The error is

$$e(2) = t(2) - a(2) = t_1 - a(2) = -1 - (-0.64) = -0.36 . \quad (10.59)$$

The new weights are

$$\mathbf{W}(3) = \mathbf{W}(2) + 2\alpha e(2)\mathbf{p}^T(2) = \begin{bmatrix} 0.016 & 1.1040 & -0.0160 \end{bmatrix} . \quad (10.60)$$

If we continue this procedure, the algorithm converges to

$$\mathbf{W}(\infty) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} . \quad (10.61)$$

Compare this result with the result of the perceptron learning rule in Chapter 4. You will notice that the ADALINE has produced the same decision boundary that we designed in Chapter 3 for the apple/orange problem. This boundary falls halfway between the two reference patterns. The perceptron rule did not produce such a boundary. This is because the perceptron rule stops as soon as the patterns are correctly classified, even though some patterns may be close to the boundaries. The LMS algorithm minimizes the mean square error. Therefore it tries to move the decision boundaries as far from the reference patterns as possible.

Adaptive Filtering

As we mentioned at the beginning of this chapter, the ADALINE network has the same major limitation as the perceptron network; it can only solve linearly separable problems. In spite of this, the ADALINE has been much more widely used than the perceptron network. In fact, it is safe to say that it is one of the most widely used neural networks in practical applications. One of the major application areas of the ADALINE has been adaptive filtering, where it is still used extensively. In this section we will demonstrate an adaptive filtering example.

Tapped Delay Line

In order to use the ADALINE network as an adaptive filter, we need to introduce a new building block, the tapped delay line. A *tapped delay line* with R outputs is shown in Figure 10.4.

The input signal enters from the left. At the output of the tapped delay line we have an R-dimensional vector, consisting of the input signal at the current time and at delays of from 1 to $R-1$ time steps.

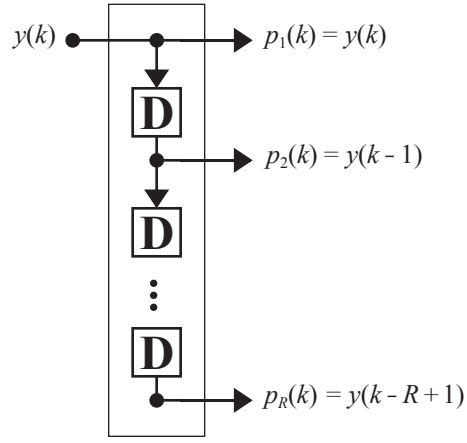


Figure 10.4 Tapped Delay Line

Adaptive Filter

If we combine a tapped delay line with an ADALINE network, we can create an *adaptive filter*, as is shown in Figure 10.5. The output of the filter is given by

$$a(k) = purelin(\mathbf{W}\mathbf{p} + b) = \sum_{i=1}^{R} w_{1,i} y(k-i+1) + b . \tag{10.62}$$

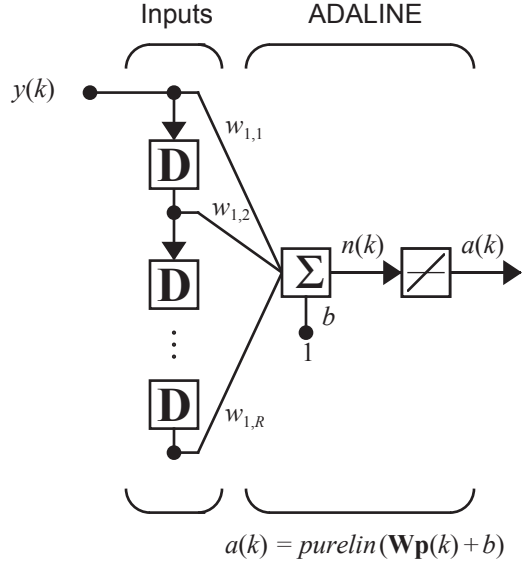


Figure 10.5 Adaptive Filter ADALINE

If you are familiar with digital signal processing, you will recognize the network of Figure 10.5 as a finite impulse response (FIR) filter [WiSt85]. It is beyond the scope of this text to review the field of digital signal processing, but we can demonstrate the usefulness of this adaptive filter through a simple, but practical, example.

