, where the shaded areas represent outputs of 1. The darkest shading indicates that both neuron outputs are 1. Note that this solution corresponds to target values of

$$\text{class 1:}\left\{\mathbf{t}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right\}, \text{ class 2: } \left\{\mathbf{t}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right\},$$

$$\text{class 3:}\left\{\mathbf{t}_5 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{t}_6 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\}, \text{ class 4: } \left\{\mathbf{t}_7 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \mathbf{t}_8 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right\}.$$

We can now select the weight vectors:

$$ {}_1\mathbf{w} = \begin{bmatrix} -3 \\ -1 \end{bmatrix} \text{ and } {}_2\mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}. $$

Note that the lengths of the weight vectors is not important, only their directions. They must be orthogonal to the decision boundaries. Now we can calculate the bias by picking a point on a boundary and satisfying Eq. (4.15):

$$ b_1 = -{}_1\mathbf{w}^T\mathbf{p} = -\begin{bmatrix} -3 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 1, $$

$$ b_2 = -{}_2\mathbf{w}^T\mathbf{p} = -\begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0. $$

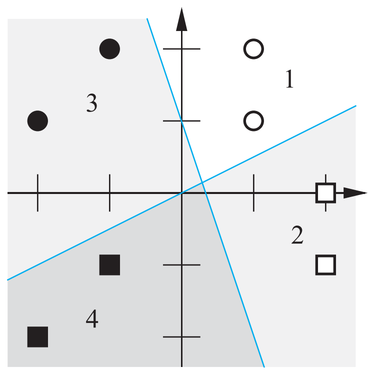


Figure P4.5 Decision Regions for Problem P4.3

In matrix form we have

$$ \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \end{bmatrix} = \begin{bmatrix} -3 & -1 \\ 1 & -2 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, $$

which completes our design.

P4.4 Solve the following classification problem with the perceptron rule. Apply each input vector in order, for as many repetitions as it takes to ensure that the problem is solved. Draw a graph of the problem only after you have found a solution.

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0 \right\} \left\{ \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 1 \right\}$$

Use the initial weights and bias:

$$\mathbf{W}(0) = \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad b(0) = 0 .$$

We start by calculating the perceptron's output a for the first input vector $\mathbf{p}_1$, using the initial weights and bias.

$$a = hardlim(\mathbf{W}(0)\mathbf{p}_1 + b(0))$$

$$= hardlim\left(\begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0 \right) = hardlim(0) = 1$$

The output a does not equal the target value t_1, so we use the perceptron rule to find new weights and biases based on the error.

$$e = t_1 - a = 0 - 1 = -1$$
$$\mathbf{W}(1) = \mathbf{W}(0) + e\mathbf{p}_1^T = \begin{bmatrix} 0 & 0 \end{bmatrix} + (-1)\begin{bmatrix} 2 & 2 \end{bmatrix} = \begin{bmatrix} -2 & -2 \end{bmatrix}$$
$$b(1) = b(0) + e = 0 + (-1) = -1$$

We now apply the second input vector $\mathbf{p}_2$, using the updated weights and bias.

$$a = hardlim(\mathbf{W}(1)\mathbf{p}_2 + b(1))$$

$$= hardlim\left(\begin{bmatrix} -2 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} - 1 \right) = hardlim(1) = 1$$

This time the output a is equal to the target t_2. Application of the perceptron rule will not result in any changes.

$$\mathbf{W}(2) = \mathbf{W}(1)$$
$$b(2) = b(1)$$

We now apply the third input vector.

$$a = hardlim(\mathbf{W}(2)\mathbf{p}_3 + b(2))$$

$$= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-2 \\ 2\end{bmatrix} - 1\right) = hardlim(-1) = 0$$

The output in response to input vector $\mathbf{p}_3$ is equal to the target t_3, so there will be no changes.

$$\mathbf{W}(3) = \mathbf{W}(2)$$
$$b(3) = b(2)$$

We now move on to the last input vector $\mathbf{p}_4$.

$$a = hardlim(\mathbf{W}(3)\mathbf{p}_4 + b(3))$$

$$= hardlim\left(\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-1 \\ 1\end{bmatrix} - 1\right) = hardlim(-1) = 0$$

This time the output a does not equal the appropriate target t_4. The perceptron rule will result in a new set of values for $\mathbf{W}$ and b.

