the relationships between the clean signal and the corrupted signal both in power-spectrum and cepstrum domains based on such a model [2].

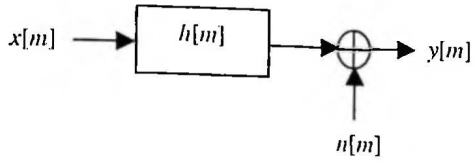


Figure 10.2 A model of the environment.

In the time domain, additive noise and linear filtering results in

$$y[m] = x[m] * h[m] + n[m] \tag{10.15}$$

It is convenient to express this in the frequency domain using the short-time analysis methods of Chapter 6. To do that, we window the signal, take a $2K$-point DFT in Eq. (10.15) and then the magnitude squared:

$$\begin{aligned} |Y(f_k)|^2 &= |X(f_k)|^2 |H(f_k)|^2 + |N(f_k)|^2 + 2\,\mathrm{Re}\{X(f_k)H(f_k)N^*(f_k)\} \\ &= |X(f_k)|^2 |H(f_k)|^2 + |N(f_k)|^2 + 2|X(f_k)||H(f_k)||N(f_k)|\cos(\theta_k) \end{aligned} \tag{10.16}$$

where $k = 0, 1, \cdots, K$, we have used upper case for frequency domain linear spectra, and θ_k is the angle between the filtered signal and the noise for bin k.

The expected value of the *cross-term* in Eq. (10.16) is zero, since $x[m]$ and $n[m]$ are statistically independent. In practice, this term is not zero for a given frame, though it is small if we average over a range of frequencies, as we often do when computing the popular mel-cepstrum (see Chapter 6). When using a filterbank, we can obtain a relationship for the energies at each of the M filters:

$$|Y(f_i)|^2 \approx |X(f_i)|^2 |H(f_i)|^2 + |N(f_i)|^2 \tag{10.17}$$

where it has been shown experimentally that this assumption works well in practice.

Equation (10.17) is also implicitly assuming that the length of $h[n]$, the filter's impulse response, is much shorter than the window length $2N$. That means that for filters with long reverberation times, Eq. (10.17) is inaccurate. For example, for $|N(f)|^2 = 0$, a window shift of T, and a filter's impulse response $h[n] = \delta[n-T]$, we have $Y_t[f_m] = X_{t-1}[f_m]$, i.e., the output spectrum at frame t does not depend on the input spectrum at that frame. This is a more serious assumption, which is why speech recognition systems tend to fail under long reverberation times.

Amazon/VB Assets
Exhibit 1012
Page 509

By taking logarithms in Eq. (10.17), and after some algebraic manipulation, we obtain

$$\ln|Y(f_i)|^2 \approx \ln|X(f_i)|^2 + \ln|H(f_i)|^2 + \ln\left(1+\exp\left(\ln|N(f_i)|^2 - \ln|X(f_i)|^2 - \ln|H(f_i)|^2\right)\right) \tag{10.18}$$

Since most speech recognition systems use cepstrum features, it is useful to see the effect of the additive noise and channel distortion directly on the cepstrum. To do that, let's define the following length-$(M+1)$ cepstrum vectors:

$$\begin{aligned}
\mathbf{x} &= \mathbf{C}\left(\ln|X(f_0)|^2 \quad \ln|X(f_1)|^2 \quad \cdots \quad \ln|X(f_M)|^2\right)\\
\mathbf{h} &= \mathbf{C}\left(\ln|H(f_0)|^2 \quad \ln|H(f_1)|^2 \quad \cdots \quad \ln|H(f_M)|^2\right)\\
\mathbf{n} &= \mathbf{C}\left(\ln|N(f_0)|^2 \quad \ln|N(f_1)|^2 \quad \cdots \quad \ln|N(f_M)|^2\right)\\
\mathbf{y} &= \mathbf{C}\left(\ln|Y(f_0)|^2 \quad \ln|Y(f_1)|^2 \quad \cdots \quad \ln|Y(f_M)|^2\right)
\end{aligned} \tag{10.19}$$

where $\mathbf{C}$ is the DCT matrix and we have used lower-case bold to represent cepstrum vectors.

Combining Eqs. (10.18) and (10.19) results in

$$\mathbf{y} = \mathbf{x} + \mathbf{h} + \mathbf{g}(\mathbf{n} - \mathbf{x} - \mathbf{h}) \tag{10.20}$$

where the nonlinear function $\mathbf{g}(\mathbf{z})$ is given by

$$\mathbf{g}(\mathbf{z}) = \mathbf{C}\ln\left(1+e^{\mathbf{C}^{-1}\mathbf{z}}\right) \tag{10.21}$$

Equations (10.20) and (10.21) say that we can compute the cepstrum of the corrupted speech if we know the cepstrum of the clean speech, the cepstrum of the noise, and the cepstrum of the filter. In practice, the DCT matrix $\mathbf{C}$ is not square, so that the dimension of the cepstrum vector is much smaller than the number of filters. This means that we are losing resolution when going back to the frequency domain, and thus Eqs. (10.20) and (10.21) represent only an approximation, though it has been shown to work reasonably well.

As discussed in Chapter 9, the distribution of the cepstrum of $\mathbf{x}$ can be modeled as a mixture of Gaussian densities. Even if we assume that $\mathbf{x}$ follows a Gaussian distribution, $\mathbf{y}$ in Eq. (10.20) is no longer Gaussian because of the nonlinearity in Eq. (10.21).

It is difficult to visualize the effect on the distribution, given the nonlinearity involved. To provide some insight, let's consider the frequency-domain version of Eq. (10.18) when no filtering is done, i.e., $H(f)=1$:

$$y = x + \ln\left(1+\exp(n-x)\right) \tag{10.22}$$

where x, n, and y represent the log-spectral energies of the clean signal, noise, and noisy signal, respectively, for a given frequency. Using simulated data, not real speech, we can

Amazon/VB Assets
Exhibit 1012
Page 510

analyze the effect of this transformation. Let's assume that both x and n are Gaussian random variables. We can use Monte Carlo simulation to draw a large number of points from those two Gaussian distributions and obtain the corresponding noisy values y using Eq. (10.22). Figure 10.3 shows the resulting distribution for several values of σ_x. We fixed $\mu_n = 0\,\text{dB}$, since it is only a relative level, and set $\sigma_n = 2\,\text{dB}$, a typical value. We also set $\mu_x = 25\,\text{dB}$ and see that the resulting distribution can be bimodal when σ_x is very large. Fortunately, for modern speech recognition systems that have many Gaussian components, σ_x is never that large and the resulting distribution is unimodal.

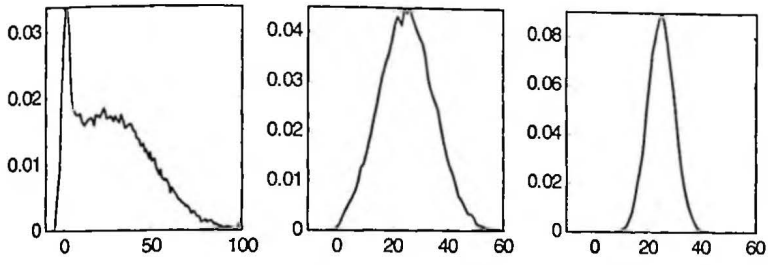


Figure 10.3 Distributions of the corrupted log-spectra y of Eq. (10.22) using simulated data. The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation of 2 dB. The distribution of the clean log-spectrum x is Gaussian with mean 25 dB and standard deviations of 25, 10, and 5 dB, respectively (the x-axis is expressed in dB). The first distribution is bimodal, whereas the other two are approximately Gaussian. Curves are plotted using Monte Carlo simulation.

Figure 10.4 shows the distribution of y for two values of μ_x, given the same values for the noise distribution, $\mu_n = 0\,\text{dB}$ and $\sigma_n = 2\,\text{dB}$, and a more realistic value for $\sigma_x = 5\,\text{dB}$. We see that the distribution is always unimodal, though not necessarily symmetric, particularly for low SNR ($\mu_x - \mu_n$).

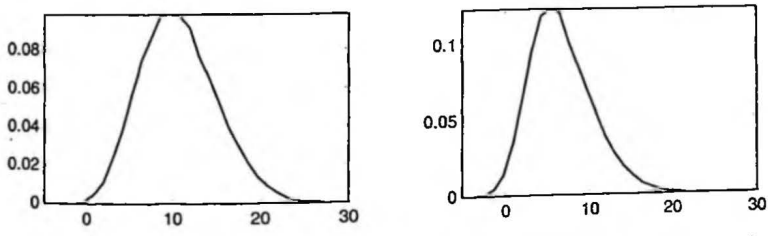


Figure 10.4 Distributions of the corrupted log-spectra y of Eq. (10.22) using simulated data. The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation of 2 dB. The distribution of the clean log-spectrum is Gaussian with standard deviation of 5 dB and means of 10 and 5 dB, respectively. The first distribution is approximately Gaussian while the second is nonsymmetric. Curves are plotted using Monte Carlo simulation.

Amazon/VB Assets
Exhibit 1012
Page 511

The distributions used in an HMM are mixtures of Gaussians so that, even if each Gaussian component is transformed into a non-Gaussian distribution, the composite distribution can be modeled adequately by another mixture of Gaussians. In fact, if you retrain the model using the standard Gaussian assumption on corrupted speech, you can get good results, so this approximation is not bad.

10.2. ACOUSTICAL TRANSDUCERS

Acoustical transducers are devices that convert the acoustic energy of sound into electrical energy (microphones) and vice versa (loudspeakers). In the case of a microphone this transduction is generally realized with a diaphragm, whose movement in response to sound pressure varies the parameters of an electrical system (a variable-resistance conductor, a condenser, etc.), producing a variable voltage that constitutes the microphone output. We focus on microphones because they play an important role in designing speech recognition systems.

There are near field or close-talking microphones, and far field microphones. Close-talking microphones, either head-mounted or telephone handsets, pick up much less background noise, though they are more sensitive to throat clearing, lip smacks, and breath noise. Placement of such a microphone is often very critical, since, if it is right in front of the mouth, it can produce pops in the signal with plosives such as /p/. Far field microphones can be lapel mounted or desktop mounted and pick up more background noise than near field microphones. Having a small but variable distance to the microphone could be worse than a larger but more consistent distance, because the corresponding HMM may have lower variability.

When used in speech recognition systems, the most important measurement is the signal-to-noise ratio (SNR), since the lower the SNR the higher the error rate. In addition, different microphones have different transfer functions, and even the same microphone offers different transfer functions depending on the distance between mouth and microphone. Varying noise and channel conditions are a challenge that speech recognition systems have to address, and in this chapter we present some techniques to combat them.

The most popular type of microphone is the condenser microphone. We shall study in detail its directionality patterns, frequency response, and electrical characteristics.

10.2.1. The Condenser Microphone

A *condenser microphone* has a capacitor consisting of a pair of metal plates separated by an insulating material called a dielectric (see Figure 10.5). Its capacitance C is given by

$$C = \varepsilon_0 \pi b^2 / h \tag{10.23}$$

Amazon/VB Assets
Exhibit 1012
Page 512

where ε_0 is a constant, b is the width of the plate, and h is the separation between the plates. If we polarize the capacitor with a voltage V_{cc}, it acquires a charge Q given by

$$Q = CV_{cc} \tag{10.24}$$

One of the plates is free to move in response to changes in sound pressure, which results in a change in the plate separation Δh, thereby changing the capacitance and producing a change in voltage $\Delta V = \Delta h V_{cc} / h$. Thus, the sensitivity[4] of the microphone depends on the polarizing voltage V_{cc}, which is why this voltage can often be 100 V or more.

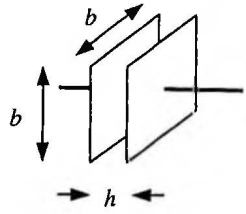


Figure 10.5 A diagram of a condenser microphone.

Electret microphones are a type of condenser microphones that do not require a special polarizing voltage V_{cc}, because a charge is impressed on either the diaphragm or the back plate during manufacturing and it remains for the life of the microphone. Electret microphones are light and, because of their small size, they offer good responses at high frequencies.

From the electrical point of view, a microphone is equivalent to a voltage source $v(t)$ with an impedance Z_M, as shown in Figure 10.6. The microphone is connected to a preamplifier which has an equivalent impedance R_L.

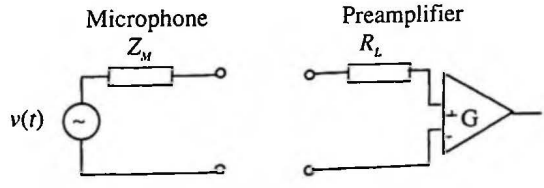


Figure 10.6 Electrical equivalent of a microphone.

[4] The sensitivity of a microphone measures the *open-circuit voltage* of the electric signal the microphone delivers for a sound wave for a given sound pressure level, often 94 dB SPL, when there is no load or a high impedance. This voltage is measured in dBV, where the 0-dB reference is 1 V rms.

Amazon/VB Assets
Exhibit 1012
Page 513

From Figure 10.6 we can see that the voltage on R_L is

$$v_R(t) = v(t)\frac{R_L}{(R_M + R_L)} \tag{10.25}$$

Maximization of $v_R(t)$ in Eq. (10.25) results in $R_L = \infty$, or in practice $R_L >> R_M$, which is called *bridging*. Thus, for highest sensitivity the impedance of the amplifier has to be at least 10 times higher than that of the microphone. If the microphone is connected to an amplifier with lower impedance, there is a *load loss* of signal level. Most low-impedance microphones are labeled as 150 ohms, though the actual values may vary between 100 and 300. Medium impedance is 600 ohms and high impedance is 600–10,000 ohms. In practice, the microphone impedance is a function of frequency. Signal power is measured in dBm, where the 0-dB reference corresponds to 1 mW dissipated in a 600-ohm resistor. Thus, 0 dBm is equivalent to 0.775 V.

Since the output impedance of a condenser microphone is very high (~ 1 Mohm), a JFET transistor must be coupled to lower the equivalent impedance. Such a transistor needs to be powered with DC voltage through a different wire, as in Figure 10.7. A standard sound card has a jack with the audio on the tip, ground on the sleeve, DC bias V_{DD} on the ring, and a medium impedance. When using *phantom power*, the V_{CC} bias is provided directly in the audio signal, which must be *balanced* to ground.

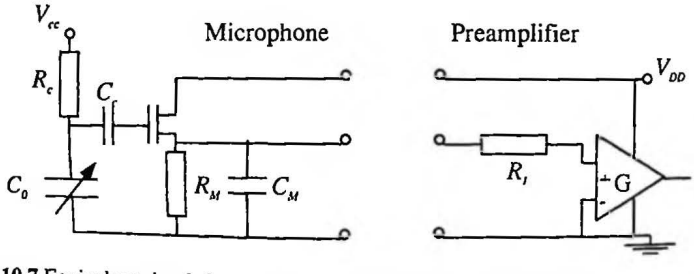


Figure 10.7 Equivalent circuit for a condenser microphone with DC bias on a separate wire.

It is important to understand how noise affects the signal of a microphone. If thermal noise arises in the resistor R_L, it will have a power

$$P_N = 4kTB \tag{10.26}$$

where $k = 1.38 \times 10^{-23}$ J/K is the Bolzmann's constant, T is the temperature in °K, and B is the bandwidth in Hz. The thermal noise in Eq. (10.26) at room temperature ($T = 297$°K) and for a bandwidth of 4 kHz is equivalent to –132 dBm. In practice, the noise is significantly higher than this because of preamplifier noise, radio-frequency noise and electromagnetic interference (poor grounding connections). It is, thus, important to keep the signal path between the microphone and the preamp as short as possible to avoid extra noise. It is desir-

Amazon/VB Assets
Exhibit 1012
Page 514

able to have a microphone with low impedance to decrease the effect of noise due to radio-frequency interference, and to decrease the signal loss if long cables are used. Most microphones specify their SNR and range where they are linear (dynamic range). For condenser microphones, a power supply is necessary (DC bias required). Microphones with balanced output (the signal appears across two inner wires not connected to ground, with the shield of the cable connected to ground) are more resistant to radio frequency interference.

10.2.2. Directionality Patterns

A microphone's directionality pattern measures its sensitivity to a particular direction. Microphones may also be classified by their directional properties as *omnidirectional* (or *non-directional*) and *directional*, the latter subdivided into bidirectional and unidirectional, based upon their response characteristics.

10.2.2.1. Omnidirectional Microphones

By definition, the response of an omnidirectional microphone is independent of the direction from which the encroaching sound wave is coming. Figure 10.8 shows the polar response of an omnidirectional mike. A microphone's *polar response*, or *pickup pattern*, graphs its output voltage for an input sound source with constant level at various angles around the mic. Typically, a polar response assumes a preferred direction, called the major axis or front of the microphone, which corresponds to the direction at which the microphone is most sensitive. The front of the mike is labeled as zero degrees on the polar plot, but since an omnidirectional mic has no particular direction at which it is the most sensitive, the omnidirectional mike has no true front and hence the zero-degree axis is arbitrary. Sounds coming from any direction around the microphone are picked up equally. Omnidirectional microphones provide no noise cancellation.

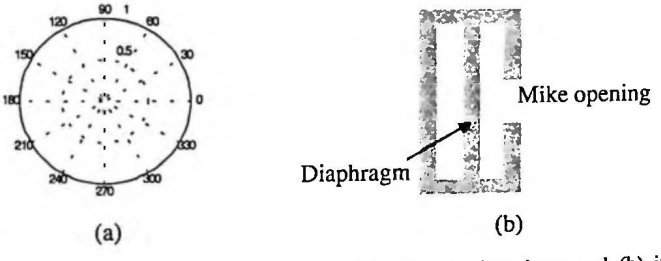


Figure 10.8 (a) Polar response of an ideal omnidirectional microphone and (b) its cross section.

Figure 10.8 shows the mechanics of the ideal[5] omnidirectional condenser microphone. A sound wave creates a pressure all around the microphone. The pressure enters the opening of the mike and the diaphragm moves. An electrical circuit converts the diaphragm move-

[5] Ideal omnidirectional microphones do not exist.

Amazon/VB Assets
Exhibit 1012
Page 515

ment into an electrical voltage, or response. Sound waves impinging on the mike create a pressure at the opening regardless of the direction from which they are coming; therefore we have a nondirectional, or omnidirectional, microphone. As we have seen in Chapter 2, if the source signal is $Be^{j\omega t}$, the signal at a distance r is given by $(A/r)e^{j\omega t}$ independently of the angle.

This is the most inexpensive of the condenser microphones, and it has the advantage of a flat frequency response that doesn't change with the angle or distance to the microphone. On the other hand, because of its uniform polar pattern, it picks up not only the desired signal but also noise from any direction. For example, if a pair of speakers is monitoring the microphone output, the sound from the speakers can reenter the microphone and create an undesirable sound called *feedback*.

10.2.2.2. Bidirectional Microphones

The bidirectional microphone is a *noise-canceling* microphone; it responds less to sounds incident from the sides. The bidirectional mike utilizes the properties of a gradient microphone to achieve its noise-canceling polar response. You can see how this is accomplished by looking at the diagram of a simplified gradient bidirectional condenser microphone, as shown in Figure 10.9. A sound impinging upon the front of the microphone creates a pressure at the front opening. A short time later, this same sound pressure enters the back of the microphone. The sound pressure never arrives at the front and back at the same time. This creates a displacement of the diaphragm and, just as with the omnidirectional mike, a corresponding electrical signal. For sounds impinging from the side, however, the pressure from an incident sound wave at the front opening is identical to the pressure at the back. Since both openings lead to one side of the diaphragm, there is no displacement of the diaphragm, and the sound is not reproduced.

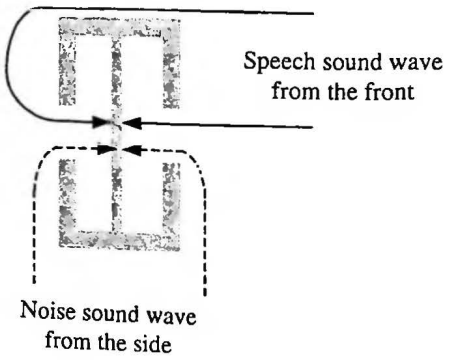


Figure 10.9 Cross section of an ideal bidirectional microphone.

Amazon/VB Assets
Exhibit 1012
Page 516

To compute the polar response of this gradient microphone let's make the approximation of Figure 10.10, where the microphone signal is the difference between the signal at the front and rear of the diaphragm, the separation between plates is $2d$, and r is the distance between the source and the center of the microphone.

