Solved Problems

P10.1 Consider the ADALINE filter in Figure P10.1.

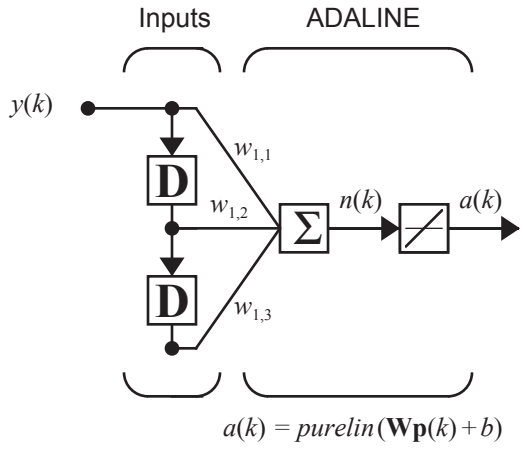


Figure P10.1 ADALINE Filter

Suppose that

$$w_{1,1} = 2, \quad w_{1,2} = -1, \quad w_{1,3} = 3,$$

and the input sequence is

$$\{y(k)\} = \{\ldots, 0, 0, 0, 5, -4, 0, 0, 0, \ldots\}$$

where $y(0) = 5$**,** $y(1) = -4$**, etc.**

i. **What is the filter output just prior to** $k = 0$**?**

ii. **What is the filter output from** $k = 0$ **to** $k = 5$**?**

iii. **How long does** $y(0)$ **contribute to the output?**

i. Just prior to $k = 0$ three zeros have entered the filter, and the output is zero.

ii. At $k = 0$ the digit "5" has entered the filter, and it will be multiplied by $w_{1,1}$, which has the value 2, so that $a(0) = 10$. This can be viewed as the matrix operation:

$$a(0) = \mathbf{W}\mathbf{p}(0) = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} y(0) \\ y(-1) \\ y(-2) \end{bmatrix} = \begin{bmatrix} 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ 0 \\ 0 \end{bmatrix} = 10 .$$

Similarly, one can calculate the next outputs as

$$a(1) = \mathbf{W}\mathbf{p}(1) = \begin{bmatrix} 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} -4 \\ 5 \\ 0 \end{bmatrix} = -13$$

$$a(2) = \mathbf{W}\mathbf{p}(2) = \begin{bmatrix} 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ -4 \\ 5 \end{bmatrix} = 19$$

$$a(3) = \mathbf{W}\mathbf{p}(3) = \begin{bmatrix} 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -4 \end{bmatrix} = -12 , \; a(4) = \mathbf{W}\mathbf{p}(4) = \begin{bmatrix} 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = 0 .$$

All remaining outputs will be zero.

iii. The effects of $y(0)$ last from $k = 0$ through $k = 2$, so it will have an influence for three time intervals. This corresponds to the length of the impulse response of this filter.

P10.2 Suppose that we want to design an ADALINE network to distinguish between various categories of input vectors. Let us first try the categories listed below:

$$\textbf{Category I:} \quad \mathbf{p}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T \textbf{ and } \mathbf{p}_2 = \begin{bmatrix} -1 & -1 \end{bmatrix}^T$$

$$\textbf{Category II:} \quad \mathbf{p}_3 = \begin{bmatrix} 2 & 2 \end{bmatrix}^T .$$

i. Can an ADALINE network be designed to make such a distinction?

ii. If the answer to part (i) is yes, what set of weights and bias might be used?

Next consider a different set of categories.

$$\textbf{Category III:} \quad \mathbf{p}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T \textbf{ and } \mathbf{p}_2 = \begin{bmatrix} 1 & -1 \end{bmatrix}^T$$

Category IV: $\mathbf{p}_3 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$.

iii. Can an ADALINE network be designed to make such a distinction?

iv. If the answer to part (iii) is yes, what set of weights and bias might be used?

i. The input vectors are plotted in Figure P10.2.