Adaptive Noise Cancellation

An adaptive filter can be used in a variety of novel ways. In the following example we will use it for noise cancellation. Take some time to look at this example, for it is a little different from what you might expect. For instance, the output "error" that the network tries to minimize is actually an approximation to the signal we are trying to recover!

Let's suppose that a doctor, in trying to review the electroencephalogram (EEG) of a distracted graduate student, finds that the signal he would like to see has been contaminated by a 60-Hz noise source. He is examining the patient on-line and wants to view the best signal that can be obtained. Figure 10.6 shows how an adaptive filter can be used to remove the contaminating signal.

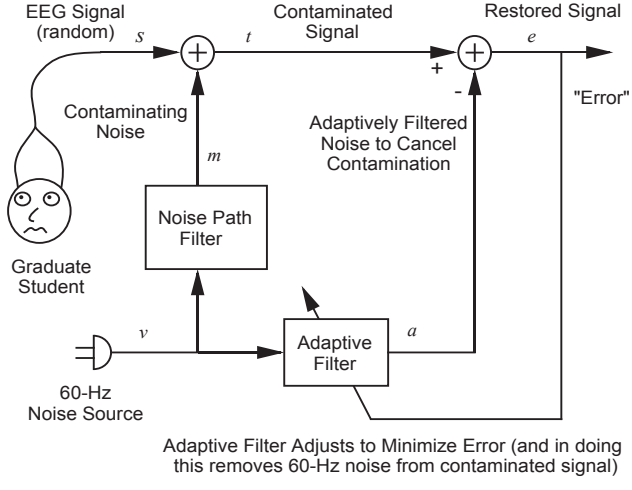


Figure 10.6 Noise Cancellation System

As shown, a sample of the original 60-Hz signal is fed to an adaptive filter, whose elements are adjusted so as to minimize the "error" e. The desired output of the filter is the contaminated EEG signal t. The adaptive filter will do its best to reproduce this contaminated signal, but it only knows about the original noise source, v. Thus, it can only reproduce the part of t that is linearly correlated with v, which is m. In effect, the adaptive filter will attempt to mimic the noise path filter, so that the output of the filter

a will be close to the contaminating noise m. In this way the error e will be close to the original uncontaminated EEG signal s.

In this simple case of a single sine wave noise source, a neuron with two weights and no bias is sufficient to implement the filter. The inputs to the filter are the current and previous values of the noise source. Such a two-input filter can attenuate and phase-shift the noise v in the desired way. The filter is shown in Figure 10.7.

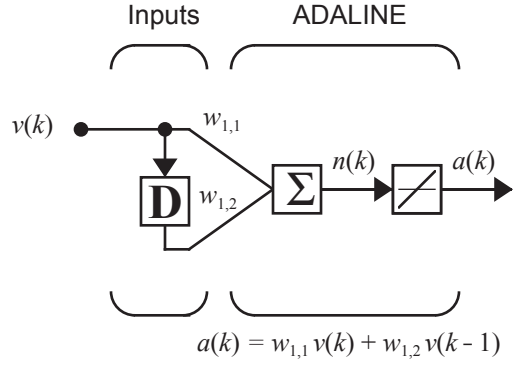


Figure 10.7 Adaptive Filter for Noise Cancellation

We can apply the mathematical relationships developed in the previous sections of this chapter to analyze this system. In order to do so, we will first need to find the input correlation matrix $\mathbf{R}$ and the input/target cross-correlation vector $\mathbf{h}$:

$$\mathbf{R} = [\mathbf{z}\mathbf{z}^T] \text{ and } \mathbf{h} = E[t\mathbf{z}]. \tag{10.63}$$