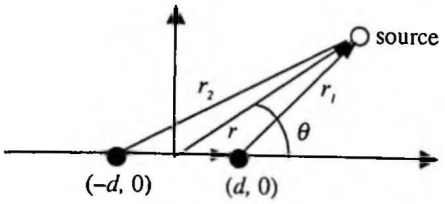


Figure 10.10 Approximation to the noise-canceling microphone of Figure 10.9.

You can see that r_1, the distance between the source and the front of the diaphragm, is the norm of the vector specifying the source location minus the vector specifying the location of the front of the diaphragm

$$r_1 = \left| re^{j\theta} - d \right| \tag{10.27}$$

Similarly, you obtain the distance between the source and the rear of the diaphragm

$$r_2 = \left| re^{j\theta} + d \right| \tag{10.28}$$

The source arrives at the front of the diaphragm with a delay $\delta_1 = r_1 / c$, where c is the speed of sound in air. Similarly, the delay to the rear of the diaphragm is $\delta_2 = r_2 / c$. If the source is a complex exponential $e^{j\omega t}$, the difference signal between the front and rear is given by

$$x(t) = \frac{A}{r_1} e^{j2\pi f(t-\delta_1)} - \frac{A}{r_2} e^{j2\pi f(t-\delta_2)} = \frac{A}{r} e^{j2\pi ft} G(f,\theta) \tag{10.29}$$

where A is a constant and, using Eqs. (10.27), (10.28) and (10.29), the gain $G(f,\theta)$ is given by

$$G(f,\theta) = \frac{e^{-j2\pi \left| e^{j\theta} - \lambda \right| \tau f}}{\left| e^{j\theta} - \lambda \right|} - \frac{e^{-j2\pi \left| e^{j\theta} + \lambda \right| \tau f}}{\left| e^{j\theta} + \lambda \right|} \tag{10.30}$$

where we have defined $\lambda = d / r$ and $\tau = r / c$.

The magnitude of Eq. (10.30) is used to plot the polar response of Figure 10.11. As can be seen by the plot, the pattern resembles a figure eight. The bidirectional mike has an interchangeable front and back, since the response has a maximum in two opposite directions. In practice, this bidirectional microphone is an ideal case, and the polar response has to be measured empirically.

Amazon/VB Assets
Exhibit 1012
Page 517

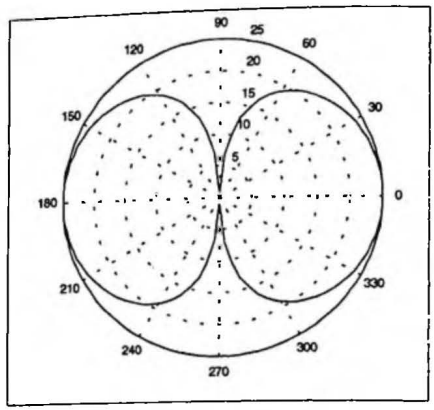


Figure 10.11 Polar response of a bidirectional microphone obtained through Eq. (10.30) with $d = 1$ cm, $r = 50$ cm, $c = 33{,}000$ cm/s, and $f = 1000$ Hz.

According to the idealized model, the frequency response of omnidirectional microphones is constant with frequency, and this approximately holds in practice for real omnidirectional microphones. On the other hand, the polar pattern of directional microphones is not constant with frequency. Clearly it is a function of frequency, as can be seen in Eq. (10.30). In fact, the frequency response of a bidirectional microphone at 0° is shown in Figure 10.12 for both near field and far field conditions.

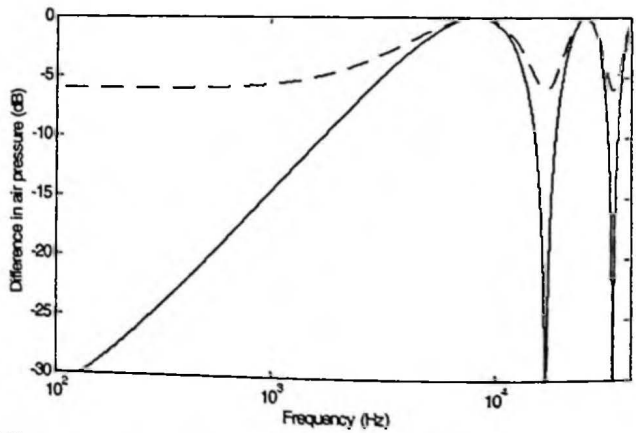


Figure 10.12 Frequency response of a bidirectional microphone with $d = 1$ cm at 0° obtained through Eq. (10.30). The larger the distance between plates, the lower the frequency of the maxima. The highest values are obtained for 8250 Hz and 24,750 Hz and the null for 16,500 Hz. The solid line corresponds to far field conditions ($\lambda = 0.02$) and the dotted line to near field conditions ($\lambda = 0.5$).

Amazon/VB Assets
Exhibit 1012
Page 518

It can be shown, after taking the derivative of $G(f,0)$ in Eq. (10.30) and equating to zero, that the maxima are given by

$$f_n = \frac{c}{4d}(2n-1) \tag{10.31}$$

with $n = 1, 2, \cdots$. We can observe from Eq. (10.31) that the larger the width of the diaphragm, the lower the first maximum.

The increase in frequency response, or sensitivity, in the near field, compared to the far field, is a measure of noise cancellation. Consequently the microphone is said to be noise canceling. The microphone is also referred to as a differential or gradient microphone, since it measures the gradient (difference) in sound pressure between two points in space. The boost in low-frequency response in the near field is also referred to as the *proximity effect*, often used by singers to boost their bass levels by getting the microphone closer to their mouths.

By evaluating Eq. (10.30) it can be seen that low-frequency sounds in a bidirectional microphone are not reproduced as well as higher frequencies, leading to a *thin* sounding mike.

Let's interpret Figure 10.12. The net sound pressure between these two points, separated by a distance $D = 2d$, is influenced by two factors: phase shift and inverse square law.

The influence of the sound-wave phase shift is less at low frequencies than at high, because the distance D between the front and rear port entries becomes a small fraction of the low-frequency wavelength. Therefore, there is little phase shift between the ports at low frequencies, as the opposite sides of the diaphragm receive nearly equal amplitude and phase. The result is slight diaphragm motion and a weak microphone output signal. At higher frequencies, the distance D between sound ports becomes a larger fraction of the wavelength. Therefore, more phase shift exists across the diaphragm. This causes a higher microphone output.

The pressure difference caused by phase shift rises with frequency at a rate of 20 dB per decade. As the frequency rises to where the microphone port spacing D equals half a wavelength, the net pressure is at its maximum. In this situation, the diaphragm movement is also at its maximum, since the front and rear see equal amplitude but in opposite polarities of the wave front. This results in a peak in the microphone frequency response, as illustrated in Figure 10.12. As the frequency continues to rise to where the microphone port spacing D equals one complete wavelength, the net pressure is at its minimum. Here, the diaphragm does not move at all, since the front and rear sides see equal amplitude at the same polarity of the wave front. This results in a dip in the microphone frequency response, as shown in Figure 10.12.

Amazon/VB Assets
Exhibit 1012
Page 519

A second factor creating a net pressure difference across the diaphragm is the impact of the inverse square law. If the sound-pressure difference between the front and rear ports of a noise-canceling microphone were measured near the sound source and again further from the source, the near field measurement would be greater than the far field. In other words, the microphone's net pressure difference and, therefore, output signal, is greater in the near sound field than in the far field. The inverse-square-law effect is independent of frequency. The net pressure that causes the diaphragm to move is a combination of both the phase shift and inverse-square-law effect. These two factors influence the frequency response of the microphone differently, depending on the distance to the sound source. For distant sound, the influence of the net pressure difference from the inverse-square-law effect is weaker than the phase-shift effect; thus, the rising 20-dB-per-decade frequency response dominates the total frequency response. As the microphone is moved closer to the sound source, the influence of the net pressure difference from the inverse square law is greater than that of the phase shift; thus the total microphone frequency response is largely flat.

The difference in near field to far field frequency response is a characteristic of all noise-canceling microphones and applies equally to both acoustic and electronic types.

10.2.2.3. Unidirectional Microphones

Unidirectional microphones are designed to pick-up the speaker's voice by directing the audio reception toward the speaker, focusing on the desired input and rejecting sounds emanating from other directions that can negatively impact clear communications, such as computer noise from fans or other sounds.

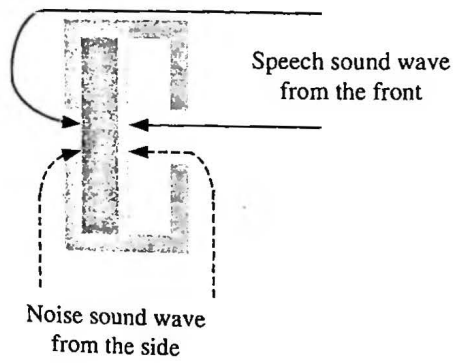


Figure 10.13 Cross section of a unidirectional microphone.

Amazon/VB Assets
Exhibit 1012
Page 520

Figure 10.13 shows the cross-section of a unidirectional microphone, which also relies upon the principles of a gradient microphone. Notice that the unidirectional mic looks similar to the bidirectional, except that there is a resistive material (often cloth or foam) between the diaphragm and the opening of one end. The material's resistive properties *slow down* the pressure on its path from the back opening to the diaphragm. If the additional delay through the back plate is given by τ_0, the gain can be given by

$$G(f,\theta) = \frac{e^{-j2\pi|e^{j\theta}-\lambda|rf}}{|e^{j\theta}-\lambda|} - \frac{e^{-j2\pi(\tau_0+|e^{j\theta}+\lambda|r)f}}{|e^{j\theta}+\lambda|} \tag{10.32}$$

which was obtained by modifying Eq. (10.30). Unidirectional microphones have the greatest response to sound waves impinging from one direction, typically referred to as the front, or major axis of the microphone. One typical response of a unidirectional microphone is the *cardioid* pattern shown in the polar plot of Figure 10.14, plotted from Eq. (10.32). The frequency response at 0° is similar to that of Figure 10.12. Because the cardioid pattern of polar response is so popular among them, unidirectional mikes are often referred to as cardioid mikes.

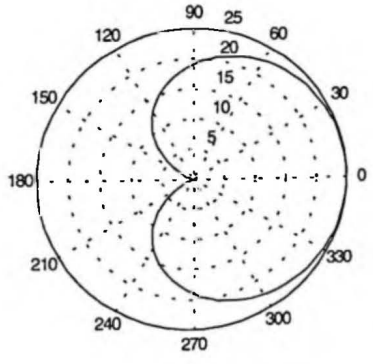


Figure 10.14 Polar response of a unidirectional microphone. The polar response was obtained through Eq. (10.32) with $d = 1$ cm, $r = 50$ cm, $c = 33{,}000$ cm/s, $f = 1$ kHz, and $\tau_0 = 0.06$ ms.

Equation (10.32) was derived under a simplified schematic based on Figure 10.10, which is an idealized model so that, in practice, the polar response of a real microphone has to be measured empirically. The frequency response and polar pattern of a commercial microphone are shown in Figure 10.15.

Amazon/VB Assets
Exhibit 1012
Page 521

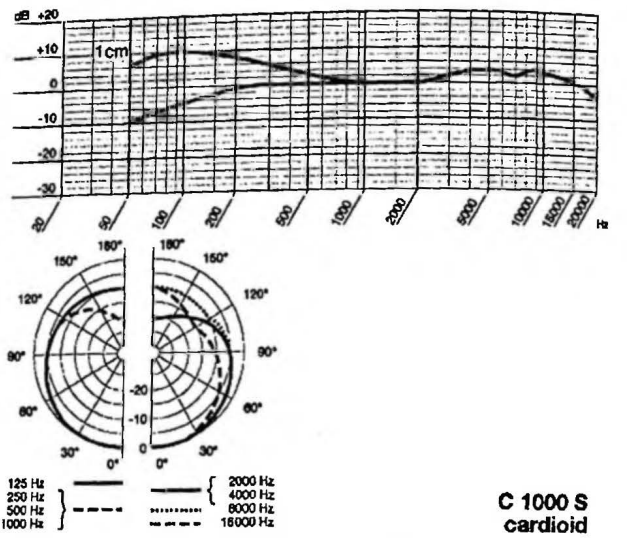


Figure 10.15 Characteristics of an AKG C1000S cardioid microphone: (top) frequency response for near and far field conditions (note the proximity effect) and (bottom) polar pattern for different frequencies.

Although this noise cancellation decreases the overall response to sound pressure (*sensitivity*) of the microphone, the directional and frequency-response improvements far outweigh the lessened sensitivity. It is particularly well suited for use as a desktop mic or as part of an embedded microphone in a laptop or desktop computer. Unidirectional microphones achieve superior noise-rejection performance over omnidirectionals. Such performance is necessary for clean audio input and for audio signal processing algorithms such as acoustic echo cancellation, which form the core of speakerphone applications.

10.2.3. Other Transduction Categories

In a *passive microphone*, sound energy is directly converted to electrical energy, whereas an *active microphone* requires an external energy source that is modulated by the sound wave. Active transducers thus require phantom power, but can have higher sensitivity.

We can also classify microphones according to the physical property to which the sound wave responds. A *pressure microphone* has an electrical response that corresponds to the pressure in a sound wave, while a *pressure gradient microphone* has a response corresponding to the difference in pressure across some distance in a sound wave. A pressure microphone is a fine reproducer of sound, but a gradient microphone typically has a response greatest in the direction of a desired signal or talker and rejects undesired background sounds. This is particularly beneficial in applications that rely upon the reproduction of only a desired signal, where any undesired signal entering the reproduction severely degrades performance. Such is the case in voice recognition or speakerphone applications.

Amazon/VB Assets
Exhibit 1012
Page 522

In terms of the mechanism by which they create an electrical signal corresponding to the sound wave they detect, microphones are classified as *electromagnetic*, *electrostatic*, and *piezoelectric*. *Dynamic* microphones are the most popular type of electromagnetic microphone and *condenser* microphones the most popular type of electrostatic microphone.

Electromagnetic microphones induce voltage based on a varying magnetic field. *Ribbon microphones* are a type of electromagnetic microphones that employ a thin metal ribbon suspended between the poles of a magnet. *Dynamic* microphones are electromagnetic microphones that employ a moving coil suspended by a light diaphragm (see Figure 10.16), acting like a speaker but in reverse. The diaphragm moves with changes in sound pressure, which in turns moves the coil, which causes current to flow as lines of flux from the magnet are cut. Dynamic microphones need no batteries or power supply, but they deliver low signal levels that need to be preamplified.

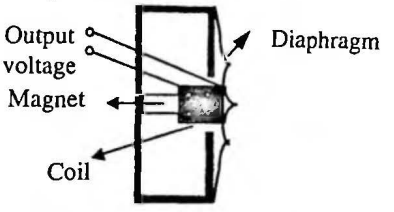


Figure 10.16 Dynamic microphone schematics.

Piezoresistive and piezoelectric microphones are based on the variation of electric resistance of their sensor induced by changes in sound pressure. *Carbon button microphones* consist of a small cylinder packed with tiny granules of carbon that, when compacted by sound pressure, reduce the electric resistance. Such microphones, often used in telephone handsets, offer a worse frequency response than condenser microphones, and lower dynamic range.

10.3. Adaptive Echo Cancellation (AEC)

If a spoken language system allows the user to talk while speech is being output through the loudspeakers, the microphone picks up not only the user's voice, but also the speech from the loudspeaker. This problem may be avoided with a *half-duplex* system that does not listen when a signal is being played through the loudspeaker, though such systems offer an unnatural user experience. On the other hand, a *full-duplex* system that allows *barge-in* by the user to interrupt the system offers a better user experience. For barge-in to work, the signal played through the loudspeaker needs to be canceled. This is achieved with echo cancellation (see Figure 10.17), as discussed in this section.

In hands-free conferencing the local user's voice is output by the remote loudspeaker, whose signal is captured by the remote microphone and after some delay is output by the local loudspeaker. People are tolerant to these echoes if either they are greatly attenuated or the delay is short. Perceptual studies have shown that the longer the delay, the greater the attenuation needed for user acceptance.

Amazon/VB Assets
Exhibit 1012
Page 523

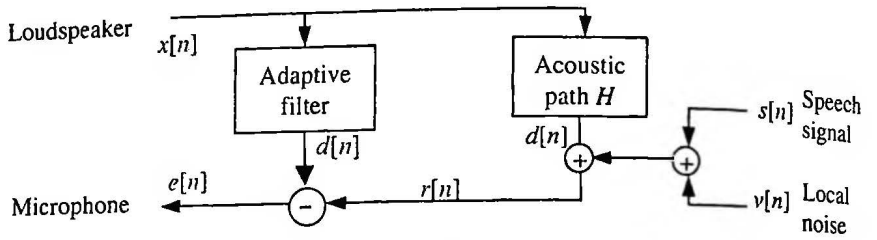


Figure 10.17 Block diagram of an echo-canceling application. $x[n]$ represents the signal from the loudspeaker, $s[n]$ the speech signal, $v[n]$ the local background noise, and $e[n]$ the signal that goes to the microphone.

The use of *echo cancellation* is mandatory in telephone communications and hands-free conferencing when it is desired to have full-duplex voice communication. This is particularly important when the call is routed through a satellite that can have delays larger than 200 ms. A block diagram is shown in Figure 10.18.

In Figure 10.17, the return signal $r[n]$, assuming no local noise, is the sum

$$r[n] = d[n] + s[n] \tag{10.33}$$

where $s[n]$ is the speech signal and $d[n]$ is the attenuated and possibly distorted version of the loudspeaker's signal $x[n]$. The purpose of the echo canceler is to remove the echo $d[n]$ from the return signal $r[n]$, which is done by means of an adaptive FIR filter whose coefficients are computed to minimize the energy of the canceled signal $e[n]$. The filter coefficients are reestimated *adaptively* to track slowly changing line conditions.

This problem is essentially that of adaptive filtering *only* when $s[n] = 0$, or in other words when the user is silent. For this reason, you have to implement a *double-talk detection* module that detects when the speaker is silent. This is typically feasible because the echo $d[n]$ is usually small, and if the return signal $r[n]$ has high energy it means that the user is

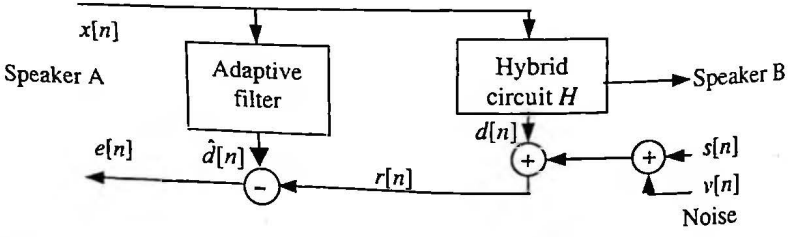


Figure 10.18 Block diagram of echo canceling for a telephone communication. $x[n]$ represents the remote call signal, $s[n]$ the local outgoing signal. The hybrid circuit H does a 2-4 wire conversion and is nonideal because of impedance mismatches.

Amazon/VB Assets
Exhibit 1012
Page 524

not silent. Errors in double-talk detection result in divergence of the filter, so it is generally preferable to be conservative in the decision and when in doubt not adapt the filter coefficients. Initialization could be done by sending a known signal with white spectrum. The quality of the filtering is measured by the so-called *echo-return loss enhancement* (ERLE):

$$ERLE(dB) = 10\log_{10}\frac{E\{d^2[n]\}}{E\{(d[n]-\hat{d}[n])^2\}} \tag{10.34}$$

The filter coefficients are chosen to maximize the ERLE. Since the telephone-line characteristics, or the acoustic path (due to speaker movement), can change over time, the filter is often adaptive. Another reason for adaptive filters is that reliable ERLE maximization requires a large number of samples, and such a delay is not tolerable.

In the following sections, we describe the fundamentals of adaptive filtering. While there are some nonlinear adaptive filters, the vast majority are linear FIR filters, with the LMS algorithm being the most important. We introduce the LMS algorithm, study its convergence properties, and present two extensions: the normalized LMS algorithm and transform-domain LMS algorithms.

10.3.1. The LMS Algorithm

Let's assume that a desired signal $d[n]$ is generated from an input signal $x[n]$ as follows

$$d[n] = \sum_{k=0}^{L-1} g_k x[n-k] + u[n] = \mathbf{G}^T\mathbf{X}[n] + u[n] \tag{10.35}$$

with $\mathbf{G} = \{g_0, g_1, \cdots g_{L-1}\}$, the input signal vector $\mathbf{X}[n] = \{x[n], x[n-1], \cdots x[n-L+1]\}$, and $u[n]$ being noise that is independent of $x[n]$.