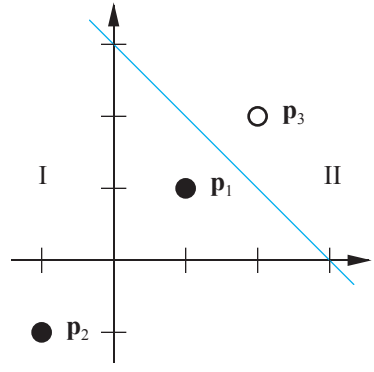


Figure P10.2 Input Vectors for Problem P10.1 (i)

The blue line in this figure is a decision boundary that separates the two categories successfully. Since they are linearly separable, an ADALINE network will do the job.

ii. The decision boundary passes through the points $(3, 0)$ and $(0, 3)$. We know these points to be the intercepts $-b/w_{1,1}$ and $-b/w_{1,2}$. Thus, a solution

$$b = 3,\ w_{1,1} = -1,\ w_{1,2} = -1,$$

is satisfactory. Note that if the output of the ADALINE is positive or zero the input vector is classified as Category I, and if the output is negative the input vector is classified as Category II. This solution also provides for error, since the decision boundary bisects the line between $\mathbf{p}_1$ and $\mathbf{p}_3$.

iii. The input vectors to be distinguished are shown in Figure P10.3. The vectors in the figure are not linearly separable, so an ADALINE network cannot distinguish between them.

iv. As noted in part (iii), an ADALINE cannot do the job, so there are no values for the weights and bias that are satisfactory.

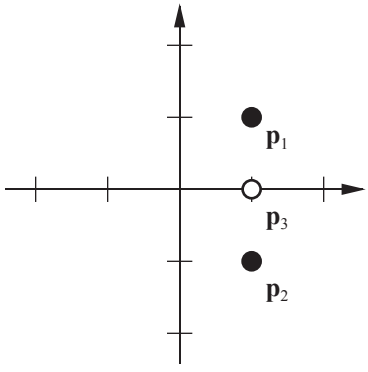


Figure P10.3 Input Vectors for Problem P10.1 (iii)

P10.3 Suppose that we have the following input/target pairs:

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\1\end{bmatrix}, t_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix}1\\-1\end{bmatrix}, t_2 = -1\right\}.$$

These patterns occur with equal probability, and they are used to train an ADALINE network with no bias. What does the mean square error performance surface look like?

First we need to calculate the various terms of the quadratic function. Recall from Eq. (10.11) that the performance index can be written as

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}.$$

Therefore we need to calculate c, $\mathbf{h}$ and $\mathbf{R}$.

The probability of each input occurring is 0.5, so the probability of each target is also 0.5. Thus, the expected value of the square of the targets is

$$c = E[t^2] = (1)^2(0.5) + (-1)^2(0.5) = 1.$$

In a similar way, the cross-correlation between the input and the target can be calculated:

$$\mathbf{h} = E[t\mathbf{z}] = (0.5)(1)\begin{bmatrix}1\\1\end{bmatrix} + (0.5)(-1)\begin{bmatrix}1\\-1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}.$$

Finally, the input correlation matrix $\mathbf{R}$ is

$$\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] = \mathbf{p}_1\mathbf{p}_1^T(0.5) + \mathbf{p}_2\mathbf{p}_2^T(0.5)$$

$$= (0.5)\left[\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1 & 1\end{bmatrix} + \begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix}\right] = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

Therefore the mean square error performance index is

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}$$

$$= 1 - 2\begin{bmatrix}w_{1,1} & w_{1,2}\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}w_{1,1} & w_{1,2}\end{bmatrix}\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}w_{1,1}\\w_{1,2}\end{bmatrix}$$

$$= 1 - 2w_{1,2} + w_{1,1}^2 + w_{1,2}^2$$

The Hessian matrix of $F(\mathbf{x})$, which is equal to $2\mathbf{R}$, has both eigenvalues at 2. Therefore the contours of the performance surface will be circular. To find the center of the contours (the minimum point), we need to solve Eq. (10.18):

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}^{-1}\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}.$$

Thus we have a minimum at $w_{1,1} = 0$, $w_{1,2} = 1$. The resulting mean square error performance surface is shown in Figure P10.4.