In our case the input vector is given by the current and previous values of the noise source:

$$\mathbf{z}(k) = \begin{bmatrix} v(k) \\ v(k-1) \end{bmatrix}, \tag{10.64}$$

while the target is the sum of the current signal and filtered noise:

$$t(k) = s(k) + m(k). \tag{10.65}$$

Now expand the expressions for $\mathbf{R}$ and $\mathbf{h}$ to give

$$\mathbf{R} = \begin{bmatrix} E[v^2(k)] & E[v(k)v(k-1)] \\ E[v(k-1)v(k)] & E[v^2(k-1)] \end{bmatrix}, \tag{10.66}$$

and

$$\mathbf{h} = \begin{bmatrix} E[(s(k)+m(k))v(k)] \\ E[(s(k)+m(k))v(k-1)] \end{bmatrix}. \tag{10.67}$$

To obtain specific values for these two quantities we must define the noise signal v, the EEG signal s and the filtered noise m. For this exercise we will assume: the EEG signal is a white (uncorrelated from one time step to the next) random signal uniformly distributed between the values -0.2 and +0.2, the noise source (60-Hz sine wave sampled at 180 Hz) is given by

$$v(k) = 1.2 \sin\left(\frac{2\pi k}{3}\right), \tag{10.68}$$

and the filtered noise that contaminates the EEG is the noise source attenuated by a factor of 10 and shifted in phase by $\pi/2$:

$$m(k) = 0.12 \sin\left(\frac{2\pi k}{3} + \frac{\pi}{2}\right). \tag{10.69}$$

Now calculate the elements of the input correlation matrix $\mathbf{R}$:

$$E[v^2(k)] = (1.2)^2 \frac{1}{3} \sum_{k=1}^{3} \left(\sin\left(\frac{2\pi k}{3}\right)\right)^2 = (1.2)^2 0.5 = 0.72, \tag{10.70}$$

$$E[v^2(k-1)] = E[v^2(k)] = 0.72, \tag{10.71}$$

$$\begin{aligned} E[v(k)v(k-1)] &= \frac{1}{3} \sum_{k=1}^{3} \left(1.2 \sin\frac{2\pi k}{3}\right)\left(1.2 \sin\frac{2\pi(k-1)}{3}\right) \\ &= (1.2)^2 0.5 \cos\left(\frac{2\pi}{3}\right) = -0.36 \end{aligned} \tag{10.72}$$

(where we have used some trigonometric identities).

Thus $\mathbf{R}$ is

$$\mathbf{R} = \begin{bmatrix} 0.72 & -0.36 \\ -0.36 & 0.72 \end{bmatrix}. \tag{10.73}$$

The terms of $\mathbf{h}$ can be found in a similar manner. We will consider the top term in Eq. (10.67) first:

$$E[(s(k)+m(k))v(k)] = E[s(k)v(k)] + E[m(k)v(k)]. \tag{10.74}$$

Here the first term on the right is zero because $s(k)$ and $v(k)$ are independent and zero mean. The second term is also zero:

$$E[m(k)v(k)] = \frac{1}{3}\sum_{k=1}^{3}\left(0.12\ \sin\left(\frac{2\pi k}{3}+\frac{\pi}{2}\right)\right)\left(1.2\sin\frac{2\pi k}{3}\right) = 0 \tag{10.75}$$

Thus, the first element of $\mathbf{h}$ is zero.