We want to estimate $d[n]$ in terms of the sum of previous samples of $x[n]$. To do that we define the estimate signal $y[n]$ as

$$y[n] = \sum_{k=0}^{L-1} w_k[n]x[n-k] = \mathbf{W}^T[n]\mathbf{X}[n] \tag{10.36}$$

where $\mathbf{W}[n] = \{w_0[n], w_1[n], \cdots w_{L-1}[n]\}$ is the time-dependent coefficient vector. The instantaneous error between the desired and the estimated signal is given by

$$e[n] = d[n] - \mathbf{W}^T[n]\mathbf{X}[n] \tag{10.37}$$

The *least mean square* (LMS) algorithm updates the value of the coefficient vector in the steepest descent direction

Amazon/VB Assets
Exhibit 1012
Page 525

$$\mathbf{W}[n+1] = \mathbf{W}[n] + \varepsilon e[n]\mathbf{X}[n] \tag{10.38}$$

where ε is the step size. This algorithm is very popular because of its simplicity and effectiveness [58].

10.3.2. Convergence Properties of the LMS Algorithm

The choice of ε is important: if it is too small, the adaptation rate will be slow and it might not even track the nonstationary trends of $x[n]$, whereas if ε is too large, the error might actually increase. We analyze the conditions under which the LMS algorithm converges.

Let's define the error in the coefficient vector $\mathbf{V}[n]$ as

$$\mathbf{V}[n] = \mathbf{G} - \mathbf{W}[n] \tag{10.39}$$

and combine Eqs. (10.35), (10.37), (10.38), and (10.39) to obtain

$$\mathbf{V}[n+1] = \mathbf{V}[n] - \varepsilon\mathbf{X}[n]\mathbf{X}^T[n]\mathbf{V}[n] - \varepsilon u[n]\mathbf{X}[n] \tag{10.40}$$

Taking expectations in Eq. (10.40) results in

$$E\{\mathbf{V}[n+1]\} = E\{\mathbf{V}[n]\} - \varepsilon E\{\mathbf{X}[n]\mathbf{X}^T[n]\mathbf{V}[n]\} \tag{10.41}$$

where we have assumed that $u[n]$ and $x[n]$ are independent and that either is a zero-mean process. Finally, we express the autocorrelation of $\mathbf{X}[n]$ as

$$\mathbf{R}_{xx} = E\{\mathbf{X}[n]\mathbf{X}^T[n]\} = \mathbf{Q}\Lambda\mathbf{Q}^T \tag{10.42}$$

where $\mathbf{Q}$ is a matrix of its eigenvectors and Λ is a diagonal matrix of its eigenvalues $\{\lambda_0, \lambda_1, \cdots, \lambda_{L-1}\}$, which are all real valued because of the symmetry of $\mathbf{R}_{xx}$.

Although we know that $\mathbf{X}[n]$ and $\mathbf{V}[n]$ are not statistically independent, we assume in this section that they are, so that we can obtain some insight on the convergence properties. With this assumption, Eq. (10.41) can be expressed as

$$E\{\mathbf{V}[n+1]\} = E\{\mathbf{V}[n]\}(1 - \varepsilon\mathbf{R}_{xx}) \tag{10.44}$$

which, applied recursively, leads to

$$E\{\mathbf{V}[n+1]\} = E\{\mathbf{V}[0]\}(1 - \varepsilon\mathbf{R}_{xx})^n \tag{10.45}$$

Amazon/VB Assets
Exhibit 1012
Page 526

Using Eqs. (10.39) and (10.42) in (10.45), we can express the $(i+1)$th element of $E\{\mathbf{W}[n]\}$ as

$$E\{w_i[n]\} = g_i + \sum_{j=0}^{L-1} q_{ij}(1-\varepsilon\lambda_j)^n E\{\tilde{v}_i[0]\} \tag{10.46}$$

where q_{ij} is the $(i+1, j+1)$th element of the eigenvector matrix $\mathbf{Q}$, and $\tilde{v}_i[n]$ is the rotated coefficient error vector defined as

$$\tilde{\mathbf{V}}[n] = \mathbf{Q}^T \mathbf{V}[n] \tag{10.47}$$

From Eq. (10.46) we see that the mean value of the LMS filter coefficients converges exponentially to the true value if

$$0 < \varepsilon < 1/\lambda_j \tag{10.48}$$

so that the adaptation constant ε must be determined from the largest eigenvalue of $\mathbf{X}[n]$ for the mean LMS algorithm to converge.

10.3.3. Normalized LMS Algorithm

In practice, mean convergence doesn't tell us the nature of the fluctuations that the coefficients experience. Analysis of the variance of $\mathbf{V}[n]$ together with some more approximations result in mean-squared convergence if

$$0 < \varepsilon < \frac{K}{L\sigma_x^2} \tag{10.49}$$

with $\sigma_x^2 = E\{x^2[n]\}$ being the input signal power and K a constant that depends weakly on the nature of the input signal statistics but not on its power.

Because of the inaccuracies of the independence assumptions above, a rule of thumb used in practice to determine the adaptation constant ε is

$$0 < \varepsilon < \frac{0.1}{L\sigma_x^2} \tag{10.50}$$

The choice of largest value for ε in Eq. (10.49) makes the LMS algorithm track nonstationary variations in x fastest, and achieve faster convergence. On the other hand, the misadjustment of the filter coefficients increases as both the filter length L and adaptation constant ε increase. For this reason, often the adaptation constant can be made a function of

Amazon/VB Assets
Exhibit 1012
Page 527

n ($\varepsilon[n]$), with larger values at first and smaller values once convergence has been determined.

The *normalized LMS algorithm* (NLMS) uses the result of Eq. (10.49) and, therefore, defines a normalized step size

$$\varepsilon[n] = \frac{\varepsilon}{\delta + L\hat{\sigma}_x^2[n]} \tag{10.51}$$

where the constant δ avoids a division by 0 and $\hat{\sigma}_x^2[n]$ is an estimate of the input signal power, which is typically done with an exponential window

$$\hat{\sigma}_x^2[n] = (1-\beta)\hat{\sigma}_x^2[n-1] + \beta x^2[n] \tag{10.52}$$

or a sliding rectangular window

$$\hat{\sigma}_x^2[n] = \frac{1}{N}\sum_{i=0}^{N-1} x^2[n-i] = \hat{\sigma}_x^2[n-1] + \frac{1}{N}\left(x^2[n] - x^2[n-N]\right) \tag{10.53}$$

where both β and N control the effective memory of the estimators in Eqs. (10.52) and (10.53), respectively. Finally, we need to pick ε so that $0 < \varepsilon < 2$ to assure convergence. Choice of the NLMS algorithm simplifies the selection of ε, and the NLMS often converges faster than the LMS algorithm in practical situations.

10.3.4. Transform-Domain LMS Algorithm

As discussed in Section 10.3.2, convergence of the LMS algorithm is determined by the largest eigenvalue of the input. Since complex exponentials are approximate eigenvectors for LTI systems, the LMS algorithm's convergence is dominated by the frequency band with largest energy, and convergence in other frequency bands is generally much slower. This is the rationale for the *subband LMS algorithm,* which performs independent LMS algorithms for different frequency bands, as proposed by Boll [14].

The *block LMS* (BLMS) algorithm keeps the coefficients unchanged for a block k of L samples

$$\mathbf{W}[k+1] = \mathbf{W}[k] + \varepsilon\sum_{m=0}^{L-1} e[kL+m]\mathbf{X}[kL+m] \tag{10.54}$$

which is represented by a linear convolution and therefore can be implemented efficiently using length-$2N$ FFTs according to overlap-save method of Figure 10.19. Notice that implementing a linear convolution with a circular convolution operator such as the FFT requires the use of the dashed box.

Amazon/VB Assets
Exhibit 1012
Page 528

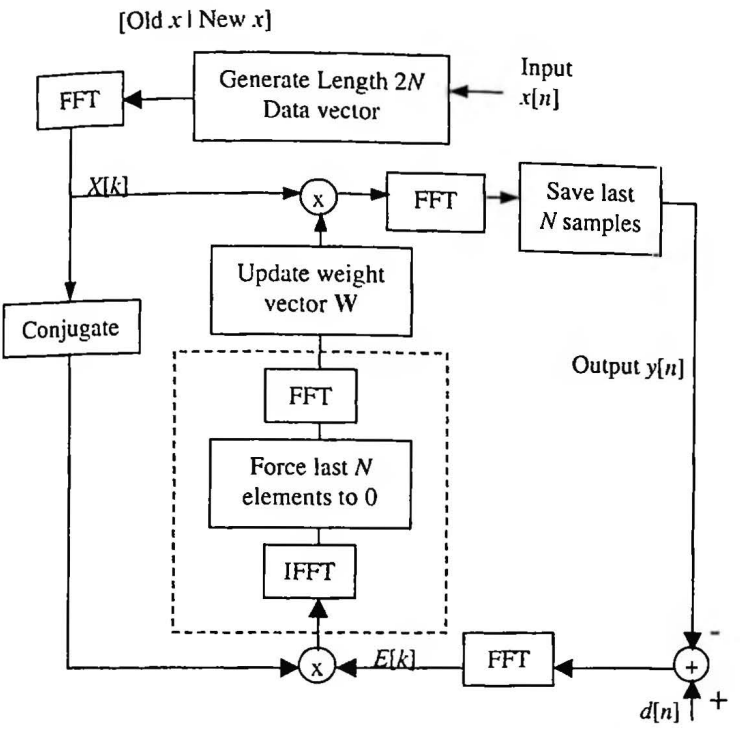


Figure 10.19 Block diagram of the constrained frequency-domain block LMS algorithm. The unconstrained version of this algorithm eliminates the computation inside the dashed box.

An unconstrained frequency-domain LMS algorithm can be implemented by removing the constraint in Figure 10.19, therefore implementing a circular instead of a linear convolution. While this is not exact, the algorithm requires only three FFTs instead of five. In some practical applications, there is no difference in convergence between the constrained and unconstrained cases.

10.3.5. The RLS Algorithm

The search for the optimum filter can be accelerated when the gradient vector is properly deviated toward the minimum. This approach uses the Newton-Raphson method to iteratively compute the root of $f(x)$ (see Figure 10.20) so that the value at iteration $i + 1$ is given by

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{10.55}$$

Amazon/VB Assets
Exhibit 1012
Page 529

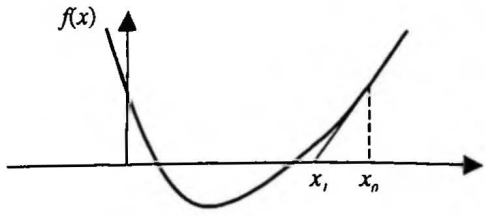


Figure 10.20 Newton-Raphson method to compute the roots of a function.

To minimize function $f(x)$ we thus compute the roots of $f'(x)$ through the above method:

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)} \tag{10.56}$$

In the case of a vector, Eq. (10.56) is transformed into

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \varepsilon[n]\left(\nabla^2 e(\mathbf{w}_i)\right)^{-1} \nabla e(\mathbf{w}_i) \tag{10.57}$$

where we add a step size $\varepsilon[n]$, and where $\nabla^2 e(\mathbf{w}_i)$ is the *Hessian* of the least-squares function which, for Eq. (10.37), equals the autocorrelation of $\mathbf{x}$:

$$\nabla^2 e(\mathbf{w}_i) = \mathbf{R}[n] = E\{\mathbf{x}[n]\mathbf{x}^T[n]\} \tag{10.58}$$

The *recursive least squares* (RLS) algorithm specifies a method of estimating Eq. (10.58) using an exponential window:

$$\mathbf{R}[n] = \lambda \mathbf{R}[n-1] + \mathbf{x}[n]\mathbf{x}^T[n] \tag{10.59}$$

While the RLS algorithm converges faster than the LMS algorithm, it also is more computationally expensive, as it requires a matrix inversion for every sample. Several algorithms have been derived to speed it up [54].

10.4. MULTIMICROPHONE SPEECH ENHANCEMENT

The use of more than one microphone is motivated by the human auditory system, in which the use of both ears has been shown to enhance detection of the direction of arrival, as well as increase SNR when one ear is covered. The methods the human auditory system uses to accomplish this task are still not completely known, and the techniques described in this section do not mimic that behavior.

Microphone arrays use multiple microphones and knowledge of the microphone locations to predict delays and thus create a beam that focuses on the direction of the desired

Amazon/VB Assets
Exhibit 1012
Page 530

speaker and rejects signals coming from other angles. Reverberation, as discussed in Section 10.1.2, can be combated with these techniques. Blind source separation techniques are another family of statistical techniques that typically do not use spatial constraints, but rather statistical independence between different sources.

While in this section we describe only linear processing, i.e., the output speech is a linearly filtered version of the microphone signals, we could also combine these techniques with the nonlinear methods of Section 10.5.

10.4.1. Microphone Arrays

The goals of microphone arrays are twofold: finding the position of a sound source in a room, and improving the SNR of the received signal. *Steering* is helpful in videoconferencing, where a camera has to follow the current speaker. Since the speaker is typically far away from the microphone, the received signal likely contains a fair amount of additive noise. Microphone arrays can also be used to increase the SNR.

Let $x[n]$ be the signal at the source S. Microphone i picks up a signal

$$y_i[n] = x[n] * g_i[n] + v_i[n] \tag{10.60}$$

that is a filtered version of the source plus additive noise $v_i[n]$. If we have N such microphones, we can attempt to recover $s[n]$ because all the signals $y_i[n]$ should be correlated.

A typical assumption made is that all the filters $g_i[n]$ are delayed versions of the same filter $g[n]$

$$g_i[n] = g[n - D_i] \tag{10.61}$$

with the delay $D_i = d_i / c$, d_i being the distance between the source S and microphone i, and c the speed of sound in air. We cannot recover signal $x[n]$ without knowledge of $g[n]$ or the signal itself, so the goal is to obtain the filtered signal $y[n]$

$$y[n] = x[n] * g[n] \tag{10.62}$$

so that, combining Eqs. (10.60), (10.61), and (10.62),

$$y_i[n] = y[n - D_i] + v_i[n] \tag{10.63}$$

Assuming $v_i[n]$ are independent and Gaussianly distributed, the optimal estimate of $x[n]$ is given by

$$\tilde{y}[n] = \frac{1}{N} \sum_{i=0}^{N-1} y_i[n + D_i] = y[n] + v[n] \tag{10.64}$$

which is the so-called *delay-and-sum beamformer* [24, 29], where the residual noise $v[n]$

Amazon/VB Assets
Exhibit 1012
Page 531

$$v[n] = \frac{1}{N}\sum_{i=0}^{N-1} v_i[n + D_i] \tag{10.65}$$

has a variance that decreases as the number of microphones N increases, since the noises $v_i[n + D_i]$ are uncorrelated.

Equation (10.64) requires estimation of the delays D_i. To attenuate the additive noise $v[n]$, it is not necessary to identify the absolute delays, but rather the delays relative to one reference microphone (for example, the center microphone). It can be shown that the maximum likelihood solution consists in maximizing the energy of $\tilde{y}[n]$ in Eq. (10.64), which is the sum of cross-correlations:

$$D_i = \arg\max_{D_i}\left(\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} R_{ij}[D_i - D_j]\right) \qquad 0 \le i < N \tag{10.66}$$

This approach assumes that we know nothing about the geometry of the microphone placement. In fact, given a point source and assuming no reflections, we can compute the delay based on the distance between the source and the microphone. The use of geometry allows us to reduce the number of parameters to estimate from $(N - 1)$ to a maximum of 3, in case we desire to estimate the exact location. This location is often described in spherical coordinates (φ, θ, ρ) with φ being the direction of arrival, θ the elevation angle, and ρ the distance to the reference microphone, as shown in Figure 10.21.

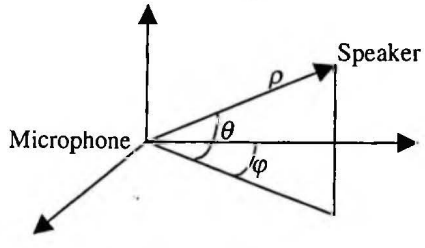


Figure 10.21 Spherical coordinates (φ, θ, ρ) with φ being the direction of arrival, θ the elevation angle, and ρ the distance to the reference microphone.

While 2-D and 3-D microphone configurations can be used, which would allow us to determine not just the steering angle φ, but also distance to the origin ρ and azimuth θ, linear microphone arrays are the most widely used configurations because they are the simplest. In a linear array all the microphones are placed on a line (see Figure 10.22). In this case, we cannot determine the elevation angle θ. To determine both φ and ρ we need at least two microphones in the array.

If the microphones are relatively close to each other compared to the distance to the source, the angle of arrival φ is approximately the same for all signals. With this assumption, the normalized delay $\bar{D}_i$ with respect to the reference microphone is given by

Amazon/VB Assets
Exhibit 1012
Page 532

$$\bar{D}_i = -a_i \sin(\varphi)/c \tag{10.67}$$

where a_i is the y-axis coordinate in Figure 10.22 for microphone i, where the reference microphone has $a_0 = 0$ and also $\bar{D}_0 = 0$.

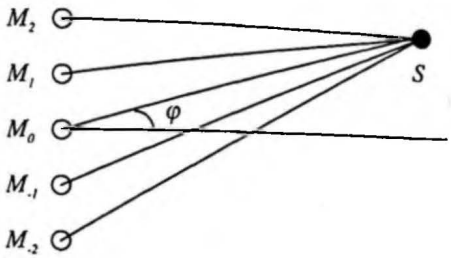


Figure 10.22 Linear microphone array (five microphones). The source signal arrives at each microphone with a different delay, which allows us to find the correct angle of arrival.

With approximation, we define $\bar{D}_i(\varphi)$, the relative delay of the signal at microphone i to the reference microphone, as a function of the direction of arrival angle φ and independent of ρ. The optimal direction of arrival φ is then that which maximizes the energy of the estimated signal $\bar{x}[n]$ over a set of samples

$$\begin{aligned}\varphi &= \arg\max_{\varphi} \sum_n \left(\frac{1}{N} \sum_{i=0}^{N-1} y_i[n + \bar{D}_i(\varphi)] \right)^2 \\ &= \arg\max_{\varphi} \sum_n \left(\frac{1}{N} \sum_{i=0}^{N-1} y_i[n - \frac{a_i}{c} \sin(\varphi)] \right)^2\end{aligned} \tag{10.68}$$

The term *beamforming* entails that this array favors a specific direction of arrival φ and that sources arriving from other directions are not in phase and therefore are attenuated. Since the source can move over time, maximization of Eq. (10.68) can be done in an adaptive fashion.

As the beam is steered away from the broadside, the system exhibits a reduction in spatial discrimination because the beam pattern broadens. Furthermore, beamwidth varies with frequency, so an array has an approximate bandwidth given by the upper f_u and lower f_l frequencies

$$\begin{aligned} f_u &= \frac{c}{d \max_{\varphi,\varphi'} |\cos\varphi - \cos\varphi'|} \\ f_l &= \frac{f_u}{N} \end{aligned} \tag{10.69}$$

Amazon/VB Assets
Exhibit 1012
Page 533

with d being the sensor spacing, φ' the steering angle measured with respect to the axis of the array, and φ the direction of the source. For a desired range of $\pm 30^\circ$ and five sensors spaced 5 cm apart, the range is approximately 880 to 4400 Hz. We see in Figure 10.23 that at very low frequencies the response is essentially omnidirectional, since the microphone spacing is small compared to the large wavelength. At high frequencies more lobes start appearing, and the array steers toward not only the preferred direction but others as well. For speech signals, the upshot is that we either need a lot of microphones to provide a directional polar pattern at low frequencies, or we need them to be spread far enough apart, or both.

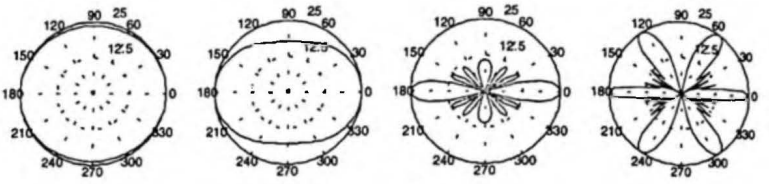


Figure 10.23 Polar pattern of a microphone array with steering angle of $\varphi' = 0$, five microphones spaced 5 cm apart for 400, 880, 4400, and 8000 Hz from left to right, respectively, for a source located at 5 m.

The polar pattern in Figure 10.23 was computed as follows:

$$P(f,r,\varphi) = \sum_{i=1}^{N} \frac{e^{-j2\pi f\left[a_i \sin\varphi' + |re^{j\varphi} - ja_i|\right]/c}}{\left|re^{j\varphi} - ja_i\right|} \tag{10.70}$$

though the sensors could be spaced nonuniformly, as in Figure 10.24, allowing for better behavior across the frequency spectrum.