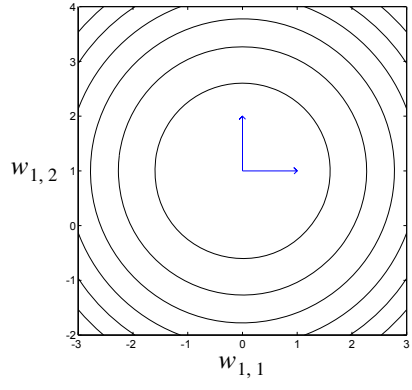


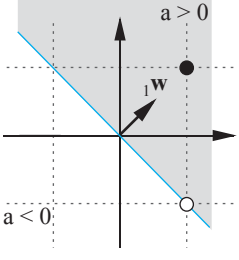
Figure P10.4 Contour Plot of $F(\mathbf{x})$ for Problem P10.3

P10.4 Consider the system of Problem P10.3 again. Train the network using the LMS algorithm, with the initial guess set to zero and a learning rate $\alpha = 0.25$. Apply each reference pattern only once during training. Draw the decision boundary at each stage.

Assume the input vector $\mathbf{p}_1$ is presented first. The output, error and new weights are calculated as follows:

$$a(0) = purelin\left[\begin{bmatrix}0 & 0\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\right] = 0,$$

$$e(0) = t(0) - a(0) = 1 - 0 = 1,$$



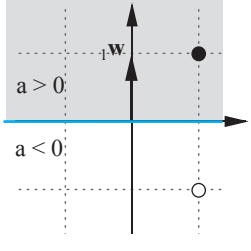
$$\mathbf{W}(1) = \mathbf{W}(0) + 2\alpha e(0)\mathbf{p}(0)^T = \begin{bmatrix}0 & 0\end{bmatrix} + 2\left(\frac{1}{4}\right)(1)\begin{bmatrix}1 & 1\end{bmatrix} = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix}.$$

The decision boundary associated with these weights is shown to the left.

Now apply the second input vector:

$$a(1) = purelin\left\{\begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\right\} = 0,$$

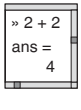
$$e(1) = t(1) - a(1) = -1 - 0 = -1,$$



$$\mathbf{W}(2) = \mathbf{W}(1) + 2\alpha e(1)\mathbf{p}(1)^T = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\end{bmatrix} + 2\left(\frac{1}{4}\right)(-1)\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}0 & 1\end{bmatrix}.$$

The decision boundary associated with these weights is shown to the left. This boundary shows real promise. It is exactly halfway between the input vectors. You might verify for yourself that each input vector, when applied, yields its correct associated target. (What set of weights would be optimal if the targets associated with the two input vectors were exchanged?)

P10.5 Now consider the convergence of the system of Problems P10.3 and P10.4. What is the maximum stable learning rate for the LMS algorithm?



The LMS convergence is determined by the learning rate α, which should not exceed the reciprocal of the largest eigenvalue of $\mathbf{R}$. We can determine this limit by finding these eigenvalues using MATLAB.

```
[V,D] = eig (R)
V =
        1       0
        0       1

D=
        1       0
        0       1
```

The diagonal terms of matrix `D` give the eigenvalues, 1 and 1, while the columns of `V` show the eigenvectors. Note, incidentally, that the eigenvectors have the same direction as those shown in Figure P10.4.

The largest eigenvalue, $\lambda_{max} = 1$, sets the upper limit on the learning rate at

$$\alpha < 1/\lambda_{max} = 1/1 = 1 .$$

The suggested learning rate in the previous problem was 0.25, and you found (perhaps) that the LMS algorithm converged quickly. What do you suppose happens when the learning rate is 1.0 or larger?