$$e = t_4 - a = 1 - 0 = 1$$
$$\mathbf{W}(4) = \mathbf{W}(3) + e\mathbf{p}_4^T = \begin{bmatrix}-2 & -2\end{bmatrix} + (1)\begin{bmatrix}-1 & 1\end{bmatrix} = \begin{bmatrix}-3 & -1\end{bmatrix}$$
$$b(4) = b(3) + e = -1 + 1 = 0$$

We now must check the first vector $\mathbf{p}_1$ again. This time the output a is equal to the associated target t_1.

$$a = hardlim(\mathbf{W}(4)\mathbf{p}_1 + b(4))$$

$$= hardlim\left(\begin{bmatrix}-3 & -1\end{bmatrix}\begin{bmatrix}2 \\ 2\end{bmatrix} + 0\right) = hardlim(-8) = 0$$

Therefore there are no changes.

$$\mathbf{W}(5) = \mathbf{W}(4)$$
$$b(5) = b(4)$$

The second presentation of $\mathbf{p}_2$ results in an error and therefore a new set of weight and bias values.

$$a = hardlim(\mathbf{W}(5)\mathbf{p}_2 + b(5))$$

$$= hardlim\left(\begin{bmatrix} -3 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} + 0\right) = hardlim(-1) = 0$$

Here are those new values:

$$e = t_2 - a = 1 - 0 = 1$$
$$\mathbf{W}(6) = \mathbf{W}(5) + e\mathbf{p}_2^T = \begin{bmatrix} -3 & -1 \end{bmatrix} + (1)\begin{bmatrix} 1 & -2 \end{bmatrix} = \begin{bmatrix} -2 & -3 \end{bmatrix}$$
$$b(6) = b(5) + e = 0 + 1 = 1.$$

Cycling through each input vector once more results in no errors.

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_3 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} -2 \\ 2 \end{bmatrix} + 1\right) = 0 = t_3$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_4 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} + 1\right) = 1 = t_4$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_1 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} + 1\right) = 0 = t_1$$

$$a = hardlim(\mathbf{W}(6)\mathbf{p}_2 + b(6)) = hardlim\left(\begin{bmatrix} -2 & -3 \end{bmatrix}\begin{bmatrix} 1 \\ -2 \end{bmatrix} + 1\right) = 1 = t_2$$

Therefore the algorithm has converged. The final solution is:

$$\mathbf{W} = \begin{bmatrix} -2 & -3 \end{bmatrix} \qquad b = 1 .$$

Now we can graph the training data and the decision boundary of the solution. The decision boundary is given by

$$n = \mathbf{W}\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = -2p_1 - 3p_2 + 1 = 0 .$$

To find the p_2 intercept of the decision boundary, set $p_1 = 0$:

$$p_2 = -\frac{b}{w_{1,2}} = -\frac{1}{-3} = \frac{1}{3} \qquad \text{if } p_1 = 0$$

To find the p_1 intercept, set $p_2 = 0$:

$$p_1 = -\frac{b}{w_{1,1}} = -\frac{1}{-2} = \frac{1}{2} \qquad \text{if } p_2 = 0 \quad .$$

The resulting decision boundary is illustrated in Figure P4.6.

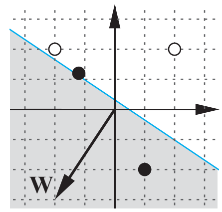


Figure P4.6 Decision Boundary for Problem P4.4

Note that the decision boundary falls across one of the training vectors. This is acceptable, given the problem definition, since the hard limit function returns 1 when given an input of 0, and the target for the vector in question is indeed 1.

P4.5 Consider again the four-class decision problem that we introduced in Problem P4.3. Train a perceptron network to solve this problem using the perceptron learning rule.