Next consider the second element of $\mathbf{h}$:

$$E[(s(k)+m(k))v(k-1)] = E[s(k)v(k-1)] + E[m(k)v(k-1)]\ . \tag{10.76}$$

As with the first element of $\mathbf{h}$, the first term on the right is zero because $s(k)$ and $v(k-1)$ are independent and zero mean. The second term is evaluated as follows:

$$E[m(k)v(k-1)] = \frac{1}{3}\sum_{k=1}^{3}\left(0.12\ \sin\left(\frac{2\pi k}{3}+\frac{\pi}{2}\right)\right)\left(1.2\ \sin\frac{2\pi(k-1)}{3}\right) = -0.0624\ . \tag{10.77}$$

Thus, $\mathbf{h}$ is

$$\mathbf{h} = \begin{bmatrix} 0 \\ -0.0624 \end{bmatrix}. \tag{10.78}$$

The minimum mean square error solution for the weights is given by Eq. (10.18):

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} 0.72 & -0.36 \\ -0.36 & 0.72 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ -0.0624 \end{bmatrix} = \begin{bmatrix} -0.0578 \\ -0.1156 \end{bmatrix}. \tag{10.79}$$

Now, what kind of error will we have at the minimum solution? To find this error recall Eq. (10.12):

$$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}\ . \tag{10.80}$$

We have just found $\mathbf{x}^*$, $\mathbf{h}$ and $\mathbf{R}$, so we only need to find c:

$$c = E[t^2(k)] = E[(s(k)+m(k))^2] = E[s^2(k)] + 2E[s(k)m(k)] + E[m^2(k)]\ . \tag{10.81}$$

The middle term is zero because $s(k)$ and $m(k)$ are independent and zero mean. The first term, the mean squared value of the random signal, can be calculated as follows:

$$E[s^2(k)] = \frac{1}{0.4}\int_{-0.2}^{0.2} s^2 ds = \frac{1}{3(0.4)} s^3\Big|_{-0.2}^{0.2} = 0.0133\,. \tag{10.82}$$

The mean square value of the filtered noise is

$$E[m^2(k)] = \frac{1}{3}\sum_{k=1}^{3}\left\{0.12\,\sin\left(\frac{2\pi}{3}+\frac{\pi}{2}\right)\right\}^2 = 0.0072\,, \tag{10.83}$$

so that

$$c = 0.0133 + 0.0072 = 0.0205\,. \tag{10.84}$$

Substituting $\mathbf{x}^*$, $\mathbf{h}$ and $\mathbf{R}$ into Eq. (10.80), we find that the minimum mean square error is

$$F(\mathbf{x}^*) = 0.0205 - 2(0.0072) + 0.0072 = 0.0133\,. \tag{10.85}$$

The minimum mean square error is the same as the mean square value of the EEG signal. This is what we expected, since the "error" of this adaptive noise canceller is in fact the reconstructed EEG signal.

Figure 10.8 illustrates the trajectory of the LMS algorithm in the weight space with learning rate $\alpha = 0.1$. The system weights $w_{1,1}$ and $w_{1,2}$ in this simulation were initialized arbitrarily to 0 and -2, respectively. You can see from this figure that the LMS trajectory looks like a noisy version of steepest descent.

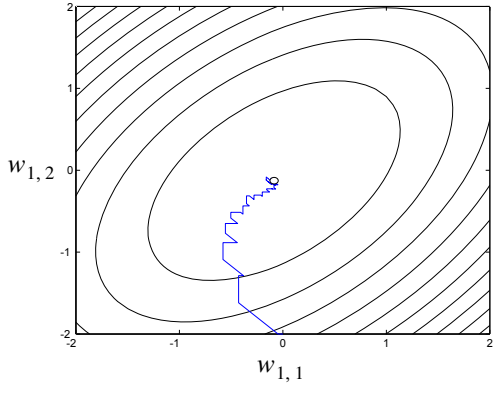


Figure 10.8 LMS Trajectory for $\alpha = 0.1$

Note that the contours in this figure reflect the fact that the eigenvalues and eigenvectors of the Hessian matrix ($\mathbf{A} = 2\mathbf{R}$) are

$$\lambda_1 = 2.16,\ \mathbf{z}_1 = \begin{bmatrix} -0.7071 \\ 0.7071 \end{bmatrix},\ \lambda_2 = 0.72,\ \mathbf{z}_2 = \begin{bmatrix} -0.7071 \\ -0.7071 \end{bmatrix}. \tag{10.86}$$

(Refer back to our discussion in Chapter 8 on the eigensystem of the Hessian matrix.)