P10.6 Consider the adaptive filter ADALINE shown in Figure P10.5. The purpose of this filter is to predict the next value of the input signal from the two previous values. Suppose that the input signal is a stationary random process, with autocorrelation function given by

$$C_y(n) = E[y(k)y(k+n)]$$

$$C_y(0) = 3 , \; C_y(1) = -1 , \; C_y(2) = -1 .$$

i. **Sketch the contour plot of the performance index (mean square error).**

ii. **What is the maximum stable value of the learning rate (α) for the LMS algorithm?**

iii. **Assume that a very small value is used for α. Sketch the path of the weights for the LMS algorithm, starting with initial guess $\mathbf{W}(0) = \begin{bmatrix} 0.75 & 0 \end{bmatrix}^T$. Explain your procedure for sketching the path.**

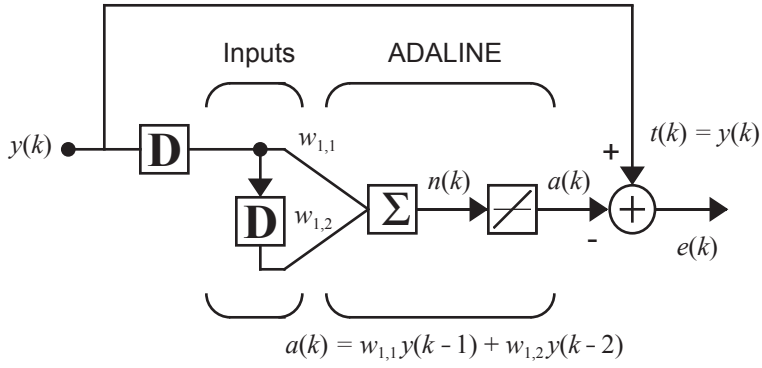


Figure P10.5 Adaptive Predictor

i. To sketch the contour plot we first need to find the performance index and the eigenvalues and eigenvectors of the Hessian matrix. First note that the input vector is given by

$$\mathbf{z}(k) = \mathbf{p}(k) = \begin{bmatrix} y(k-1) \\ y(k-2) \end{bmatrix}.$$

Now consider the performance index. Recall from Eq. (10.12) that

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}.$$

We can calculate the constants in the performance index as shown below:

$$c = E[t^2(k)] = E[y^2(k)] = C_y(0) = 3,$$

$$\mathbf{R} = E[\mathbf{z}\mathbf{z}^T] = E\begin{bmatrix} y^2(k-1) & y(k-1)y(k-2) \\ y(k-1)y(k-2) & y^2(k-2) \end{bmatrix}$$

$$= \begin{bmatrix} C_y(0) & C_y(1) \\ C_y(1) & C_y(0) \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$\mathbf{h} = E[t\,\mathbf{z}] = E\begin{bmatrix} y(k)y(k-1) \\ y(k)y(k-2) \end{bmatrix} = \begin{bmatrix} C_y(1) \\ C_y(2) \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}.$$

The optimal weights are

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3/8 & 1/8 \\ 4/8 & 3/8 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1/2 \\ -1/2 \end{bmatrix}.$$

The Hessian matrix is

$$\nabla^2 F(\mathbf{x}) = \mathbf{A} = 2\mathbf{R} = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}.$$

Now we can get the eigenvalues:

$$\left| \mathbf{A} - \lambda \mathbf{I} \right| = \begin{vmatrix} 6-\lambda & -2 \\ -2 & 6-\lambda \end{vmatrix} = \lambda^2 - 12\lambda + 32 = (\lambda - 8)(\lambda - 4).$$

Thus,

$$\lambda_1 = 4, \qquad \lambda_2 = 8.$$

To find the eigenvectors we use

$$\left[\mathbf{A} - \lambda \mathbf{I}\right]\mathbf{v} = 0.$$

For $\lambda_1 = 4$,

$$\begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \mathbf{v}_1 = 0 \qquad \mathbf{v}_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

and for $\lambda_2 = 8$,

$$\begin{bmatrix} -2 & -2 \\ -2 & -2 \end{bmatrix} \mathbf{v}_2 = 0 \qquad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Therefore the contours of $F(\mathbf{x})$ will be elliptical, with the long axis of each ellipse along the first eigenvector, since the first eigenvalue has the smallest magnitude. The ellipses will be centered at $\mathbf{x}^*$. The contour plot is shown in Figure P10.6.