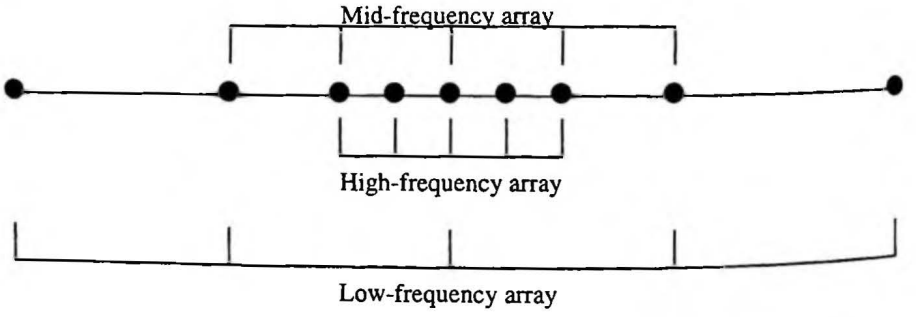


Figure 10.24 Nonuniform linear microphone array containing three subarrays for the high, mid, and low frequencies.

Amazon/VB Assets
Exhibit 1012
Page 534

Once a microphone array has been steered towars a direction φ', it attenuates noise source coming from other directions. The beamwidth depends not only on the frequency of the signal, but also on the steering direction. If the beam is steered toward a direction φ', then the direction of the source for which the beam response fall to half its power has been found empirically to be

$$\varphi_{3dB}(f) = \cos^{-1}\left\{\cos\varphi' \pm \frac{K}{Ndf}\right\} \tag{10.71}$$

with K being a constant. Equation (10.71) shows that the smaller the array, the wider the beam, and that lower frequencies yield wider beams also. Figure 10.25 shows that the bandwidth of the array when steering toward a 30° direction is lower than when steering at 0°.

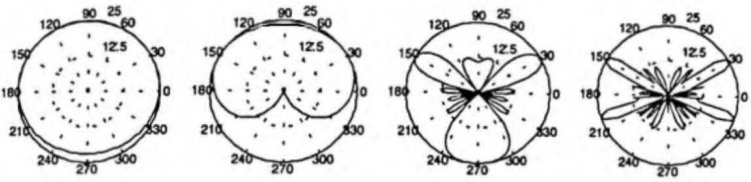


Figure 10.25 Polar pattern of a microphone array with steering angle of $\varphi' = 30°$, five microphones spaced 5 cm apart for 400, 880, 3000, and 4400 Hz from left to right, respectively, for a source located at 5 m.

Microphone arrays have been shown to improve recognition accuracy when the microphones and the speaker are far apart [51]. Several companies are commercializing microphone arrays for teleconferencing or speech recognition applications.

Only in anechoic chambers does the assumption in Eq. (10.61) hold, since in practice many reflections take place, which are also different for different microphones. In addition, the assumption of a common direction of arrival for all microphones may not hold either. For this case of reverberant environments, single beamformers typically fail. While computing the direction of arrival is much more difficult in this case, the SNR can still be improved.

Let's define the desired signal $d[n]$ as that obtained in the reference microphone. We can estimate the vector $\mathbf{H}[n] = \{h_{11}, \cdots, h_{1L}, h_{21}, \cdots, h_{2L}, \cdots, h_{(N-1)1}, \cdots, h_{(N-1)L}\}$ for the $(N\text{-}1)$ L-tap filters that minimizes the error array [25]

$$e[n] = d[n] - \mathbf{H}[n]\mathbf{Y}[n] \tag{10.72}$$

where the $(N-1)$ microphone signals are represented in the vector

$$\mathbf{Y}[n] = \{y_1[n], \cdots, y_1[n-L-1], y_2[n], \cdots, y_2[n-L-1], \cdots, y_{N-1}[n], \cdots, y_{N-1}[n-L-1]\}$$

The filter coefficients $\mathbf{G}[n]$ can be estimated through the adaptive filtering techniques described in Section 10.3. The clean signal is then estimated as

Amazon/VB Assets
Exhibit 1012
Page 535

$$\hat{x}[n] = \frac{1}{2}\left(d[n] + \mathbf{H}[n]\mathbf{Y}[n]\right) \tag{10.73}$$

This last method does not assume anything about the geometry of the microphone array.

10.4.2. Blind Source Separation

The problem of separating the desired speech from interfering sources, the *cocktail party effect* [15], has been one of the holy grails in signal processing. *Blind source separation* (BSS) is a set of techniques that assume no information about the mixing process or the sources, apart from their mutual statistical independence, hence is termed blind. *Independent component analysis* (ICA), developed in the last few years [19, 38], is a set of techniques to solve the BSS problem that estimate a set of linear filters to separate the mixed signals under the assumption that the original sources are statistically independent.

Let's first consider *instantaneous mixing*. Let's assume that R microphone signals $y_i[n]$, denoted by $\mathbf{y}[n] = \left(y_1[n], y_2[n], \cdots, y_R[n]\right)$, are obtained by a linear combination of R unobserved source signals $x_i[n]$, denoted by $\mathbf{x}[n] = \left(x_1[n], x_2[n], \cdots, x_R[n]\right)$:

$$\mathbf{y}[n] = \mathbf{G}\mathbf{x}[n] \tag{10.74}$$

for all n, with $\mathbf{G}$ being the $R \times R$ mixing matrix. This mixing is termed instantaneous, since the sensor signals at time n depend on the sources at the same, but no earlier, time point. Had the mixing matrix been given, its inverse could have been applied to the sensor signals to recover the sources by $\mathbf{x}[n] = \mathbf{G}^{-1}\mathbf{y}[n]$. In the absence of any information about the mixing, the blind separation problem consists of estimating a separating matrix $\mathbf{H} = \mathbf{G}^{-1}$ from the observed microphone signals alone. The source signals can then be recovered by

$$\mathbf{x}[n] = \mathbf{H}\mathbf{y}[n] \tag{10.75}$$

We'll use here the probabilistic formulation of ICA, though alternate frameworks for ICA have been derived also [18]. Let $p_{\mathbf{x}}(\mathbf{x}[n])$ be the probability density function (pdf) of the source signals, so that the pdf of microphone signals $\mathbf{y}[n]$ is given by

$$p_{\mathbf{y}}(\mathbf{y}[n]) = \mid \mathbf{H} \mid p_{\mathbf{x}}(\mathbf{H}\mathbf{y}[n]) \tag{10.76}$$

and if we furthermore assume the sources $\mathbf{x}[n]$ are independent from themselves in time, $\mathbf{x}[n+i] \;\; i \neq 0$, then the joint probability is given by

$$p_{\mathbf{y}}(\mathbf{y}[0], \mathbf{y}[1], \cdots, \mathbf{y}[N-1]) = \prod_{n=0}^{N-1} p_{\mathbf{y}}(\mathbf{y}[n]) = \mid \mathbf{H} \mid^{N} \prod_{n=0}^{N-1} p_{\mathbf{x}}(\mathbf{H}\mathbf{y}[n]) \tag{10.77}$$

whose normalized log-likelihood is given by

Amazon/VB Assets
Exhibit 1012
Page 536

$$\Psi = \frac{1}{N} \ln p_{\mathbf{y}}(\mathbf{y}[0], \mathbf{y}[1], \cdots, \mathbf{y}[N-1]) = \ln |\mathbf{H}| + \frac{1}{N} \sum_{n=0}^{N-1} \ln p_{\mathbf{x}}(\mathbf{H}\mathbf{y}[n]) \tag{10.78}$$

It can be shown that

$$\frac{\partial \ln |\mathbf{H}|}{\partial \mathbf{H}} = \left(\mathbf{H}^T\right)^{-1} \tag{10.79}$$

so that that the gradient of Ψ [38] in Eq. (10.78) is given by

$$\frac{\partial \Psi}{\partial \mathbf{H}} = \left(\mathbf{H}^T\right)^{-1} + \frac{1}{N} \sum_{n=0}^{N-1} \phi(\mathbf{H}\mathbf{y}[n]) \left(\mathbf{y}[n]\right)^T \tag{10.80}$$

where $\phi(\mathbf{x})$ is expressed as

$$\phi(\mathbf{x}) = \frac{\partial \ln p_{\mathbf{x}}(\mathbf{x})}{\partial \mathbf{x}} \tag{10.81}$$

If we further assume the distribution is a zero mean Gaussian distribution with standard deviation σ, then Eq. (10.81) results in

$$\phi(\mathbf{x}) = -\frac{\mathbf{x}}{\sigma^2} \tag{10.82}$$

which inserted into Eq. (10.80) yields

$$\frac{\partial \Psi}{\partial \mathbf{H}} = \left(\mathbf{H}^T\right)^{-1} - \frac{\mathbf{H}}{\sigma^2} \left(\frac{1}{N} \sum_{n=0}^{N-1} \mathbf{y}[n] \left(\mathbf{y}[n]\right)^T \right) = \left(\mathbf{H}^T\right)^{-1} - \frac{\mathbf{H}}{\sigma^2} \mathbf{R} \tag{10.83}$$

with $\mathbf{R}$ being the matrix of cross-correlations, i.e.,

$$R_{ij} = \frac{1}{N} \sum_{n=0}^{N-1} y_i[n] y_j[n] \tag{10.84}$$

Setting Eq. (10.83) to $\mathbf{0}$ results in maximization of Eq. (10.78) under the Gaussian assumption:

$$\mathbf{H}^T \mathbf{H} = \sigma^2 \mathbf{R}^{-1} \tag{10.85}$$

which can be solved using the Cholesky decomposition described in Chapter 6.

Since σ is generally not known, it can be shown from Eq. (10.85) that the sources can be recovered only to within a scaling factor [17]. Scaling is in general not a big problem, since speech recognition systems perform automatic gain control (AGC). Moreover, the sources can be recovered to within a permutation. To see this, let's define a two-dimensional matrix $\mathbf{A}$

Amazon/VB Assets
Exhibit 1012
Page 537

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{10.86}$$

which is orthogonal:

$$\mathbf{A}^T\mathbf{A} = \mathbf{I} \tag{10.87}$$

If $\mathbf{H}$ is a solution of Eq. (10.85), then $\mathbf{AH}$ is also a solution. Thus, a permutation of the sources yields the same correlation matrix in Eq. (10.84). Although we have shown it only under the Gaussian assumption, separation up to a scaling factor and source permutation is a general result in blind source separation [17].

Unfortunately, the Gaussian assumption does not guarantee separation. To see this, we can define a two-dimensional rotation matrix $\mathbf{A}$

$$\mathbf{A} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{10.88}$$

which is also orthogonal, so that if $\mathbf{H}$ is a solution of Eq. (10.85), then $\mathbf{AH}$ is also a solution.

The Gaussian assumption entails considering only second-order statistics, and to ensure separation we could consider higher-order statistics. Since speech signals do not follow a Gaussian distribution, we could use a Laplacian distribution, as we saw in Chapter 7:

$$p_x(x) = \frac{\beta}{2} e^{-\beta|x|} \tag{10.89}$$

which, using Eq. (10.81), results in

$$\phi(x) = \begin{cases} -\beta & x > 0 \\ \beta & x < 0 \end{cases} \tag{10.90}$$

and thus a nonlinear function of $\mathbf{H}$ for Eq. (10.80). Since a closed-form solution is not possible, a common solution in this case is gradient descent, where the gradient is given by

$$\frac{\partial \Psi}{\partial \mathbf{H}_n} = \left(\mathbf{H}_n^T\right)^{-1} + \phi(\mathbf{H}_n\mathbf{y}[n])(\mathbf{y}[n])^T \tag{10.91}$$

and the update formula by

$$\mathbf{H}_{n+1} = \mathbf{H}_n - \varepsilon \frac{\partial \Psi}{\partial \mathbf{H}_n} = \mathbf{H}_n - \varepsilon\left[\left(\mathbf{H}_n^T\right)^{-1} + \phi(\mathbf{H}_n\mathbf{y}[n])(\mathbf{y}[n])^T\right] \tag{10.92}$$

which is the so-called *infomax* rule [10].

Amazon/VB Assets
Exhibit 1012
Page 538

Often the nonlinearity in Eq. (10.90) is replaced by a sigmoid[6] function:

$$\phi(x) = -\beta \tanh(\beta x) \tag{10.93}$$

which implies a density function

$$p_x(x) = \frac{\beta}{2\pi \cosh(\beta x)} \tag{10.94}$$

The sigmoid converges to the Laplacian as $\beta \to \infty$. Nonlinear functions in Eqs. (10.90) and (10.93) can be expanded in Taylor series so that all the moments of the observed signals are used and not just the second order, as in the case of the Gaussian assumption. These nonlinearities have been shown to be more effective in separating the sources. The use of more accurate density functions for $p_x(\mathbf{x})$, such as a mixture of Gaussians [9], also results in nonlinear $\phi(\mathbf{x})$ functions that have shown better separation.

A problem of Eq. (10.92) is that it requires a matrix inversion at every iteration. The so-called *natural gradient* [7] was suggested to avoid this, also providing faster convergence. To do this we can multiply the gradient of Eq. (10.91) by a positive definite matrix, the inverse of the Fisher's information matrix $\mathbf{H}_n^T\mathbf{H}_n$, for example, to whiten the signal:

$$\mathbf{H}_{n+1} = \mathbf{H}_n - \varepsilon \frac{\partial \Psi}{\partial \mathbf{H}} \mathbf{H}_n^T \mathbf{H}_n \tag{10.95}$$

which, combined with Eq. (10.91), results in

$$\mathbf{H}_{n+1} = \mathbf{H}_n - \varepsilon \left[\mathbf{I} + \phi(\hat{\mathbf{x}}[n])(\hat{\mathbf{x}}[n])^T\right]\mathbf{H}_n \tag{10.96}$$

where the estimated sources are given by

$$\hat{\mathbf{x}}[n] = \mathbf{H}_n \mathbf{y}[n] \tag{10.97}$$

Notice the similarity of this approach to the RLS algorithm of Section 10.3.5. Similarly to most Newton-Raphson methods, the convergence of this approach is quadratic instead of linear as long as we are close enough to the maximum.

Another way of overcoming the lack of separation under the independent Gaussian assumption is to make use of temporal information, which we know is important for speech signals. If the model of Eq. (10.74) is extended to contain additive noise

$$\mathbf{y}[n] = \mathbf{G}\mathbf{x}[n] + \mathbf{v}[n] \tag{10.98}$$

[6] The sigmoid function can be expressed in terms of the hyperbolic tangent $\tanh(x) = \sinh(x)/\cosh(x)$, where $\sinh(x) = (e^x - e^{-x})/2$ and $\cosh(x) = (e^x + e^{-x})/2$.

Amazon/VB Assets
Exhibit 1012
Page 539

we can compute the autocorrelation of $\mathbf{y}[n]$ as

$$\mathbf{R}_y[n] = \mathbf{G}\mathbf{R}_x[n]\mathbf{G}^T + \mathbf{R}_v[n] \tag{10.99}$$

or, after some manipulation,

$$\mathbf{R}_x[n] = \mathbf{H}(\mathbf{R}_y[n] - \mathbf{R}_v[n])\mathbf{H}^T \tag{10.100}$$

which we know must be diagonal because the sources $\mathbf{x}$ are independent, and thus $\mathbf{H}$ can be estimated to minimize the squared error of the off-diagonal terms of $\mathbf{R}_x[n]$ for several values of n [11]. Equation (10.100) is a generalization of Eq. (10.85) when considering temporal correlation and additive noise.

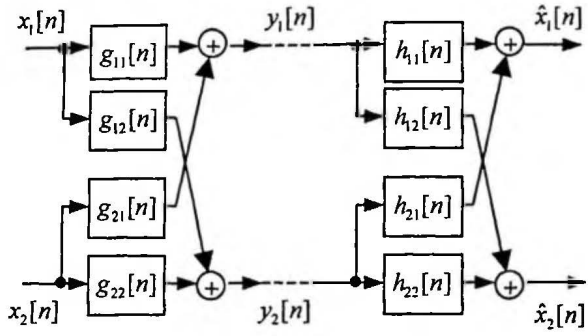


Figure 10.26 Convolutional model for the case of two microphones.

The case of instantaneous mixing is not realistic, as we need to consider the transfer functions between the sources and the microphones created by the room acoustics. It can be shown that the reconstruction filters $h_{ij}[n]$ in Figure 10.26 will completely recover the original signals $x_i[n]$ *if and only if* their z-transforms are the inverse of the z-transforms of the mixing filters $g_{ij}[n]$:

$$\begin{aligned}\begin{pmatrix} H_{11}(z) & H_{12}(z) \\ H_{21}(z) & H_{22}(z) \end{pmatrix} &= \begin{pmatrix} G_{11}(z) & G_{12}(z) \\ G_{21}(z) & G_{22}(z) \end{pmatrix}^{-1} \\ &= \frac{1}{G_{11}(z)G_{22}(z) - G_{12}(z)G_{21}(z)} \begin{pmatrix} G_{11}(z) & G_{12}(z) \\ G_{21}(z) & G_{22}(z) \end{pmatrix}\end{aligned} \tag{10.101}$$

If the matrix in Eq. (10.101) is not invertible, separability is impossible. This can happen if both microphones pick up the same signal, which could happen if either the two mi-

Amazon/VB Assets
Exhibit 1012
Page 540

crophones are too close to each other or the two sources are too close to each other. It's reasonable to assume the mixing filters $g_{ij}[n]$ to be FIR filters, whose length will generally depend on the reverberation time, which in turn depends on the room size, microphone position, wall absorbance, and so on. In general this means that the reconstruction filters $h_{ij}[n]$ have an infinite impulse response. In addition, the filters $g_{ij}[n]$ may have zeroes outside the unit circle, so that perfect reconstruction filters would need to have poles outside the unit circle. For this reason it is not possible, in general, to recover the original signals exactly.

In practice, it's convenient to assume such filters to be FIR of length q, which means that the original signals $x_1[n]$ and $x_2[n]$, will not be recovered exactly. Thus the problem consists in estimating the reconstruction filters $h_{ij}[n]$ directly from the microphone signals $y_1[n]$ and $y_2[n]$, so that the estimated signals $\hat{x}_i[n]$ are as close as possible to the original signals. Often we are satisfied if the resulting signals are separated, even if they contain some amount of reverberation.

An approach commonly used to combat this problem consists of taking a filterbank and assuming instantaneous mixing within each filter [38]. This approach can separate real sources much more effectively, but it suffers from the problem of permutations, which in this case is more severe because frequencies from different sources can be mixed together. To avoid this, we may need a probabilistic model of the sources that takes into account correlations across frequencies [3]. Another problem occurs when the number of sources is larger than the number of microphones.

10.5. ENVIRONMENT COMPENSATION PREPROCESSING

The goal of this section is to present a number of techniques used to *clean up* the signal of additive noise and/or channel distortions prior to the speech recognition system. Although the techniques presented here are developed for the case of one microphone, they can be generalized to the case where several microphones are available using the approaches described in Section 10.4. These techniques can also be used to *enhance* the signal captured with a speakerphone or a desktop microphone in teleconferencing applications.

Since the use of human auditory system is so robust to changes in acoustical environment, many researchers have attempted to develop signal processing schemes that mimic the functional organization of the peripheral auditory system [27, 49]. The PLP cepstrum described in Chapter 6 has also been shown to be very effective in combating noise and channel distortions [60].

Another alternative is to consider the feature vector as an integral part of the recognizer, and thus researchers have investigated its design so as to maximize recognition accuracy, as discussed in Chapter 9. Such approaches include LDA [34] and neural networks [45]. These discriminatively trained features can also be optimized to operate better under noisy conditions, thus possibly beating the standard mel-cepstrum, especially when several independent features are combined [50]. The mel-cepstrum is the most popular feature vector for speech recognition. In this context we present a number of techniques that have been proposed over the years to compensate for the effects of additive noise and channel distortions on the cepstrum.

Amazon/VB Assets
Exhibit 1012
Page 541

10.5.1. Spectral Subtraction

The basic assumption in this section is that the desired clean signal $x[m]$ has been corrupted by additive noise $n[m]$:

$$y[m] = x[m] + n[m] \tag{10.102}$$

and that both $x[m]$ and $n[m]$ are statistically independent, so that the power spectrum of the output $y[m]$ can be approximated as the sum of the power spectra:

$$|Y(f)|^2 \approx |X(f)|^2 + |N(f)|^2 \tag{10.103}$$

with equality if we take expected values, as the expected value of the cross term vanishes (see Section 10.1.3).