If the learning rate is decreased, the LMS trajectory is smoother than that shown in Figure 10.8, but the learning proceeds more slowly. If the learning rate is increased, the trajectory is more jagged and oscillatory. In fact, as noted earlier in this chapter, if the learning rate is increased too much the system does not converge at all. The maximum stable learning rate is $\alpha < 2/2.16 = 0.926$.

In order to judge the performance of our noise canceller, consider Figure 10.9. This figure illustrates how the filter adapts to cancel the noise. The top graph shows the restored and original EEG signals. At first the restored signal is a poor approximation of the original EEG signal. It takes about 0.2 second (with $\alpha = 0.1$) for the filter to adjust to give a reasonable restored signal. The mean square difference between the original and restored signal over the last half of the experiment was 0.002. This compares favorably with the signal mean square value of 0.0133. The difference between the original and restored signal is shown in the lower graph.

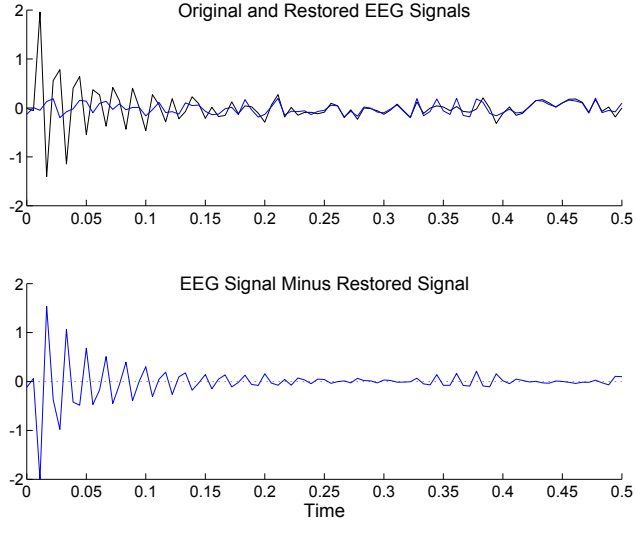


Figure 10.9 Adaptive Filter Cancellation of Contaminating Noise

You might wonder why the error does not go to zero. This is because the LMS algorithm is approximate steepest descent; it uses an estimate of the gradient, not the true gradient, to update the weights. The estimate of the gradient is a noisy version of the true gradient. This will cause the weights to continue to change slightly, even after the mean square error is at the minimum point. You can see this effect in Figure 10.8.

To experiment with the use of this adaptive noise cancellation filter, use the MATLAB® Neural Network Design Demonstration Adaptive Noise Cancellation (**nnd10nc**). *A more complex noise source and actual EEG data are used in the Demonstration* Electroencephalogram Noise Cancellation (**nnd10eeg**).

Echo Cancellation

Another very important practical application of adaptive noise cancellation is echo cancellation. Echoes are common in long distance telephone lines because of impedance mismatch at the "hybrid" device that forms the junction between the long distance line and the customer's local line. You may have experienced this effect on international telephone calls.

Figure 10.10 illustrates how an adaptive noise cancellation filter can be used to reduce these echoes [WiWi85]. At the end of the long distance line the incoming signal is sent to an adaptive filter, as well as to the hybrid device. The target output of the filter is the output of the hybrid. The filter thus tries to cancel the part of the hybrid output that is correlated with the input signal — the echo.