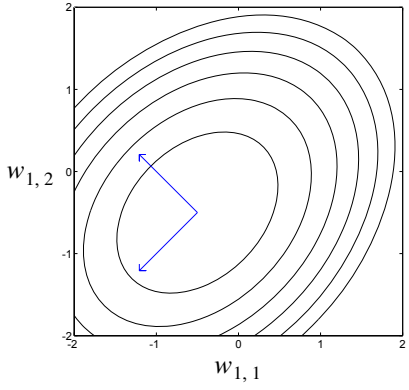


Figure P10.6 Error Contour for Problem P10.6

You might check your sketch by writing a MATLAB M-file to plot the contours.

ii. The maximum stable learning rate is the reciprocal of the maximum eigenvalue of $\mathbf{R}$, which is the same as twice the reciprocal of the largest eigenvalue of the Hessian matrix $\nabla^2 F(\mathbf{x}) = \mathbf{A}$:

$$\alpha < 2/\lambda_{max} = 2/8 = 0.25 .$$

iii. The LMS algorithm is approximate steepest descent, so the trajectory for small learning rates will move perpendicular to the contour lines, as shown in Figure P10.7.

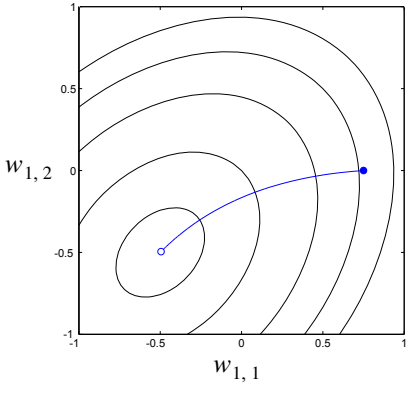


Figure P10.7 LMS Weight Trajectory

P10.7 The pilot of an airplane is talking into a microphone in his cockpit. The sound received by the air traffic controller in the tower is garbled because the pilot's voice signal has been contaminated by engine noise that reaches his microphone. Can you suggest an adaptive ADALINE filter that might help reduce the noise in the signal received by the control tower? Explain your system.

The engine noise that has been inadvertently added to the microphone input can be minimized by using the adaptive filtering system shown in Figure P10.8. A sample of the engine noise is supplied to an adaptive filter through a microphone in the cockpit. The desired output of the filter is the contaminated signal coming from the pilot's microphone. The filter attempts to reduce the "error" signal to a minimum. It can do this only by subtracting the component of the contaminated signal that is linearly correlated with the engine noise (and presumably uncorrelated with the pilot's voice). The result is that a clear voice signal is sent to the control tower, in spite of the fact that the engine noise got into the pilot's microphone along with his voice signal. (See [WiSt85] for discussion of similar noise cancellation systems.)

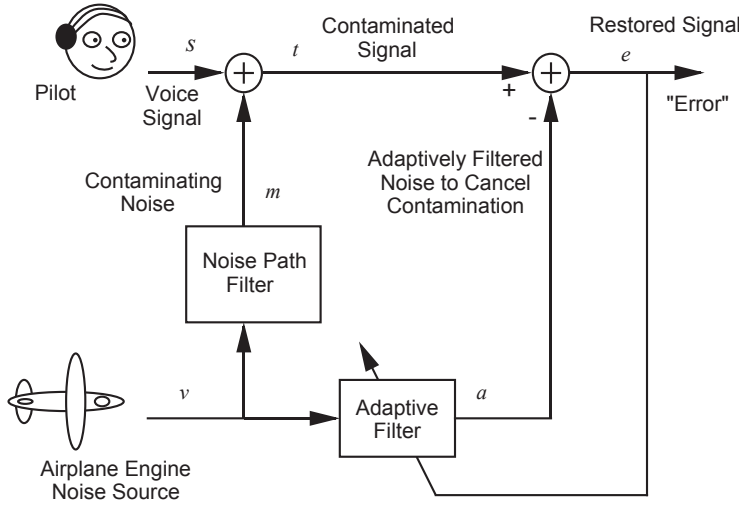


Figure P10.8 Filtering Engine Noise from Pilot's Voice Signal

P10.8 This is a classification problem like that described in Problems P4.3 and P4.5, except that here we will use an ADALINE network and the LMS learning rule rather than the perceptron learning rule. First we will describe the problem.

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

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

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

Train an ADALINE network to solve this problem using the LMS learning rule. Assume that each pattern occurs with probability $1/8$**.**

Let's begin by displaying the input vectors, as in Figure P10.9. 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. These input vectors can be plotted as shown in Figure P10.9.