Although we don't know $|N(f)|^2$, we can obtain an estimate using the average periodogram over M frames that are known to be just noise (i.e., when no signal is present) as long as the noise is stationary

$$|\hat{N}(f)|^2 = \frac{1}{M}\sum_{i=0}^{M-1}|Y_i(f)|^2 \tag{10.104}$$

Spectral subtraction supplies an intuitive estimate for $|X(f)|$ using Eqs. (10.103) and (10.104) as

$$|\hat{X}(f)|^2 = |Y(f)|^2 - |\hat{N}(f)|^2 = |Y(f)|^2\left(1 - \frac{1}{SNR(f)}\right) \tag{10.105}$$

where we have defined the frequency-dependent signal-to-noise ratio $SNR(f)$ as

$$SNR(f) = \frac{|Y(f)|^2}{|\hat{N}(f)|^2} \tag{10.106}$$

Equation (10.105) describes the magnitude of the Fourier transform but not the phase. This is not a problem if we are interested in computing the mel-cepstrum as discussed in Chapter 6. We can just modify the magnitude and keep the original phase of $Y(f)$ using a filter $H_{ss}(f)$:

$$\hat{X}(f) = Y(f)H_{ss}(f) \tag{10.107}$$

where, according to Eq. (10.105), $H_{ss}(f)$ is given by

$$H_{ss}(f) = \sqrt{1 - \frac{1}{SNR(f)}} \tag{10.108}$$

Amazon/VB Assets
Exhibit 1012
Page 542

Since $\left|\hat{X}(f)\right|^2$ is a power spectral density, it has to be positive, and therefore

$$SNR(f) \geq 1 \tag{10.109}$$

but we have no guarantee that $SNR(f)$, as computed by Eq. (10.106), satisfies Eq. (10.109). In fact, it is easy to see that noise frames do not comply. To enforce this constraint, Boll [13] suggested modifying Eq. (10.108) as follows:

$$H_{ss}(f) = \sqrt{\max\left(1 - \frac{1}{SNR(f)}, a\right)} \tag{10.110}$$

with $a \geq 0$, so that the quantity within the square root is always positive, and where $f_{ss}(x)$ is given by

$$f_{ss}(x) = \sqrt{\max\left(1 - \frac{1}{x}, a\right)} \tag{10.111}$$

It is useful to express $SNR(f)$ in dB so that

$$\overline{x} = 10 \log_{10} SNR \tag{10.112}$$

and the gain of the filter in Eq. (10.111) also in dB:

$$g_{ss}(\overline{x}) = 20 \log_{10} f_{ss}(\overline{x}) \tag{10.113}$$

Using Eqs. (10.111) and (10.112), we can express Eq. (10.113) by

$$g_{ss}(\overline{x}) = \max\left(10 \log_{10}\left(1 - 10^{-\overline{x}/10}\right), -A\right) \tag{10.114}$$

after expressing the attenuation a in dB:

$$a = 10^{-A/10} \tag{10.115}$$

Equation (10.114) is plotted in Figure 10.27 for A = 10 dB.

The spectral subtraction rule in Eq. (10.111) is quite intuitive. To implement it we can do a short-time analysis, as shown in Chapter 6, by using overlapping windowed segments, zero-padding, computing the FFT, modifying the magnitude spectrum, taking the inverse FFT, and adding the resulting windows.

This implementation results in output speech that has significantly less noise, though it exhibits what is called *musical noise* [12]. This is caused by frequency bands f for which $|Y(f)|^2 \approx |\hat{N}(f)|^2$. As shown in Figure 10.27, a frequency f_0 for which $|Y(f_0)|^2 < |\hat{N}(f_0)|^2$ is attenuated by A dB, whereas a neighboring frequency f_1, where $|Y(f_1)|^2 > |\hat{N}(f_1)|^2$, has a much smaller attenuation. These rapid changes with frequency introduce tones at varying frequencies that appear and disappear rapidly.

Amazon/VB Assets
Exhibit 1012
Page 543

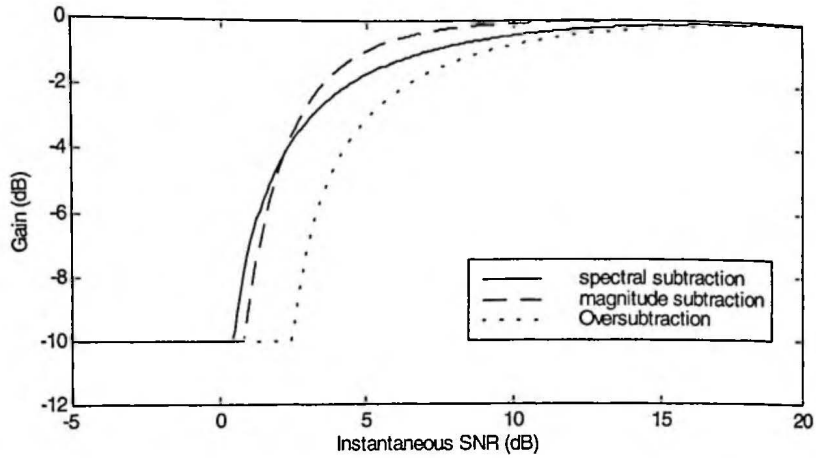


Figure 10.27 Magnitude of the spectral subtraction filter gain as a function of the input instantaneous SNR for A = 10 dB, for the spectral subtraction of Eq. (10.114), magnitude subtraction of Eq. (10.118), and oversubtraction of Eq. (10.119) with β = 2 dB.

The main reason for the presence of musical noise is that the estimates of $SNR(f)$ through Eqs. (10.104) and (10.106) are poor. This is partly because $SNR(f)$ is computed independently for each frequency, whereas we know that $SNR(f_0)$ and $SNR(f_1)$ are correlated if f_0 and f_1 are close to each other. Thus, one possibility is to smooth the filter in Eq. (10.114) over frequency. This approach suppresses a smaller amount of noise, but it does not distort the signal as much, and thus may be preferred by listeners. Similarly, smoothing over time

$$SNR(f,t) = \gamma SNR(f,t-1) + (1-\gamma)\frac{|Y(f)|^2}{|\hat{N}(f)|^2} \tag{10.116}$$

can also be done to reduce the distortion, at the expense of a smaller noise attenuation. Smoothing over both time and frequency can be done to obtain more accurate SNR measurements and thus less distortion. As shown in Figure 10.28, use of spectral subtraction can reduce the error rate.

Additionally, the attenuation A can be made a function of frequency. This is useful when we want to suppress more noise at one frequency than another, which is a tradeoff between noise reduction and nonlinear distortion of speech.

Other enhancements to the basic algorithm have been proposed to reduce the musical noise. Sometimes Eq. (10.111) is generalized to

$$f_{ms}(x) = \left(\max\left(1 - \frac{1}{x^{\alpha/2}}, a \right) \right)^{1/\alpha} \tag{10.117}$$

Amazon/VB Assets
Exhibit 1012
Page 544

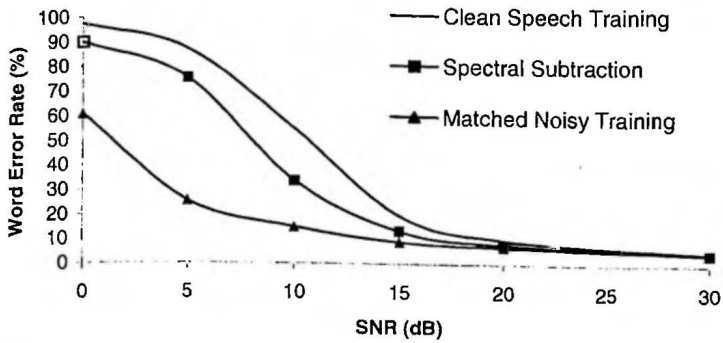


Figure 10.28 Word error rate as a function of SNR (dB) using Whisper on the *Wall Street Journal* 5000-word dictation task. White noise was added at different SNRs. The solid line represents the baseline system trained with clean speech, the line with squares the use of spectral subtraction with the previous clean HMMs. They are compared to a system trained on the same speech with the same SNR as the speech tested on.

where $\alpha = 2$ corresponds to the *power spectral subtraction* rule in Eq. (10.111), and $\alpha = 1$ corresponds to the *magnitude subtraction* rule (plotted in Figure 10.27 for A = 10 dB):

$$g_{ms}(\overline{x}) = \max\left(20\log_{10}\left(1-10^{-\overline{x}/5}\right), -A\right) \tag{10.118}$$

Another variation, called oversubtraction, consists of multiplying the estimate of the noise power spectral density $|\hat{N}(f)|^2$ in Eq. (10.104) by a constant $10^{\beta/10}$, where $\beta > 0$, which causes the power spectral subtraction rule in Eq. (10.114) to be transformed to another function

$$g_{ns}(\overline{x}) = \max\left(10\log_{10}\left(1-10^{-(\overline{x}-\beta)/10}\right), -A\right) \tag{10.119}$$

This causes $|Y(f)|^2 < |\hat{N}(f)|^2$ to occur more often than $|Y(f)|^2 > |\hat{N}(f)|^2$ for frames for which $|Y(f)|^2 \approx |\hat{N}(f)|^2$, and thus reduces the musical noise.

10.5.2. Frequency-Domain MMSE from Stereo Data

You have seen that several possible functions, such as Eqs. (10.114), (10.118), or (10.119), can be used to attenuate the noise, and it is not clear that any one of them is better than the others, since each has been obtained through different assumptions. This opens the possibility of estimating the curve $g(\overline{x})$ using a different criterion, and, thus, different approximations than those used in Section 10.5.1.

One interesting possibility occurs when we have pairs of stereo utterances that have been recorded simultaneously in noise-free conditions in one channel and noisy conditions

Amazon/VB Assets
Exhibit 1012
Page 545

in the other channel. In this case, we can estimate $f(x)$ using a minimum mean squared criterion (Porter and Boll [47], Ephraim and Malah [23]), so that

$$\hat{f}(x) = \arg\min_{f(x)} \left\{ \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(X_i(f_j) - f\left(SNR(f_j)\right) Y_i(f_j) \right)^2 \right\} \tag{10.120}$$

or $g(x)$ as

$$\hat{g}(x) = \arg\min_{g(x)} \left\{ \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left(10\log_{10} \left|X_i(f_j)\right|^2 - g\left(SNR(f_j)\right) - 10\log_{10} \left|Y_i(f_j)\right|^2 \right)^2 \right\} \tag{10.121}$$

which can be solved by discretizing $f(x)$ and $g(x)$ into several bins and summing over all M frequencies and N frames. This approach results in a curve that is smoother and thus offers less musical noise and lower distortion. Stereo utterances of noise-free and noisy speech are needed to estimate $f(x)$ and $g(x)$ through Eqs. (10.120) and (10.121) for any given acoustical environment and can be collected with two microphones, or the noisy speech can be obtained by adding to the clean speech artificial noise from the testing environment.

Another generalization of this approach is to use a different function $f(x)$ or $g(x)$ for every frequency [2] as shown in Figure 10.29. This also allows for a lower squared error at the expense of having to store more data tables. In the experiments of Figure 10.29, we note that more subtraction is needed at lower frequencies than at higher frequencies in this case.

If such stereo data is available to estimate these curves, it makes the enhanced speech sound better [23] than does spectral subtraction. When used in speech recognition systems, it also leads to higher accuracies [2].

10.5.3. Wiener Filtering

Let's reformulate Eq. (10.102) from the statistical point of view. The process $\mathbf{y}[n]$ is the sum of random process $\mathbf{x}[n]$ and the additive noise $\mathbf{v}[n]$ process:

$$\mathbf{y}[n] = \mathbf{x}[n] + \mathbf{v}[n] \tag{10.122}$$

We wish to find a linear estimate $\hat{\mathbf{x}}[n]$ in terms of the process $\mathbf{y}[n]$:

$$\hat{\mathbf{x}}[n] = \sum_{m=-\infty}^{\infty} h[m]\mathbf{y}[n-m] \tag{10.123}$$

which is the result of a linear time-invariant filtering operation. The MMSE estimate of $h[n]$ in Eq. (10.123) minimizes the squared error

$$E\left\{ \left[\mathbf{x}[n] - \sum_{m=-\infty}^{\infty} h[m]\mathbf{y}[n-m] \right]^2 \right\} \tag{10.124}$$

Amazon/VB Assets
Exhibit 1012
Page 546

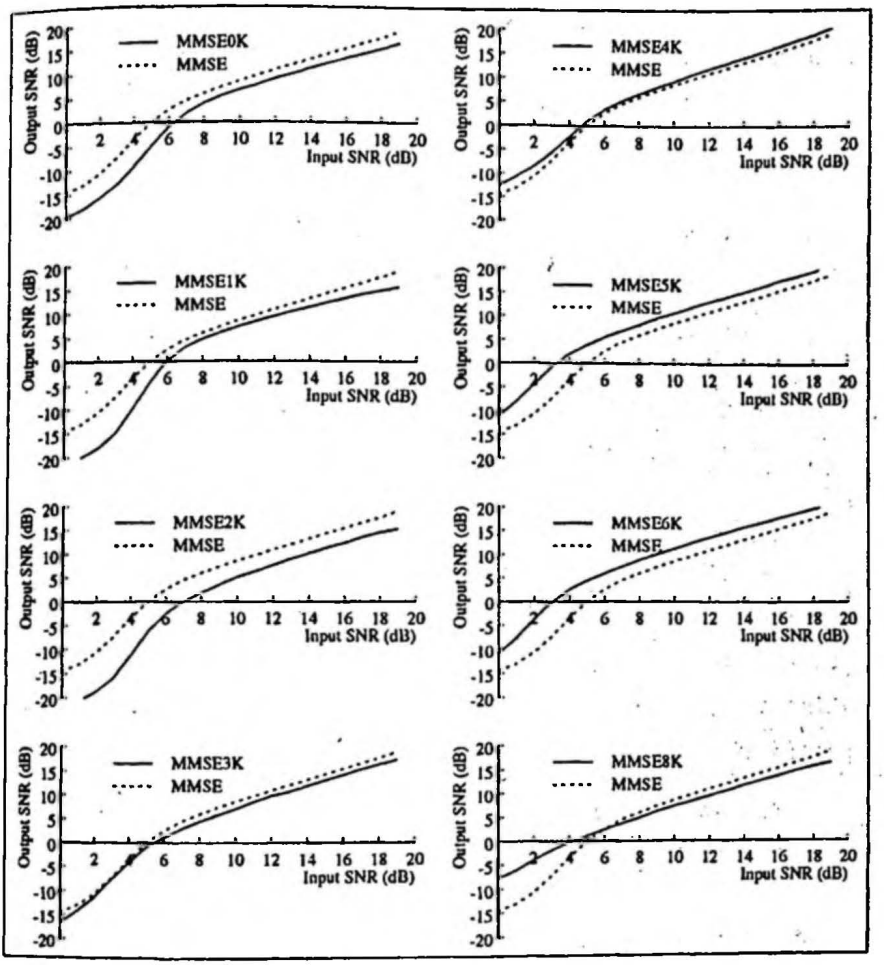


Figure 10.29 Empirical curves for input-to-output instantaneous SNR. Eight different curves for 0, 1, 2, 3, 4, 5, 6, 7 and 8 kHz are obtained following Eq. (10.121) [2] using speech recorded simultaneously from a close-talking microphone and a desktop microphone.

which results in the famous *Wiener-Hopf* equation

$$R_{xy}[l] = \sum_{m=-\infty}^{\infty} h[m] R_{yy}[l-m] \tag{10.125}$$

Amazon/VB Assets
Exhibit 1012
Page 547

so that, taking Fourier transforms, the resulting filter can be expressed in the frequency domain as

$$H(f) = \frac{S_{xy}(f)}{S_{yy}(f)} \tag{10.126}$$

If the signal $\mathbf{x}[n]$ and the noise $\mathbf{v}[n]$ are orthogonal, which is often the case, then

$$S_{xy}(f) = S_{xx}(f) \text{ and } S_{yy}(f) = S_{xx}(f) + S_{vv}(f) \tag{10.127}$$

so that Eq. (10.126) is given by

$$H(f) = \frac{S_{xx}(f)}{S_{xx}(f) + S_{vv}(f)} \tag{10.128}$$

Equation (10.128) is called the *noncausal Wiener filter*. This can be realized only if we know the power spectra of both the noise and the signal. Of course, if $S_{xx}(f)$ and $S_{vv}(f)$ do not overlap, then $H(f) = 1$ in the band of the signal and $H(f) = 0$ in the band of the noise.

In practice, $S_{xx}(f)$ is unknown. If it were known, we could compute its mel-cepstrum, which would coincide exactly with the mel-cepstrum before noise addition. To solve this chicken-and-egg problem, we need some kind of model. Ephraim [22] proposed the use of an HMM where, if we know what state the current frame falls under, we can use its mean spectrum as $S_{xx}(f)$. In practice we do not know what state each frame falls into either, so he proposed to weigh the filters for each state by the a posterior probability that the frame falls into each state. This algorithm, when used in speech enhancement, results in gains of 7 dB or more.

A causal version of the Wiener filter can also be derived. A dynamical state model algorithm called the Kalman filter (see [42] for details) is also an extension of the Wiener filter.

10.5.4. Cepstral Mean Normalization (CMN)

Different microphones have different transfer functions, and even the same microphone has a varying transfer function depending on the distance to the microphone and the room acoustics. In this section we describe a powerful and simple technique that is designed to handle convolutional distortions and, thus, increases the robustness of speech recognition systems to unknown linear filtering operations.

Given a signal $x[n]$, we compute its cepstrum through short-time analysis, resulting in a set of T cepstral vectors $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{T-1}\}$. Its sample mean $\bar{\mathbf{x}}$ is given by

$$\bar{\mathbf{x}} = \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{x}_t \tag{10.129}$$

Amazon/VB Assets
Exhibit 1012
Page 548

Cepstral mean normalization (CMN) (Atal [8]) consists of subtracting $\bar{\mathbf{x}}$ from each vector $\mathbf{x}_t$ to obtain the normalized cepstrum vector $\hat{\mathbf{x}}_t$:

$$\hat{\mathbf{x}}_t = \mathbf{x}_t - \bar{\mathbf{x}} \tag{10.130}$$

Let's now consider a signal $y[n]$, which is the output of passing $x[n]$ through a filter $h[n]$. We can compute another sequence of cepstrum vectors $\mathbf{Y} = \{\mathbf{y}_0, \mathbf{y}_1, \cdots, \mathbf{y}_{T-1}\}$. Now let's further define a vector $\mathbf{h}$ as

$$\mathbf{h} = \mathbf{C}\left(\ln\left|H(\omega_0)\right|^2 \quad \cdots \quad \ln\left|H(\omega_{M'})\right|^2\right) \tag{10.131}$$

where $\mathbf{C}$ is the DCT matrix. We can see that

$$\mathbf{y}_t = \mathbf{x}_t + \mathbf{h} \tag{10.132}$$

and thus the sample mean $\bar{\mathbf{y}}_t$ equals

$$\bar{\mathbf{y}} = \frac{1}{T}\sum_{t=0}^{T-1}\mathbf{y}_t = \frac{1}{T}\sum_{t=0}^{T-1}(\mathbf{x}_t + \mathbf{h}) = \bar{\mathbf{x}} + \mathbf{h} \tag{10.133}$$

and its normalized cepstrum is given by

$$\hat{\mathbf{y}}_t = \mathbf{y}_t - \bar{\mathbf{y}}_t = \hat{\mathbf{x}}_t \tag{10.134}$$

which indicates that cepstral mean normalization is immune to linear filtering operations. This procedure is performed on every utterance for both training and testing. Intuitively, the mean vector $\bar{\mathbf{x}}$ conveys the spectral characteristics of the current microphone and room acoustics. In the limit, when $T \to \infty$ for each utterance, we should expect means from utterances from the same recording environment to be the same. Use of CMN to the cepstrum vectors does not modify the delta or delta-delta cepstrum.

Let's analyze the effect of CMN on a short utterance. Assume that our utterance contains a single phoneme, say /s/. The mean $\bar{\mathbf{x}}$ will be very similar to the frames in this phoneme, since /s/ is quite stationary. Thus, after normalization, $\hat{\mathbf{x}}_t \approx 0$. A similar result will happen for other fricatives, which means that it would be impossible to distinguish these ultrashort utterances, and the error rate will be very high. If the utterance contains more than one phoneme but is still short, this problem is not insurmountable, but the confusion among phonemes is still higher than if no CMN had been applied. Empirically, it has been found that this procedure does not degrade the recognition rate on utterances from the same acoustical environment, as long as they are longer than 2–4 seconds. Yet the method provides significant robustness against linear filtering operations. In fact, for telephone recordings, where each call has a different frequency response, the use of CMN has been shown to provide as much as 30% relative decrease in error rate. When a system is trained on one microphone and tested on another, CMN can provide significant robustness.