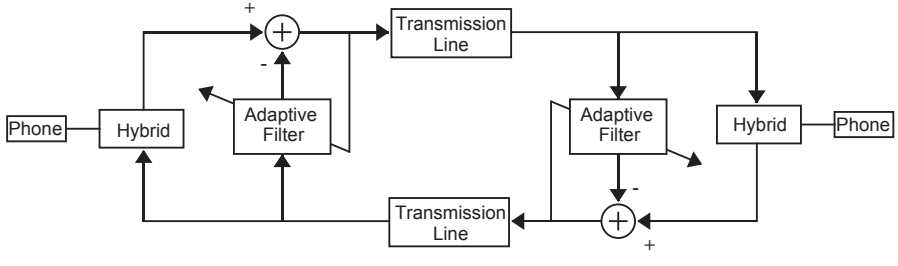
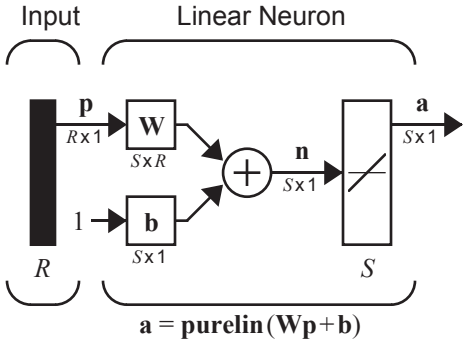


Figure 10.10 Echo Cancellation System

Summary of Results

ADALINE



Mean Square Error

$$F(\mathbf{x}) = E[e^2] = E[(t-a)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$$

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

$$c = E[t^2],\ \mathbf{h} = E[t\mathbf{z}] \text{ and } \mathbf{R} = E[\mathbf{z}\mathbf{z}^T]$$

Unique minimum, if it exists, is $\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$.

Where $\mathbf{x} = \begin{bmatrix} {}_1\mathbf{w} \\ b \end{bmatrix}$ and $\mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$.

LMS Algorithm

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

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k)$$

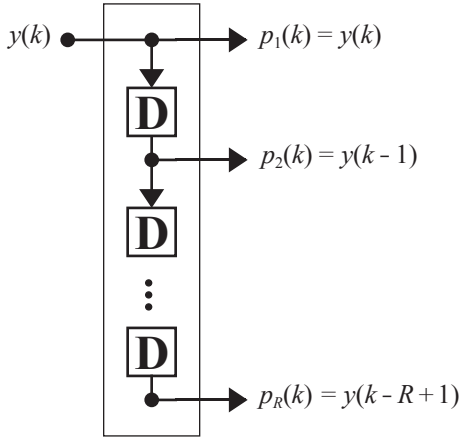
Convergence Point

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$$

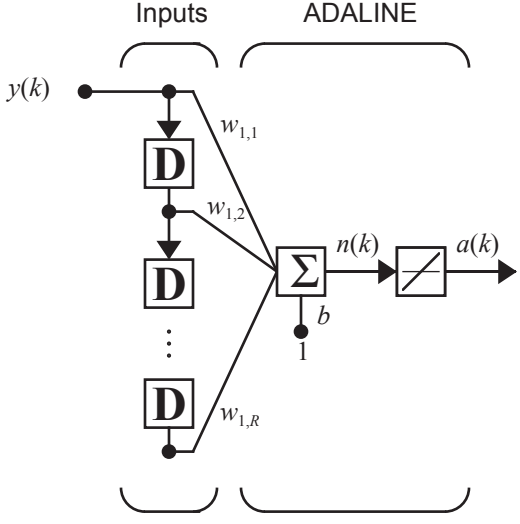
Stable Learning Rate

$0 < \alpha < 1/\lambda_{max}$ where λ_{max} is the maximum eigenvalue of $\mathbf{R}$

Tapped Delay Line



Adaptive Filter ADALINE



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

$$a(k) = purelin(\mathbf{W}\mathbf{p} + b) = \sum_{i=1}^{R} w_{1,i}\, y(k - i + 1) + b$$