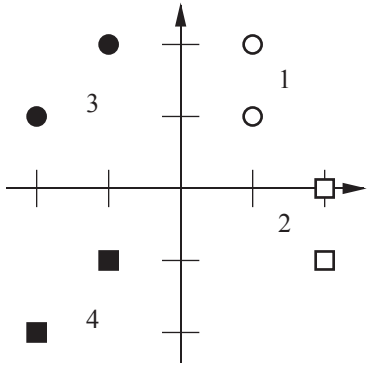


Figure P10.9 Input Vectors for Problem P10.8

We will use target vectors similar to the ones we introduced in Problem P4.3, except that we will replace any targets of 0 by targets of -1. (The perceptron could only output 0 or 1.) Thus, the training set will be:

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

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

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

Also, we will begin as in Problem P4.5 with the following initial weights and biases:

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

Now we are almost ready to train an ADALINE network using the LMS rule. We will use a learning rate of $\alpha = 0.04$, and we will present the input vectors in order according to their subscripts. The first iteration is

$$\mathbf{a}(0) = purelin\,(\mathbf{W}(0)\mathbf{p}(0) + \mathbf{b}(0)) = purelin\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\mathbf{e}(0) = \mathbf{t}(0) - \mathbf{a}(0) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

$$\mathbf{W}(1) = \mathbf{W}(0) + 2\alpha\mathbf{e}(0)\mathbf{p}^T(0)$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + 2(0.04)\begin{bmatrix} -3 \\ -3 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.76 & -0.24 \\ -0.24 & 0.76 \end{bmatrix}$$

$$\mathbf{b}(1) = \mathbf{b}(0) + 2\alpha\mathbf{e}(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2(0.04)\begin{bmatrix} -3 \\ -3 \end{bmatrix} = \begin{bmatrix} 0.76 \\ 0.76 \end{bmatrix}.$$

The second iteration is

$$\mathbf{a}(1) = purelin\,(\mathbf{W}(1)\mathbf{p}(1) + \mathbf{b}(1))$$

$$= purelin\left(\begin{bmatrix} 0.76 & -0.24 \\ -0.24 & 0.76 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.76 \\ 0.76 \end{bmatrix}\right) = \begin{bmatrix} 1.04 \\ 2.04 \end{bmatrix}$$

$$\mathbf{e}(1) = \mathbf{t}(1) - \mathbf{a}(1) = \begin{bmatrix} -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 1.04 \\ 2.04 \end{bmatrix} = \begin{bmatrix} -2.04 \\ -3.04 \end{bmatrix}$$

$$\mathbf{W}(2) = \mathbf{W}(1) + 2\alpha\mathbf{e}(1)\mathbf{p}^T(1)$$

$$= \begin{bmatrix} 0.76 & -0.24 \\ -0.24 & 0.76 \end{bmatrix} + 2(0.04)\begin{bmatrix} -2.04 \\ -3.04 \end{bmatrix}\begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 0.5968 & -0.5664 \\ -0.4832 & 0.2736 \end{bmatrix}$$

$$\mathbf{b}(2) = \mathbf{b}(1) + 2\alpha\mathbf{e}(1) = \begin{bmatrix} 0.76 \\ 0.76 \end{bmatrix} + 2(0.04)\begin{bmatrix} -2.04 \\ -3.04 \end{bmatrix} = \begin{bmatrix} 0.5968 \\ 0.5168 \end{bmatrix}.$$

If we continue until the weights converge we find

$$\mathbf{W}(\infty) = \begin{bmatrix} -0.5948 & -0.0523 \\ 0.1667 & -0.6667 \end{bmatrix}, \quad \mathbf{b}(\infty) = \begin{bmatrix} 0.0131 \\ 0.1667 \end{bmatrix}.$$

The resulting decision boundaries are shown in Figure P10.10. Compare this result with the final decision boundaries created by the perceptron learning rule in Problem P4.5 (Figure P4.7). The perceptron rule stops training when all the patterns are classified correctly. The LMS algorithm moves the boundaries as far from the patterns as possible.