Interestingly enough, it has been found in practice that the error rate for utterances within the same environment is actually somewhat lower, too. This is surprising, given that

Amazon/VB Assets
Exhibit 1012
Page 549

there is no mismatch in channel conditions. One explanation is that, even for the same microphone and room acoustics, the distance between the mouth and the microphone varies for different speakers, which causes slightly different transfer functions, as we studied in Section 10.2. In addition, the cepstral mean characterizes not only the channel transfer function, but also the average frequency response of different speakers. By removing the long-term speaker average, CMN can act as sort of speaker normalization.

One drawback of CMN is it does not discriminate silence and voice in computing the utterance mean. An extension to CMN consists in computing different means for noise and speech [5]:

$$\begin{aligned} \mathbf{h}^{(j+1)} &= \frac{1}{N_s}\sum_{t\in q_s}\mathbf{x}_t - \mathbf{m}_s \\ \mathbf{n}^{(j+1)} &= \frac{1}{N_n}\sum_{t\in q_n}\mathbf{x}_t - \mathbf{m}_n \end{aligned} \tag{10.135}$$

i.e., the difference between the average vector for speech frames in the utterance and the average vector $\mathbf{m}_s$ for speech frames in the training data, and similarly for the noise frames $\mathbf{m}_n$. Speech/noise discrimination could be done by classifying frames into speech frames and noise frames, computing the average cepstra for each, and subtracting them from the average in the training data. This procedure works well as long as the speech/noise classification is accurate. It's best done by the recognizer, since other speech detection algorithms can fail in high background noise (see Section 10.6.2). To avoid errors in transitions between speech and noise, delta and delta-delta can be computed prior to this speech/noise mean normalization so that they are unaffected. As shown in Figure 10.30, this algorithm has been shown to improve robustness not only to varying channels but also to noise.

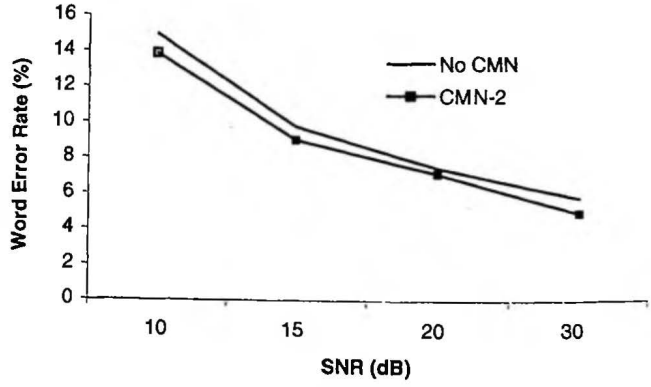


Figure 10.30 Word error rate as a function of SNR (dB) for both no CMN and CMN-2 [5]. White noise was added at different SNRs and the system was trained with speech with the same SNR. The Whisper system is used on the 5000-word *Wall Street Journal* task using a bigram language model.

Amazon/VB Assets
Exhibit 1012
Page 550

10.5.5. Real-Time Cepstral Normalization

CMN requires the complete utterance to compute the cepstral mean; thus, it cannot be used in a real-time system, and an approximation needs to be used. In this section we discuss a modified version of CMN that can address this problem, as well as a set of techniques called RASTA that attempt to do the same thing.

We can interpret CMN as the operation of subtracting a low-pass filter $d[n]$, where all the T coefficients are identical and equal $1/T$, which is a high-pass filter with a cutoff frequency ω_c that is arbitrarily close to 0. This interpretation indicates that we can implement other types of high-pass filters. One that has been found to work well in practice is the exponential filter, so the cepstral mean $\bar{\mathbf{x}}_t$ is a function of time

$$\bar{\mathbf{x}}_t = \alpha \mathbf{x}_t + (1-\alpha)\bar{\mathbf{x}}_{t-1} \tag{10.136}$$

where α is chosen so that the filter has a time constant[7] of at least 5 seconds of speech.

Other types of filters have been proposed in the literature. In fact, a popular approach consists of an IIR bandpass filter with the transfer function:

$$H(z) = 0.1z^4 * \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}} \tag{10.137}$$

which is used in the so-called relative spectral processing or RASTA [32]. As in CMN, the high-pass portion of the filter is expected to alleviate the effect of convolutional noise introduced in the channel. The low-pass filtering helps to smooth some of the fast frame-to-frame spectral changes present. Empirically, it has been shown that the RASTA filter behaves similarly to the real-time implementation of CMN, albeit with a slightly higher error rate. Both the RASTA filter and real-time implementations of CMN require the filter to be properly initialized. Otherwise, the first utterance may use an incorrect cepstral mean. The original derivation of RASTA includes a few stages prior to the bandpass filter, and this filter is performed on the spectral energies, not the cepstrum.

10.5.6. The Use of Gaussian Mixture Models

Algorithms such as spectral subtraction of Section 10.5.1 or the frequency-domain MMSE of Section 10.5.2 implicitly assume that different frequencies are uncorrelated from each other. Because of that, the spectrum of the enhanced signal may exhibit abrupt changes across frequency and not look like spectra of real speech signals. Using the model of the

[7] The time constant τ of a low-pass filter is defined as the value for which the output is cut in half. For an exponential filter of parameter α and sampling rate F_s, $\alpha = \ln 2/(TF_s)$.

Amazon/VB Assets
Exhibit 1012
Page 551

environment of Section 10.1.3, we can express the clean-speech cepstral vector $\mathbf{x}$ as a function of the observed noisy cepstral vector $\mathbf{y}$ as

$$\mathbf{x} = \mathbf{y} - \mathbf{h} - \mathbf{C}\ln\left(1 - e^{\mathbf{C}^{-1}(\mathbf{n}-\mathbf{y})}\right) \tag{10.138}$$

where the noise cepstral vector $\mathbf{n}$ is a random vector. The MMSE estimate of $\mathbf{x}$ is given by

$$\hat{\mathbf{x}}_{MMSE} = E\{\mathbf{x} \mid \mathbf{y}\} = \mathbf{y} - \mathbf{h} - \mathbf{C}E\left\{\ln\left(1 - e^{\mathbf{C}^{-1}(\mathbf{n}-\mathbf{y})}\right) \mid \mathbf{y}\right\} \tag{10.139}$$

where the expectation uses the distribution of $\mathbf{n}$. Solution to Eq. (10.139) results in a nonlinear function which can be learned, for example, with a neural network [53].

A popular model to attack this problem consists in modeling the probability distribution of the noisy speech $\mathbf{y}$ as a mixture of K Gaussians:

$$p(\mathbf{y}) = \sum_{k=0}^{K-1} p(\mathbf{y} \mid k)P[k] = \sum_{k=0}^{K-1} \mathrm{N}(\mathbf{y}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)P[k] \tag{10.140}$$

where $P[k]$ is the prior probability of each Gaussian component k. If $\mathbf{x}$ and $\mathbf{y}$ are jointly Gaussian within class k, then $p(\mathbf{x} \mid \mathbf{y}, k)$ is also Gaussian [42] with mean:

$$E\{\mathbf{x} \mid \mathbf{y}, k\} = \boldsymbol{\mu}_{\mathbf{x}}^k + \boldsymbol{\Sigma}_{\mathbf{xy}}^k \left(\boldsymbol{\Sigma}_{\mathbf{y}}^k\right)^{-1} (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}}^k) = \mathbf{C}_k\mathbf{y} + \mathbf{r}_k \tag{10.141}$$

so that the joint distribution of $\mathbf{x}$ and $\mathbf{y}$ is given by

$$\begin{aligned} p(\mathbf{x},\mathbf{y}) &= \sum_{k=0}^{K-1} p(\mathbf{x},\mathbf{y} \mid k)P[k] = \sum_{k=0}^{K-1} p(\mathbf{x} \mid \mathbf{y}, k)p(\mathbf{y} \mid k)P[k] \\ &= \sum_{k=0}^{K-1} \mathrm{N}(\mathbf{x}, \mathbf{C}_k\mathbf{y} + \mathbf{r}_k, \boldsymbol{\Gamma}_k)\mathrm{N}(\mathbf{y}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)P[k] \end{aligned} \tag{10.142}$$

where $\mathbf{r}_k$ is called the *correction vector*, $\mathbf{C}_k$ is the *rotation matrix*, and the matrix $\boldsymbol{\Gamma}_k$ tells us how uncertain we are about the compensation.

A maximum likelihood estimate of $\mathbf{x}$ maximizes the joint probability in Eq. (10.142). Assuming the Gaussians do not overlap very much (as in the FCDCN algorithm [2]):

$$\hat{\mathbf{x}}_{ML} \approx \arg\max_k p(\mathbf{x},\mathbf{y},k) = \arg\max_k \mathrm{N}(\mathbf{y}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\mathrm{N}(\mathbf{x}, \mathbf{C}_k\mathbf{y} + \mathbf{r}_k, \boldsymbol{\Gamma}_k)P[k] \tag{10.143}$$

Amazon/VB Assets
Exhibit 1012
Page 552

whose solution is

$$\hat{\mathbf{x}}_{ML} = \mathbf{C}_{\hat{k}}\mathbf{y} + \mathbf{r}_{\hat{k}} \tag{10.144}$$

where

$$\hat{k} = \arg\max_{k} \mathrm{N}(\mathbf{y}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) P[k] \tag{10.145}$$

It is often more robust to compute the MMSE estimate of $\mathbf{x}$ (as in the CDCN [2] and RATZ [43] algorithms):

$$\hat{\mathbf{x}}_{MMSE} = E\{\mathbf{x} \mid \mathbf{y}\} = \sum_{k=0}^{K-1} p(k \mid \mathbf{y}) E\{\mathbf{x} \mid \mathbf{y}, k\} = \sum_{k=0}^{K-1} p(k \mid \mathbf{y})(\mathbf{C}_k \mathbf{y} + \mathbf{r}_k) \tag{10.146}$$

as a weighted sum for all mixture components, where the posterior probability $p(k \mid \mathbf{y})$ is given by

$$p(k \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid k) P[k]}{\sum_{k=0}^{K-1} p(\mathbf{y} \mid k) P[k]} \tag{10.147}$$

where the rotation matrix $\mathbf{C}_k$ in Eq. (10.144) can be replaced by $\mathbf{I}$ with a modest degradation in performance in return for faster computation [21].

A number of different algorithms [2, 43] have been proposed that vary in how the parameters $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, $\mathbf{r}_k$, and $\boldsymbol{\Gamma}_k$ are estimated. If stereo recordings are available from both the clean signal and the noisy signal, then we can estimate $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$ by fitting a mixture Gaussian model to $\mathbf{y}$ as described in Chapter 3. Then $\mathbf{C}_k$, $\mathbf{r}_k$ and $\boldsymbol{\Gamma}_k$ can be estimated directly by linear regression of $\mathbf{x}$ and $\mathbf{y}$. The FCDCN algorithm [2, 6] is a variant of this approach when it is assumed that $\boldsymbol{\Sigma}_k = \sigma^2 \mathbf{I}$, $\boldsymbol{\Gamma}_k = \gamma^2 \mathbf{I}$, and $\mathbf{C}_k = \mathbf{I}$, so that $\boldsymbol{\mu}_k$ and $\mathbf{r}_k$ are estimated through a VQ procedure and $\mathbf{r}_k$ is the average difference $(\mathbf{y} - \mathbf{x})$ for vectors $\mathbf{y}$ that belong to mixture component k. An enhancement is to use the instantaneous SNR of a frame, defined as the difference between the log-energy of that frame and the average log-energy of the background noise. It is advantageous to use different correction vectors for different instantaneous SNR levels. The log-energy can be replaced by the zeroth-order cepstral coefficient with little change in recognition accuracy.

Amazon/VB Assets
Exhibit 1012
Page 553

Often, stereo recordings are not available and we need other means of estimating parameters μ_k, Σ_k, $\mathbf{r}_k$, and Γ_k. CDCN [6] is one such algorithm that has a model of the environment as described in Section 10.1.3, which defines a nonlinear relationship between $\mathbf{x}$, $\mathbf{y}$ and the environmental parameters $\mathbf{n}$ and $\mathbf{h}$ for the noise and channel. This method also uses an MMSE approach where the correction vector is a weighted average of the correction vectors for all classes. An extension of CDCN using a vector Taylor series approximation [44] for that nonlinear function has been shown to offer improved results. Other methods that do not require stereo recordings or a model of the environment are presented in [43].

10.6. ENVIRONMENTAL MODEL ADAPTATION

We describe a number of techniques that achieve compensation by adapting the HMM to the noisy conditions. The most straightforward method is to retrain the whole HMM with the speech from the new acoustical environment. Another option is to apply standard adaptive techniques discussed in Chapter 9 to the case of environment adaptation. We consider a model of the environment that allows constrained adaptation methods for more efficient adaptation in comparison to the general techniques.

10.6.1. Retraining on Corrupted Speech

If there is a mismatch between acoustical environments, it is sensible to retrain the HMM. This is done in practice for telephone speech where only telephone speech, and no clean high-bandwidth speech, is used in the training phase.

Unfortunately, training a large-vocabulary speech recognizer requires a very large amount of data, which is often not available for a specific noisy condition. For example, it is difficult to collect a large amount of training data in a car driving at 50 mph, whereas it is much easier to record it at idle speed. Having a small amount of matched-conditions training data could be worse than a large amount of mismatched-conditions training data. Often we want to adapt our model given a relatively small sample of speech from the new acoustical environment.

One option is to take a noise waveform from the new environment, add it to all the utterances in our database, and retrain the system. If the noise characteristics are known ahead of time, this method allows us to adapt the model to the new environment with a relatively small amount of data from the new environment, yet use a large amount of training data. Figure 10.31 shows the benefit of this approach over a system trained on clean speech for the case of additive white noise.. If the target acoustical environment also has a different channel, we can also filter all the utterances in the training data prior to retraining. This method allows us to adapt the model to the new environment with a relatively small amount of data from the new environment.

If the noise sample is available offline, this simple technique can provide good results at no cost during recognition. Otherwise the noise addition and model retraining would need

Amazon/VB Assets
Exhibit 1012
Page 554

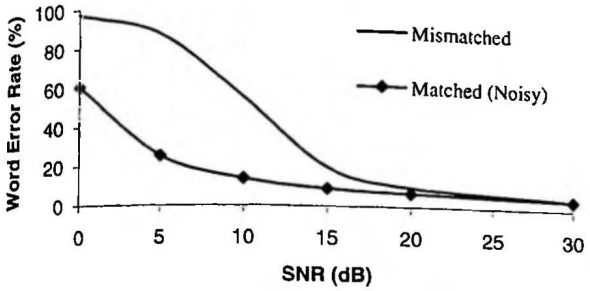


Figure 10.31 Word error rate as a function of the testing data SNR (dB) for Whisper trained on clean data and a system trained on noisy data at the same SNR as the testing set as in Figure 10.30. White noise at different SNRs is added.

to occur at runtime. This is feasible for speaker-dependent small-vocabulary systems where the training data can be kept in memory and where the retraining time can be small, but it is generally not feasible for large-vocabulary speaker-independent systems because of memory and computational limitations.

One possibility is to create a number of artificial acoustical environments by corrupting our clean database with noise samples of varying levels and types, as well as varying channels. Then all those waveforms from multiple acoustical environments can be used in training. This is called *multistyle training* [39], since our training data comes from different conditions. Because of the diversity of the training data, the resulting recognizer is more robust to varying noise conditions. In Figure 10.32 we see that, though generally the error-rate curve is above the matched-condition curve, particularly for clean speech, multistyle training does not require knowledge of the specific noise level and thus is a viable alternative to the theoretical lower bound of matched conditions.

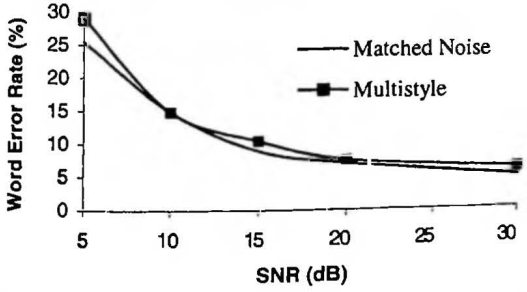


Figure 10.32 Word error rates of multistyle training compared to matched-noise training as a function of the SNR in dB for additive white noise. Whisper is trained as in Figure 10.30. The error rate of multistyle training is between 12% (for low SNR) and 25% (for high SNR) higher in relative terms than that of matched-condition training. Nonetheless, multistyle training does better than a system trained on clean data for all conditions other than clean speech.

Amazon/VB Assets
Exhibit 1012
Page 555

10.6.2. Model Adaptation

We can also use the standard adaptation methods used for speaker adaptation, such as MAP or MLLR described in Chapter 9. Since MAP is an unstructured method, it can offer results similar to those of matched conditions, but it requires a significant amount of adaptation data. MLLR can achieve reasonable performance with about a minute of speech for minor mismatches [41]. For severe mismatches, MLLR also requires a large number of transformations, which, in turn, require a larger amount of adaptation data as discussed in Chapter 9.

Let's analyze the case of a single MLLR transform, where the affine transformation is simply a bias. In this case the MLLR transform consists only of a vector $\mathbf{h}$ that, as in the case of CMN described in Section 10.5.4, can be estimated from a single utterance. Instead of estimating $\mathbf{h}$ as the average cepstral mean, this method estimates $\mathbf{h}$ as the maximum likelihood estimate, given a set of sample vectors $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{T-1}\}$ and an HMM model λ [48], and it is a version of the EM algorithm where all the vector means are tied together (see Algorithm 10.2). This procedure for estimating the cepstral bias has a very slight reduction in error rates over CMN, although the improvement is larger for short utterances [48].

ALGORITHM 10.2: *MLE SIGNAL BIAS REMOVAL*

Step 1: Initialize $\mathbf{h}^{(0)} = \mathbf{0}$ at iteration $j = 0$
Step 2: Obtain model $\lambda^{(j)}$ by updating the means from $\mathbf{m}_k$ to $\mathbf{m}_k + \mathbf{h}^{(j)}$, for all Gaussians k.
Step 3: Run recognition with model $\lambda^{(j)}$ on the current utterance and determine a state segmentation $\theta[t]$ for each frame t.
Step 4: Estimate $\mathbf{h}^{(j+1)}$ as the vector that maximizes the likelihood, which, using covariance matrices $\boldsymbol{\Sigma}_k$, is given by:

$$\mathbf{h}^{(j+1)} = \left(\sum_{t=0}^{T-1} \boldsymbol{\Sigma}_{\theta[t]}^{-1} \right)^{-1} \sum_{t=0}^{T-1} \boldsymbol{\Sigma}_{\theta[t]}^{-1} \left(\mathbf{x}_t - \mathbf{m}_{\theta[t]} \right) \tag{10.148}$$

Step 5: If converged, stop; otherwise, increment j and go to Step 2. In practice two iterations are often sufficient.

If both additive noise and linear filtering are applied, the cepstrum for the noise and that for most speech frames are affected differently. The *speech/noise mean normalization* [5] algorithm can be extended similarly, as shown in Algorithm 10.3. The idea is to estimate a vector $\bar{\mathbf{n}}$ and $\bar{\mathbf{h}}$, such that all the Gaussians associated to the noise model are shifted by $\bar{\mathbf{n}}$, and all remaining Gaussians are shifted by $\bar{\mathbf{h}}$.

We can make Eq. (10.150) more efficient by tying all the covariance matrices. This transforms Eq. (10.150) into

$$\begin{aligned} \mathbf{h}^{(j+1)} &= \frac{1}{N_s} \sum_{t \in q_s} \mathbf{x}_t - \mathbf{m}_s \\ \mathbf{n}^{(j+1)} &= \frac{1}{N_n} \sum_{t \in q_n} \mathbf{x}_t - \mathbf{m}_n \end{aligned} \tag{10.149}$$

Amazon/VB Assets
Exhibit 1012
Page 556

i.e., the difference between the average vector for speech frames in the utterance and the average vector $\mathbf{m}_s$ for speech frames in the training data, and similarly for the noise frames $\mathbf{m}_n$. This is essentially the same equation as in the speech-noise cepstral mean normalization described in Section 10.5.4. The difference is that the speech/noise discrimination is done by the recognizer instead of by a separate classifier. This method is more accurate in high-background-noise conditions where traditional speech/noise classifiers can fail. As a compromise, a codebook with considerably fewer Gaussians than a recognizer can be used to estimate $\bar{\mathbf{n}}$ and $\bar{\mathbf{h}}$.

ALGORITHM 10.3: *SPEECH/NOISE MEAN NORMALIZATION*

Step 1: Initialize $\mathbf{h}^{(0)} = \mathbf{0}$, $\mathbf{n}^{(0)} = \mathbf{0}$ at iteration $j = 0$

Step 2: Obtain model $\lambda^{(j)}$ by updating the means of speech Gaussians from $\mathbf{m}_k$ to $\mathbf{m}_k + \mathbf{h}^{(j)}$, and of noise Gaussians from $\mathbf{m}_l$ to $\mathbf{m}_l + \mathbf{n}^{(j)}$.