If we use the same target vectors that we introduced in Problem P4.3, the training set will be:

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\1\end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix}0\\0\end{bmatrix}\right\} \left\{\mathbf{p}_2 = \begin{bmatrix}1\\2\end{bmatrix}, \mathbf{t}_2 = \begin{bmatrix}0\\0\end{bmatrix}\right\} \left\{\mathbf{p}_3 = \begin{bmatrix}2\\-1\end{bmatrix}, \mathbf{t}_3 = \begin{bmatrix}0\\1\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_4 = \begin{bmatrix}2\\0\end{bmatrix}, \mathbf{t}_4 = \begin{bmatrix}0\\1\end{bmatrix}\right\} \left\{\mathbf{p}_5 = \begin{bmatrix}-1\\2\end{bmatrix}, \mathbf{t}_5 = \begin{bmatrix}1\\0\end{bmatrix}\right\} \left\{\mathbf{p}_6 = \begin{bmatrix}-2\\1\end{bmatrix}, \mathbf{t}_6 = \begin{bmatrix}1\\0\end{bmatrix}\right\}$$

$$\left\{\mathbf{p}_7 = \begin{bmatrix}-1\\-1\end{bmatrix}, \mathbf{t}_7 = \begin{bmatrix}1\\1\end{bmatrix}\right\} \left\{\mathbf{p}_8 = \begin{bmatrix}-2\\-2\end{bmatrix}, \mathbf{t}_8 = \begin{bmatrix}1\\1\end{bmatrix}\right\}.$$

Let's begin the algorithm with the following initial weights and biases:

$$\mathbf{W}(0) = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}, \ \mathbf{b}(0) = \begin{bmatrix}1\\1\end{bmatrix}.$$

The first iteration is

$$\mathbf{a} = hardlim\,(\mathbf{W}(0)\mathbf{p}_1 + \mathbf{b}(0)) = hardlim\,(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

$$\mathbf{e} = \mathbf{t}_1 - \mathbf{a} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

$$\mathbf{W}(1) = \mathbf{W}(0) + \mathbf{e}\mathbf{p}_1^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix},$$

$$\mathbf{b}(1) = \mathbf{b}(0) + \mathbf{e} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The second iteration is

$$\mathbf{a} = hardlim\,(\mathbf{W}(1)\mathbf{p}_2 + \mathbf{b}(1)) = hardlim\,(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\mathbf{e} = \mathbf{t}_2 - \mathbf{a} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$\mathbf{W}(2) = \mathbf{W}(1) + \mathbf{e}\mathbf{p}_2^T = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix},$$

$$\mathbf{b}(2) = \mathbf{b}(1) + \mathbf{e} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The third iteration is

$$\mathbf{a} = hardlim\,(\mathbf{W}(2)\mathbf{p}_3 + \mathbf{b}(2)) = hardlim\,(\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

$$\mathbf{e} = \mathbf{t}_3 - \mathbf{a} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix},$$

$$\mathbf{W}(3) = \mathbf{W}(2) + \mathbf{e}\mathbf{p}_3^T = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & -1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix},$$

$$\mathbf{b}(3) = \mathbf{b}(2) + \mathbf{e} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Iterations four through eight produce no changes in the weights.

$$\mathbf{W}(8) = \mathbf{W}(7) = \mathbf{W}(6) = \mathbf{W}(5) = \mathbf{W}(4) = \mathbf{W}(3)$$

$$\mathbf{b}(8) = \mathbf{b}(7) = \mathbf{b}(6) = \mathbf{b}(5) = \mathbf{b}(4) = \mathbf{b}(3)$$

The ninth iteration produces

$$\mathbf{a} = hardlim\,(\mathbf{W}(8)\mathbf{p}_1 + \mathbf{b}(8)) = hardlim\,(\begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\mathbf{e} = \mathbf{t}_1 - \mathbf{a} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix},$$

$$\mathbf{W}(9) = \mathbf{W}(8) + \mathbf{e}\mathbf{p}_1^T = \begin{bmatrix} -2 & 0 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix},$$

$$\mathbf{b}(9) = \mathbf{b}(8) + \mathbf{e} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix}.$$

At this point the algorithm has converged, since all input patterns will be correctly classified. The final decision boundaries are displayed in Figure P4.7. Compare this result with the network we designed in Problem P4.3.

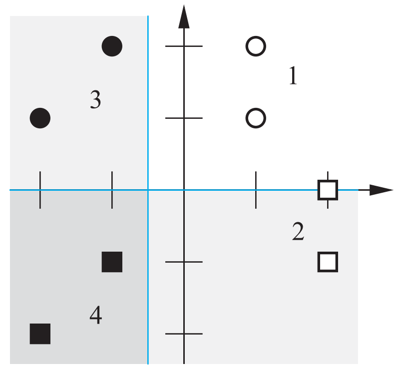


Figure P4.7 Final Decision Boundaries for Problem P4.5