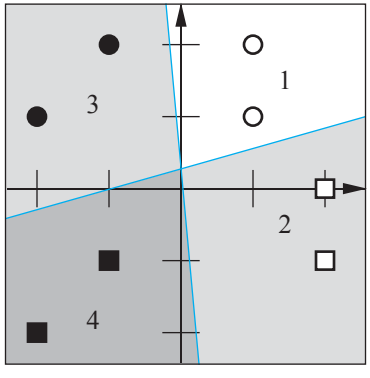


Figure P10.10 Final Decision Boundaries for Problem P10.8

P10.9 Repeat the work of Widrow and Hoff on a pattern recognition problem from their classic 1960 paper [WiHo60]. They wanted to design a recognition system that would classify the six patterns shown in Figure P10.11.

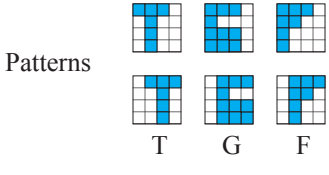


Figure P10.11 Patterns and Their Classification Targets

These patterns represent the letters T, G and F, in an original form on the top and in a shifted form on the bottom. The targets for these letters (in their original and shifted forms) are +60, 0 and -60, respectively. (The values of 60, 0 and -60 were nice for use on the face of a meter that Widrow and Hoff used to display their network output.) The objective is to train a network so that it will classify the six patterns into the appropriate T, G or F groups.

The blue squares in the letters will be assigned the value +1, and the white squares will be assigned the value -1. First we convert each of the letters into a single 16-element vector. We choose to do this by starting at the upper left corner, going down the left column, then going down the second column, etc. For example, the vector corresponding to the unshifted letter T is

$$\mathbf{p}_1 = \begin{bmatrix} 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}^T$$

We have such an input vector for each of the six letters.

The ADALINE network that we will use is shown in Figure P10.12.

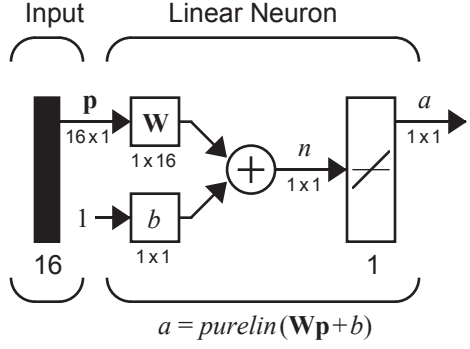


Figure P10.12 Adaptive Pattern Classifier

(Widrow and Hoff built their own machine to realize this ADALINE. According to them, it was “about the size of a lunch pail.”)

Now we will present the six vectors to the network in a random sequence and adjust the weights of the network after each presentation using the LMS algorithm with a learning rate of $\alpha = 0.03$. After each adjustment of weights, all six vectors will be presented to the network to generate their outputs and corresponding errors. The sum of the squares of the errors will be examined as a measure of the quality of the network.

Figure P10.13 illustrates the convergence of the network. The network is trained to recognize these six characters in about 60 presentations, or roughly 10 for each of the possible input vectors.

The results shown in Figure P10.13 are quite like those obtained and published by Widrow and Hoff some 35 years ago. Widrow and Hoff did good science. One can indeed duplicate their work, even decades later (without a lunch pail).

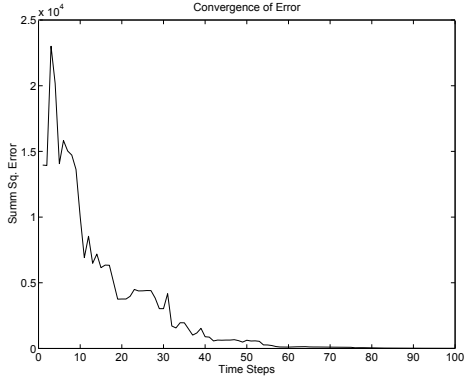


Figure P10.13 Error Convergence with Learning Rate of 0.03

To experiment with this character recognition problem, use the MATLAB® Neural Network Design Demonstration Linear Pattern Classification (`nnd10lc`). *Notice the sensitivity of the network to noise in the input pattern.*