Step 3: Run recognition with model $\lambda^{(j)}$ on the current utterance and determine a state segmentation $\theta[t]$ for each frame t.

Step 4: Estimate $\mathbf{h}^{(j+1)}$ and $\mathbf{n}^{(j+1)}$ as the vectors that maximize the likelihood for speech frames ($t \in q_s$) and noise frames ($t \in q_n$), respectively:

$$\mathbf{h}^{(j+1)} = \left(\sum_{t \in q_s} \Sigma_{\theta[t]}^{-1} \right)^{-1} \sum_{t \in q_s} \Sigma_{\theta[t]}^{-1} \left(\mathbf{x}_t - \mathbf{m}_{\theta[t]} \right)$$

$$\mathbf{n}^{(j+1)} = \left(\sum_{t \in q_n} \Sigma_{\theta[t]}^{-1} \right)^{-1} \sum_{t \in q_n} \Sigma_{\theta[t]}^{-1} \left(\mathbf{x}_t - \mathbf{m}_{\theta[t]} \right) \qquad (10.150)$$

Step 5: If converged, stop; otherwise, increment j and go to Step 2.

10.6.3. Parallel Model Combination

By using the clean-speech models and a noise model, we can approximate the distributions obtained by training a HMM with corrupted speech. The memory requirements for the algorithm are then significantly reduced, as the training data is not needed online. *Parallel model combination* (PMC) is a method to obtain the distribution of noisy speech given the distribution of clean speech and noise as mixture of Gaussians. As discussed in Section 10.1.3, if the clean-speech cepstrum follows a Gaussian distribution and the noise cepstrum follows another Gaussian distribution, the noisy speech has a distribution that is no longer Gaussian. The PMC method nevertheless makes the assumption that the resulting distribution is Gaussian whose mean and covariance matrix are the mean and covariance matrix of the resulting non-Gaussian distribution. If it is assumed that the distribution of clean speech is a mixture of N Gaussians, and the distribution of the noise is a mixture of M Gaussians, the distribution of the noisy speech contains NM Gaussians. The feature vector is often composed of the cepstrum, delta cepstrum, and delta-delta cepstrum. The model combination can be seen in Figure 10.33.

Amazon/VB Assets
Exhibit 1012
Page 557

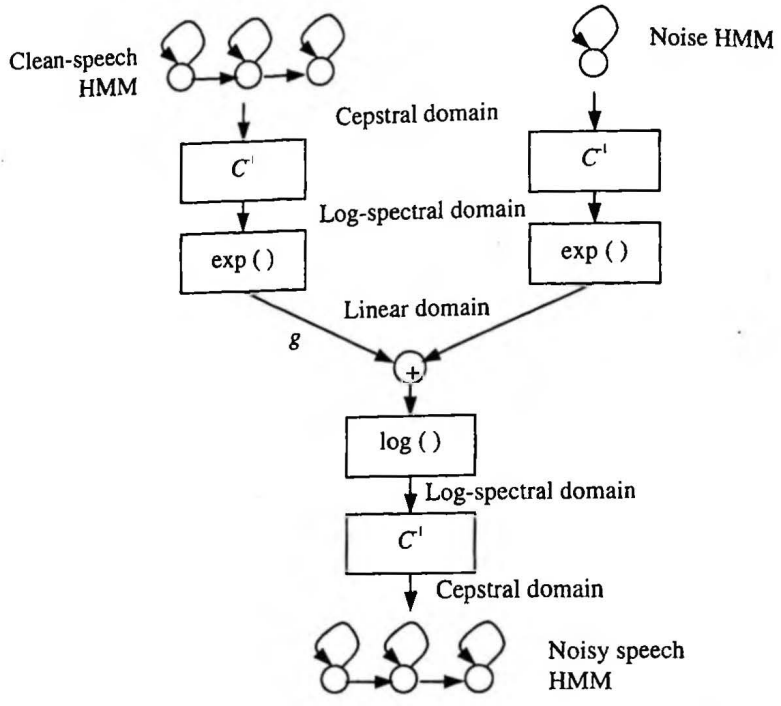


Figure 10.33 Parallel model combination for the case of one-state noise HMM.

If the mean and covariance matrix of the cepstral noise vector $\mathbf{n}$ are given by $\boldsymbol{\mu}_n^c$ and $\boldsymbol{\Sigma}_n^c$, respectively, we first compute the mean and covariance matrix in the log-spectral domain:

$$\begin{aligned}\boldsymbol{\mu}_n^l &= \mathbf{C}^{-1}\boldsymbol{\mu}_n^c \\ \boldsymbol{\Sigma}_n^l &= \mathbf{C}^{-1}\boldsymbol{\Sigma}_n^c(\mathbf{C}^{-1})^T\end{aligned} \tag{10.151}$$

In the linear domain $\mathbf{N} = e^{\mathbf{n}}$, the distribution is lognormal, whose mean vector $\boldsymbol{\mu}_N$ and covariance matrix $\boldsymbol{\Sigma}_N$ can be shown (see Chapter 3) to be given by

$$\begin{aligned}\boldsymbol{\mu}_N[i] &= \exp\left\{\boldsymbol{\mu}_n^l[i] + \boldsymbol{\Sigma}_n^l[i,i]/2\right\} \\ \boldsymbol{\Sigma}_N[i,j] &= \boldsymbol{\mu}_N[i]\boldsymbol{\mu}_N[j]\left(\exp\left\{\boldsymbol{\Sigma}_n^l[i,j]\right\} - 1\right)\end{aligned} \tag{10.152}$$

with expressions similar to Eqs. (10.151) and (10.152) for the mean and covariance matrix of $\mathbf{X}$.

Amazon/VB Assets
Exhibit 1012
Page 558

Using the model of the environment with no filter is equivalent to obtaining a random linear spectral vector $\mathbf{Y}$ given by (see Figure 10.33)

$$\mathbf{Y} = \mathbf{X} + \mathbf{N} \tag{10.153}$$

and, since $\mathbf{X}$ and $\mathbf{N}$ are independent, we can obtain the mean and covariance matrix of $\mathbf{Y}$ as

$$\begin{aligned} \mu_Y &= \mu_X + \mu_N \\ \Sigma_Y &= \Sigma_X + \Sigma_N \end{aligned} \tag{10.154}$$

Although the sum of two lognormal distributions is not lognormal, the popular *log-normal approximation* [26] consists in assuming that $\mathbf{Y}$ is lognormal. In this case we can apply the inverse formulae of Eq. (10.152) to obtain the mean and covariance matrix in the log-spectral domain:

$$\begin{aligned} \Sigma_y^l[i,j] &= \ln\left\{\frac{\Sigma_Y[i,j]}{\mu_Y[i]\mu_Y[j]} + 1\right\} \\ \mu_y^l[i] &= \ln\mu_Y[i] - \frac{1}{2}\ln\left\{\frac{\Sigma_Y[i,j]}{\mu_Y[i]\mu_Y[j]} + 1\right\} \end{aligned} \tag{10.155}$$

and finally return to the cepstrum domain applying the inverse of Eq. (10.151):

$$\begin{aligned} \mu_y^c &= \mathbf{C}\mu_y^l \\ \Sigma_y^c &= \mathbf{C}\Sigma_y^l\mathbf{C}^T \end{aligned} \tag{10.156}$$

The lognormal approximation cannot be used directly for the delta and delta-delta cepstrum. Another variant that can be used in this case and is more accurate than the lognormal approximation is the *data-driven parallel model combination* (DPMC) [26], which uses Monte Carlo simulation to draw random cepstrum vectors from both the clean-speech HMM and noise distribution to create cepstrum of the noisy speech by applying Eqs. (10.20) and (10.21) to each sample point. These composite cepstrum vectors are not kept in memory, only their means and covariance matrices are, therefore reducing the required memory though still requiring a significant amount of computation. The number of vectors drawn from the distribution was at least 100 in [26]. A way of reducing the number of random vectors needed to obtain good Monte Carlo simulations is proposed in [56]. A version of PMC using numerical integration, which is very computationally expensive, yielded the best results.

Figure 10.34 and Figure 10.35 compare the values estimated through the lognormal approximation to the true value, where for simplicity we deal with scalars. Thus x, n, and y represent the log-spectral energies of the clean signal, noise, and noisy signal, respectively, for a given frequency. Assuming x and n to be Gaussian with means μ_x and μ_n and variances σ_x and σ_n respectively, we see that the lognormal approximation is accurate when the standard deviations σ_x and σ_n are small.

Amazon/VB Assets
Exhibit 1012
Page 559

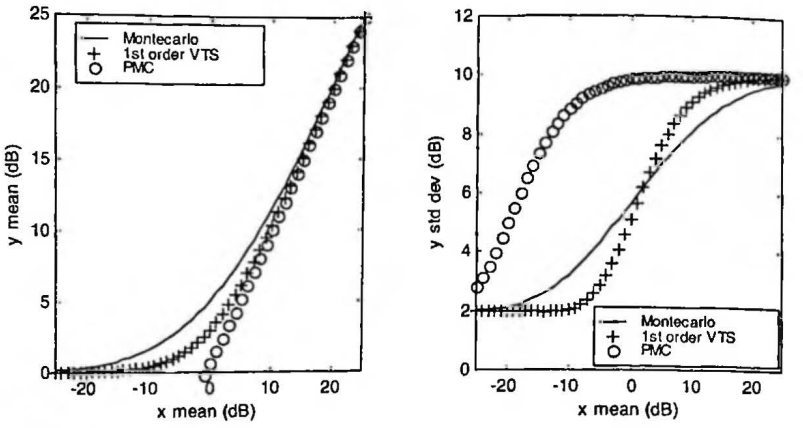


Figure 10.34 Means and standard deviation of noisy log-spectrum y in dB according to Eq. (10.165). The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation 2 dB. The distribution of the clean log-spectrum x is Gaussian, having a standard deviation of 10 dB and a mean varying from –25 to 25 dB. Both the mean and the standard deviation of y are more accurate in first-order VTS than in PMC.

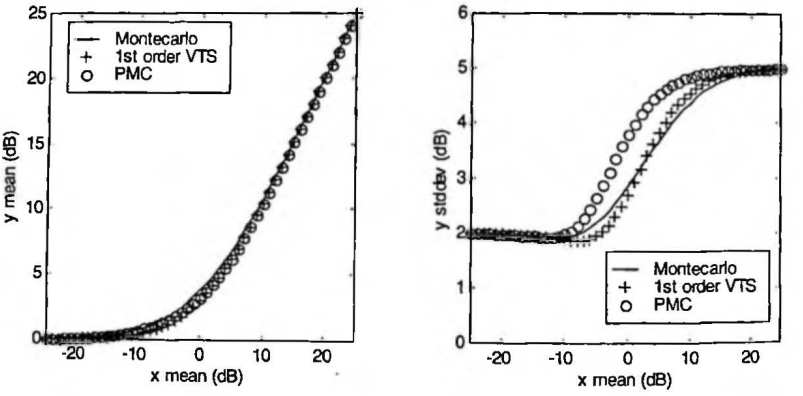


Figure 10.35 Means and standard deviation of noisy log-spectrum y in dB according to Eq. (10.165). The distribution of the noise log-spectrum n is Gaussian with mean 0 dB and standard deviation of 2 dB. The distribution of the clean log-spectrum x is Gaussian with a standard deviation of 5 dB and a mean varying from –25 dB to 25 dB. The mean of y is well estimated in both PMC and first-order VTS. The standard deviation of y is more accurate in first-order VTS than in PMC.

Amazon/VB Assets
Exhibit 1012
Page 560

10.6.4. Vector Taylor Series

The model of the acoustical environment described in Section 10.1.3 describes the relationship between the cepstral vectors **x**, **n**, and **y** of the clean speech, noise, and noisy speech, respectively:

$$\mathbf{y} = \mathbf{x} + \mathbf{h} + \mathbf{g}(\mathbf{n} - \mathbf{x} - \mathbf{h}) \tag{10.157}$$

where **h** is the cepstrum of the filter, and the nonlinear function $\mathbf{g}(\mathbf{z})$ is given by

$$\mathbf{g}(\mathbf{z}) = \mathbf{C}\ln\left(1 + e^{\mathbf{C}^{-1}\mathbf{z}}\right) \tag{10.158}$$

Moreno [44] suggests the use of Taylor series to approximate the nonlinearity in Eq. (10.158), though he applies it in the spectral instead of the cepstral domain. We follow that approach to compute the mean and covariance matrix of **y** [4].

Assume that **x**, **h**, and **n** are Gaussian random vectors with means μ_x, μ_h, and μ_n and covariance matrices Σ_x, Σ_h, and Σ_n, respectively, and furthermore that **x**, **h**, and **n** are independent. After algebraic manipulation it can be shown that the Jacobian of Eq. (10.157) with respect to **x**, **h**, and **n** evaluated at $\mu = \mu_n - \mu_x - \mu_h$ can be expressed as

$$\begin{aligned} \left.\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right|_{(\mu_n,\mu_x,\mu_h)} &= \left.\frac{\partial \mathbf{y}}{\partial \mathbf{h}}\right|_{(\mu_n,\mu_x,\mu_h)} = \mathbf{A} \\ \left.\frac{\partial \mathbf{y}}{\partial \mathbf{n}}\right|_{(\mu_n,\mu_x,\mu_h)} &= \mathbf{I} - \mathbf{A} \end{aligned} \tag{10.159}$$

where the matrix **A** is given by

$$\mathbf{A} = \mathbf{C}\mathbf{F}\mathbf{C}^{-1} \tag{10.160}$$

and **F** is a diagonal matrix whose elements are given by vector $\mathbf{f}(\mu)$, which in turn is given by

$$\mathbf{f}(\mu) = \frac{1}{1 + e^{\mathbf{C}^{-1}\mu}} \tag{10.161}$$

Using Eq. (10.159) we can then approximate Eq. (10.157) by a first-order Taylor series expansion around (μ_n, μ_x, μ_h) as

$$\begin{aligned} \mathbf{y} &\approx \mu_x + \mu_h + \mathbf{g}(\mu_n - \mu_x - \mu_h) \\ &+ \mathbf{A}(\mathbf{x} - \mu_x) + \mathbf{A}(\mathbf{h} - \mu_h) + (\mathbf{I} - \mathbf{A})(\mathbf{n} - \mu_n) \end{aligned} \tag{10.162}$$

The mean of **y**, μ_y, can be obtained from Eq. (10.162) as

Amazon/VB Assets
Exhibit 1012
Page 561

$$\boldsymbol{\mu}_y \approx \boldsymbol{\mu}_x + \boldsymbol{\mu}_h + \mathbf{g}(\boldsymbol{\mu}_n - \boldsymbol{\mu}_x - \boldsymbol{\mu}_h) \tag{10.163}$$

and its covariance matrix $\boldsymbol{\Sigma}_y$ by

$$\boldsymbol{\Sigma}_y \approx \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T + \mathbf{A}\boldsymbol{\Sigma}_h\mathbf{A}^T + (\mathbf{I}-\mathbf{A})\boldsymbol{\Sigma}_n(\mathbf{I}-\mathbf{A})^T \tag{10.164}$$

so that even if $\boldsymbol{\Sigma}_x$, $\boldsymbol{\Sigma}_h$, and $\boldsymbol{\Sigma}_n$ are diagonal, $\boldsymbol{\Sigma}_y$ is no longer diagonal. Nonetheless, we can assume it to be diagonal, because this way we can transform a clean HMM to a corrupted HMM that has the same functional form and use a decoder that has been optimized for diagonal covariance matrices.

It is difficult to visualize how good the approximation is, given the nonlinearity involved. To provide some insight, let's consider the frequency-domain version of Eqs. (10.157) and (10.158) when no filtering is done:

$$y = x + \ln\left(1 + \exp\left(n - x\right)\right) \tag{10.165}$$

where x, n, and y represent the log-spectral energies of the clean signal, noise, and noisy signal, respectively, for a given frequency. In Figure 10.34 we show the mean and standard deviation of the noisy log-spectral energy y in dB as a function of the mean of the clean log-spectral energy x with a standard deviation of 10 dB. The log-spectral energy of the noise n is Gaussian with mean 0 dB and standard deviation 2 dB. We compare the correct values obtained through Monte Carlo simulation (or DPMC) with the values obtained through the lognormal approximation of Section 10.6.3 and the first-order VTS approximation described here. We see that the VTS approximation is more accurate than the lognormal approximation for the mean and especially for the standard deviation of y, assuming the model of the environment described by Eq. (10.165).

Figure 10.35 is similar to Figure 10.34 except that the standard deviation of the clean log-energy x is only 5 dB, a more realistic number in speech recognition systems. In this case, both the lognormal approximation and the first-order VTS approximation are good estimates of the mean of y, though the standard deviation estimated through the lognormal approximation in PMC is not as good as that obtained through first-order VTS, again assuming the model of the environment described by Eq. (10.165). The overestimate of the variance in the lognormal approximation might, however, be useful if the model of the environment is not accurate.

To compute the means and covariance matrices of the delta and delta-delta parameters, let's take the derivative of the approximation of $\mathbf{y}$ in Eq. (10.162) with respect to time:

$$\frac{\partial \mathbf{y}}{\partial t} \approx \mathbf{A}\frac{\partial \mathbf{x}}{\partial t} \tag{10.166}$$

so that the delta-cepstrum computed through $\Delta\mathbf{x}_t = \mathbf{x}_{t+2} - \mathbf{x}_{t-2}$, is related to the derivative [28] by

Amazon/VB Assets
Exhibit 1012
Page 562

$$\Delta \mathbf{x} \approx 4 \frac{\partial \mathbf{x}_t}{\partial t} \tag{10.167}$$

so that

$$\mu_{\Delta y} \approx \mathbf{A} \mu_{\Delta x} \tag{10.168}$$

and similarly

$$\Sigma_{\Delta y} \approx \mathbf{A} \Sigma_{\Delta x} \mathbf{A}^T + (\mathbf{I} - \mathbf{A}) \Sigma_{\Delta n} (\mathbf{I} - \mathbf{A})^T \tag{10.169}$$

where we assumed that $\mathbf{h}$ is constant within an utterance, so that $\Delta \mathbf{h} = 0$.

Similarly, for the delta-delta cepstrum, the mean is given by

$$\mu_{\Delta^2 y} \approx \mathbf{A} \mu_{\Delta^2 x} \tag{10.170}$$

and the covariance matrix by

$$\Sigma_{\Delta^2 y} \approx \mathbf{A} \Sigma_{\Delta^2 x} \mathbf{A}^T + (\mathbf{I} - \mathbf{A}) \Sigma_{\Delta^2 n} (\mathbf{I} - \mathbf{A})^T \tag{10.171}$$

where we again assumed that $\mathbf{h}$ is constant within an utterance, so that $\Delta^2 \mathbf{h} = 0$.

Equations (10.163), (10.168), and (10.170) resemble the MLLR adaptation formulae of Chapter 9 for the means, though in this case the matrix is different for each Gaussian and is heavily constrained.

We are interested in estimating the environmental parameters μ_n, μ_h, and Σ_n, given a set of T observation frames $\mathbf{y}_t$. This estimation can be done iteratively using the EM algorithm on Eq. (10.162). If the noise process is stationary, $\Sigma_{\Delta n}$ could be approximated, assuming independence between $\mathbf{n}_{t+2}$ and $\mathbf{n}_{t-2}$, by $\Sigma_{\Delta n} = 2\Sigma_n$. Similarly, $\Sigma_{\Delta^2 n}$ could be approximated, assuming independence between $\Delta \mathbf{n}_{t+1}$ and $\Delta \mathbf{n}_{t-1}$, by $\Sigma_{\Delta^2 n} = 4\Sigma_n$. If the noise process is not stationary, it is best to estimate $\Sigma_{\Delta n}$ and $\Sigma_{\Delta^2 n}$ from input data directly.

If the distribution of $\mathbf{x}$ is a mixture of N Gaussians, each Gaussian is transformed according to the equations above. If the distribution of $\mathbf{n}$ is also a mixture of M Gaussians, the composite distribution has NM Gaussians. While this increases the number of Gaussians, the decoder is still functionally the same as for clean speech. Because normally you do not want to alter the number of Gaussians of the system when you do noise adaptation, it is often assumed that $\mathbf{n}$ is a single Gaussian.

10.6.5. Retraining on Compensated Features

We have discussed adapting the HMM to the new acoustical environment using the standard front-end features, in most cases the mel-cepstrum. Section 10.5 dealt with cleaning the noisy feature without retraining the HMMs. It's logical to consider a combination of both, where the features are cleaned to remove noise and channel effects and then the HMMs are retrained to take into account that this processing stage is not perfect. This idea is illustrated

Amazon/VB Assets
Exhibit 1012
Page 563

in Figure 10.36, where we compare the word error rate of the standard matched-noise-condition training with the matched-noise-condition training after it has been compensated by a variant of the mixture Gaussian algorithms described in Section 10.5.6 [21]. An improvement is obtained by retraining on compensated features, which beats the unprocessed matched-condition training.

The low error rates of both curves in Figure 10.36 are hard to obtain in practice, because they assume we know exactly what the noise level and type are ahead of time, which in general is hard to do. On the other hand, this could be combined with the multistyle training discussed in Section 10.6.1 or with a set of clustered models discussed in Chapter 9.

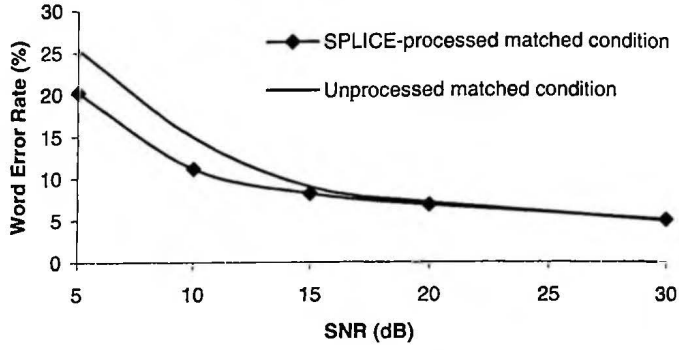


Figure 10.36 Word error rates of matched-noise training without feature preprocessing and with the SPLICE algorithm [21] as a function of the SNR in dB for additive white noise. Whisper is trained as in Figure 10.30. Error rate with the mixture Gaussian model is up to 30% lower than that of standard noisy matched conditions for low SNRs while it is about the same for high SNRs.

10.7. MODELING NONSTATIONARY NOISE

The previous sections deal mostly with stationary noise. In practice, there are many nonstationary noises that often match a random word in the system's lexicon better than the silence model. In this case, the benefit of using speech recognition vanishes quickly.

The most typical types of noise present in desktop applications are mouth noise (lip smacks, throat clearings, coughs, nasal clearings, heavy breathing, uhms and uhs, etc), computer noise (keyboard typing, microphone adjustment, computer fan, disk head seeking, etc.), and office noise (phone rings, paper rustles, shutting door, interfering speakers, etc.). We can use a simple method that has been successful in speech recognition [57], as shown in Algorithm 10.4. This method consists of adding noise words modeled with HMMs to absorb these nonstationary noises.

Amazon/VB Assets
Exhibit 1012
Page 564

In practice, updating the transcription turns out to be important, because human labelers often miss short noises that the system can uncover. Since the noise training data are often limited in terms of coverage, some noises can be easily matched to short word models, such as: *if*, *two*. Due to the unique characteristics of noise rejection, we often need to further augment confidence measures such as those described in Chapter 9. In practice, we need an additional classifier to provide more detailed discrimination between speech and noise. We can use a two-level classifier for this purpose. The ratio between the *all-speech model* score (fully connected context-independent phone models) and the *all-noise model score* (fully connected silence and noise phone models) can be used.

Another approach [55] consists of having an HMM for noise with several states to deal with nonstationary noises. The decoder needs to conduct a three-dimensional Viterbi search which evaluates at each frame every possible speech state as well as every possible noise state to achieve the *speech/noise decomposition* (see Figure 10.37). The computational complexity of such an approach is very large, though it can handle nonstationary noises quite well in theory.

ALGORITHM 10.4: *EXPLICIT NOISE MODELING*

Step 1: Augmenting the vocabulary with *noise words* (such as ++SMACK++), each composed of a single *noise phoneme* (such as +SMACK+), which are thus modeled with a single HMM. These noise words have to be labeled in the transcriptions so that they can be trained.
Step 2: Training noise models, as well as the other models, using the standard HMM training procedure.
Step 3: Updating the transcription. To do that, convert the transcription into a network, where the noise words can be optionally inserted between each word in the original transcription. A forced alignment segmentation is then conducted with the current HMM optional noise words inserted. The segmentation with the highest likelihood is selected, thus yielding an optimal transcription.
Step 4: If converged, stop; otherwise go to Step 2.

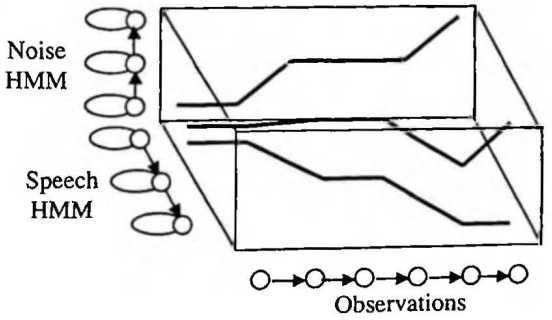


Figure 10.37 Speech noise decomposition and a three-dimensional Viterbi decoder.

Amazon/VB Assets
Exhibit 1012
Page 565

10.8. HISTORICAL PERSPECTIVE AND FURTHER READING

This chapter contains a number of diverse topics that are often described in different fields; no single reference covers it all. For further reading on adaptive filtering, you can check the books by Widrow and Stearns [59] and Haykin [30]. Theodoridis and Bellanger provide [54] a good summary of adaptive filtering, and Breining et al. [16] a good summary of echo-canceling techniques. Lee [38] has a good summary of independent component analysis for blind source separation. Deller et al. [20] provide a number of techniques for speech enhancement. Juang [35] and Junqua [37] survey techniques used in improving the robustness of speech recognition systems to noise. Acero [2] compares a number of feature transformation techniques in the cepstral domain and introduces the model of the environment used in this chapter.

Adaptive filtering theory emerged early in the 1900s. The Wiener and LMS filters were derived by Wiener and Widrow in 1919 and 1960, respectively. Norbert Wiener joined the MIT faculty in 1919 and made profound contributions to generalized harmonic analysis, the famous Wiener-Hopf equation, and the resulting Wiener filter. The LMS algorithm was developed by Widrow and his colleagues at Stanford University in the early 1960s.

From a practical point of view, the use of gradient microphones (Olsen [46]) has proven to be one of the more important contributions to increased robustness. Directional microphones are commonplace today in most speech recognition systems.

Boll [13] first suggested the use of spectral subtraction. This has been the cornerstone for noise suppression, and many systems nowadays still use a variant of Boll's original algorithm.

The Cepstral mean normalization algorithm was proposed by Atal [8] in 1974, although it wasn't until the early 1990s that it became commonplace in most speech recognition systems evaluated in the DARPA speech programs [33]. Hermansky proposed PLP [31] in 1990. The work of Rich Stern's robustness group at CMU (especially the Ph.D. thesis work of Acero [1] and Moreno [43]) and the Ph.D. thesis of Gales [26] also represented advances in the understanding of the effect of noise in the cepstrum.

Bell and Sejnowski [10] gave the field of independent component analysis a boost in 1995 with their infomax rule. The field of source separation is a promising alternative to improve the robustness of speech recognition systems when more than one microphone is available.

REFERENCES

[1] Acero, A., Acoustical and Environmental Robustness in Automatic Speech Recognition, PhD Thesis in Electrical and Computer Engineering 1990, Carnegie Mellon University, Pittsburgh.

[2] Acero, A., Acoustical and Environmental Robustness in Automatic Speech Recognition, 1993, Boston, Kluwer Academic Publishers.

Amazon/VB Assets
Exhibit 1012
Page 566

[3] Acero, A., S. Altschuler, and L. Wu, "Speech/Noise Separation Using Two Microphones and a VQ Model of Speech Signals," Int. Conf. on Spoken Language Processing, 2000, Beijing, China.

[4] Acero, A., et al., "HMM Adaptation Using Vector Taylor Series for Noisy Speech Recognition," Int. Conf. on Spoken Language Processing, 2000, Beijing, China.

[5] Acero, A. and X.D. Huang, "Augmented Cepstral Normalization for Robust Speech Recognition," Proc. of the IEEE Workshop on Automatic Speech Recognition, 1995, Snowbird, UT.

[6] Acero, A. and R. Stern, "Environmental Robustness in Automatic Speech Recognition," Int. Conf. on Acoustics, Speech and Signal Processing, 1990, Albuquerque, NM, pp. 849-852.

[7] Amari, S., A. Cichocki, and H.H. Yang, eds. A New Learning Algorithm for Blind Separation, Advances in Neural Information Processing Systems, 1996, Cambridge, MA, MIT Press.

[8] Atal, B.S., "Effectiveness of Linear Prediction Characteristics of the Speech Wave for Automatic Speaker Identification and Verification," Journal of the Acoustical Society of America, 1974, **55**(6), pp. 1304-1312.

[9] Attias, H., "Independent Factor Analysis," Neural Computation, 1998, **11**, pp. 803-851.

[10] Bell, A.J. and T.J. Sejnowski, "An Information Maximisation Approach to Blind Separation and Blind Deconvolution," Neural Computation, 1995, **7**(6), pp. 1129-1159.

[11] Belouchrani, A., et al., "A Blind Source Separation Technique Using Second Order Statistics," IEEE Trans. on Signal Processing, 1997, **45**(2), pp. 434-444.

[12] Berouti, M., R. Schwartz, and J. Makhoul, "Enhancement of Speech Corrupted by Acoustic Noise," Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1979, pp. 208-211.

[13] Boll, S.F., "Suppression of Acoustic Noise in Speech Using Spectral Subtraction," IEEE Trans. on Acoustics, Speech and Signal Processing, 1979, **27**(Apr.), pp. 113-120.

[14] Boll, S.F. and D.C. Pulsipher, "Suppression of Acoustic Noise in Speech Using Two Microphone Adaptive Noise Cancellation," IEEE Trans. on Acoustics Speech and Signal Processing, 1980, **28**(December), pp. 751-753.

[15] Bregman, A.S., Auditory Scene Analysis, 1990, Cambridge MA, MIT Press.

[16] Breining, C., Acoustic Echo Control, in IEEE Signal Processing Magazine, 1999. pp. 42-69.

[17] Cardoso, J., "Blind Signal Separation: Statistical Principles," Proc. of the IEEE, 1998, **9**(10), pp. 2009-2025.

[18] Cardoso, J.F., "Infomax and Maximum Likelihood for Blind Source Separation," IEEE Signal Processing Letters, 1997, **4**, pp. 112-114.

[19] Comon, P., "Independent Component Analysis: A New Concept," Signal Processing, 1994, **36**, pp. 287-314.

Amazon/VB Assets
Exhibit 1012
Page 567

[20] Deller, J.R., J.H.L. Hansen, and J.G. Proakis, Discrete-Time Processing of Speech Signals, 2000, IEEE Press.

[21] Deng, L., et al., "Large-Vocabulary Speech Recognition Under Adverse Acoustic Environments," Int. Conf. on Spoken Language Processing, 2000, Beijing, China.

[22] Ephraim, Y., "Statistical Model-Based Speech Enhancement System," Proc. of the IEEE, 1992, **80**(1), pp. 1526-1555.

[23] Ephraim, Y. and D. Malah, "Speech Enhancement Using Minimum Mean Square Error Short Time Spectral Amplitude Estimator," IEEE Trans. on Acoustics, Speech and Signal Processing, 1984, **32**(6), pp. 1109-1121.

[24] Flanagan, J.L., et al., "Computer-Steered Microphone Arrays for Sound Transduction in Large Rooms," Journal of the Acoustical Society of America, 1985, **78**(5), pp. 1508-1518.

[25] Frost, O.L., "An Algorithm for Linearly Constrained Adaptive Array Processing," Proc. of the IEEE, 1972, **60**(8), pp. 926-935.

[26] Gales, M.J., Model Based Techniques for Noise Robust Speech Recognition, PhD Thesis in Engineering Department, 1995, Cambridge University.

[27] Ghitza, O., "Robustness against Noise: The Role of Timing-Synchrony Measurement," Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1987, pp. 2372-2375.

[28] Gopinath, R.A., et al., "Robust Speech Recognition in Noise—Performance of the IBM Continuous Speech Recognizer on the ARPA Noise Spoke Task," Proc. ARPA Workshop on Spoken Language Systems Technology, 1995, pp. 127-133.

[29] Griffiths, L.J. and C.W. Jim, "An Alternative Approach to Linearly Constrained Adaptive Beamforming," IEEE Trans. on Antennas and Propagation, 1982, **30**(1), pp. 27-34.

[30] Haykin, S., Adaptive Filter Theory, 2nd ed, 1996, Upper Saddle River, NJ, Prentice-Hall.

[31] Hermansky, H., "Perceptual Linear Predictive (PLP) Analysis of Speech," Journal of the Acoustical Society of America, 1990, **87**(4), pp. 1738-1752.

[32] Hermansky, H. and N. Morgan, "RASTA Processing of Speech," IEEE Trans. on Speech and Audio Processing, 1994, **2**(4), pp. 578-589.

[33] Huang, X.D., et al., "The SPHINX-II Speech Recognition System: An Overview," Computer Speech and Language, 1993, pp. 137-148.

[34] Hunt, M. and C. Lefebre, "A Comparison of Several Acoustic Representations for Speech Recognition with Degraded and Undegraded Speech," Int. Conf. on Acoustic, Speech and Signal Processing, 1989, pp. 262-265.

[35] Juang, B.H., "Speech Recognition in Adverse Environments," Computer Speech and Language, 1991, **5**, pp. 275-294.

[36] Junqua, J.C., "The Lombard Reflex and Its Role in Human Listeners and Automatic Speech Recognition," Journal of the Acoustical Society of America, 1993, **93**(1), pp. 510-524.

[37] Junqua, J.C. and J.P. Haton, Robustness in Automatic Speech Recognition, 1996, Kluwer Academic Publishers.

Amazon/VB Assets
Exhibit 1012
Page 568

[38] Lee, T.W., Independent Component Analysis: Theory and Applications, 1998, Kluwer Academic Publishers.

[39] Lippmann, R.P., E.A. Martin, and D.P. Paul, "Multi-Style Training for Robust Isolated-Word Speech Recognition," Int. Conf. on Acoustics, Speech and Signal Processing, 1987, Dallas, TX, pp. 709-712.

[40] Lombard, E., "Le Signe de l'élévation de la Voix," Ann. Maladies Oreille, Larynx, Nez, Pharynx, 1911, **37**, pp. 101-119.

[41] Matassoni, M., M. Omologo, and D. Giuliani, "Hands-Free Speech Recognition Using a Filtered Clean Corpus and Incremental HMM Adaptation," Proc. Int. Conf. on Acoustics, Speech and Signal Processing, 2000, Istanbul, Turkey, pp. 1407-1410.

[42] Mendel, J.M., Lessons in Estimation Theory for Signal Processing, Communications, and Control, 1995, Upper Saddle River, NJ, Prentice Hall.

[43] Moreno, P., Speech Recognition in Noisy Environments, PhD Thesis in Electrical and Computer Engineering 1996, Carnegie Mellon University, Pittsburgh.

[44] Moreno, P.J., B. Raj, and R.M. Stern, "A Vector Taylor Series Approach for Environment Independent Speech Recognition," Int. Conf. on Acoustics, Speech and Signal Processing, 1996, Atlanta, pp. 733-736.

[45] Morgan, N. and H. Bourlard, Continuous Speech Recognition: An Introduction to Hybrid HMM/Connectionist Approach, in IEEE Signal Processing Magazine, 1995, pp. 25-42.

[46] Olsen, H.F., "Gradient Microphones," Journal of the Acoustical Society of America, 1946, **17**,(3), pp. 192-198.

[47] Porter, J.E. and S.F. Boll, "Optimal Estimators for Spectral Restoration of Noisy Speech," Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1984, San Diego, CA, pp. 18.A.2.1-4.

[48] Rahim, M.G. and B.H. Juang, "Signal Bias Removal by Maximum Likelihood Estimation for Robust Telephone Speech Recognition," IEEE Trans. on Speech and Audio Processing, 1996, **4**(1), pp. 19-30.

[49] Seneff, S., "A Joint Synchrony/Mean-Rate Model of Auditory Speech Processing," Journal of Phonetics, 1988, **16**(1), pp. 55-76.

[50] Sharma, S., et al., "Feature Extraction Using Non-Linear Transformation for Robust Speech Recognition on the Aurora Database," Int. Conf. on Acoustics, Speech and Signal Processing, 2000, Istanbul, Turkey, pp. 1117-1120.

[51] Sullivan, T.M. and R.M. Stern, "Multi-Microphone Correlation-Based Processing for Robust Speech Recognition," Int. Conf. on Acoustics, Speech and Signal Processing, 1993, Minneapolis, pp. 2091-2094.

[52] Suzuki, Y., et al., "An Optimum Computer-Generated Pulse Signal Suitable for the Measurement of Very Long Impulse Responses," Journal of the Acoustical Society of America, 1995, **97**(2), pp. 1119-1123.

[53] Tamura, S. and A. Waibel, "Noise Reduction Using Connectionist Models," Int. Conf. on Acoustics, Speech and Signal Processing, 1988, New York, pp. 553-556.

Amazon/VB Assets
Exhibit 1012
Page 569

[54] Theodoridis, S. and M.G. Bellanger, Adaptive Filters and Acoustic Echo Control, in IEEE Signal Processing Magazine, 1999, pp. 12-41.

[55] Varga, A.P. and R.K. Moore, "Hidden Markov Model Decomposition of Speech and Noise," Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1990 pp. 845-848.

[56] Wan, E.A., R.V.D. Merwe, and A.T. Nelson, "Dual Estimation and the Unscented Transformation" in Advances in Neural Information Processing Systems, S.A. Solla, T.K. Leen, and K.R. Muller, eds. 2000, Cambridge, MA, MIT Press, pp. 666-672.

[57] Ward, W., "Modeling Non-Verbal Sounds for Speech Recognition," Proc. Speech and Natural Language Workshop, 1989, Cape Cod, MA, Morgan Kauffman, pp. 311-318.

[58] Widrow, B. and M.E. Hoff, "Adaptive Switching Algorithms," IRE Wescon Convention Record, 1960, pp. 96-104.

[59] Widrow, B. and S.D. Stearns, Adaptive Signal Processing, 1985, Upper Saddle River, NJ, Prentice Hall.

[60] Woodland, P.C., "Improving Environmental Robustness in Large Vocabulary Speech Recognition," Int. Conf. on Acoustics, Speech and Signal Processing, 1996, Atlanta, Georgia, pp. 65-68.

Amazon/VB Assets
Exhibit 1012
Page 570

CHAPTER 11

Language Modeling

Acoustic pattern matching, as discussed in Chapter 9, and knowledge about language are equally important in recognizing and understanding natural speech. Lexical knowledge (i.e., vocabulary definition and word pronunciation) is required, as are the syntax and semantics of the language (the rules that determine what sequences of words are grammatically well-formed and meaningful). In addition, knowledge of the pragmatics of language (the structure of extended discourse, and what people are likely to say in particular contexts) can be important to achieving the goal of spoken language understanding systems. In practical speech recognition, it may be impossible to separate the use of these different levels of knowledge, since they are often tightly integrated.

In this chapter we review the basic concept of Chomsky's formal language theory and the probabilistic language model. For the formal language model, two things are fundamental: the grammar and the parsing algorithm. The *grammar* is a formal specification of the permissible structures for the language. The *parsing* technique is the method of analyzing the sentence to see if its structure is compliant with the grammar. With the advent of bodies

Amazon/VB Assets
Exhibit 1012
Page 571

of text (*corpora*) that have had their structures hand-annotated, it is now possible to generalize the formal grammar to include accurate probabilities. Furthermore, the probabilistic relationship among a sequence of words can be directly derived and modeled from the corpora with the so-called stochastic language models, such as *n*-gram, avoiding the need to create broad coverage formal grammars. Stochastic language models play a critical role in building a working spoken language system, and we discuss a number of important issues associated with them.

11.1. FORMAL LANGUAGE THEORY

In constructing a syntactic grammar for a language, it is important to consider the generality, the selectivity, and the understandability of the grammar. The *generality* and *selectivity* basically determine the range of sentences the grammar accepts and rejects. The *understandability* is important, since it is up to the authors of the system to create and maintain the grammar. For SLU systems described in Chapter 17, we need to have a grammar that covers and generalizes to most of the typical sentences for an application. The system also needs to distinguish the kind of sentences for different actions in a given application. Without understandability, it is almost impossible to improve a practical SLU system since it typically involves a large number of developers to maintain and refine the grammar.

The most common way of representing the grammatical structure of a sentence, "*Mary loves that person,*" is by using a tree, as illustrated in Figure 11.1. The node labeled *S* is the parent node of the nodes labeled *NP* and *VP* for noun phrase and verb phrase, respectively. The *VP* node is the parent node of node *V*—for verb. Each leaf is associated with the word

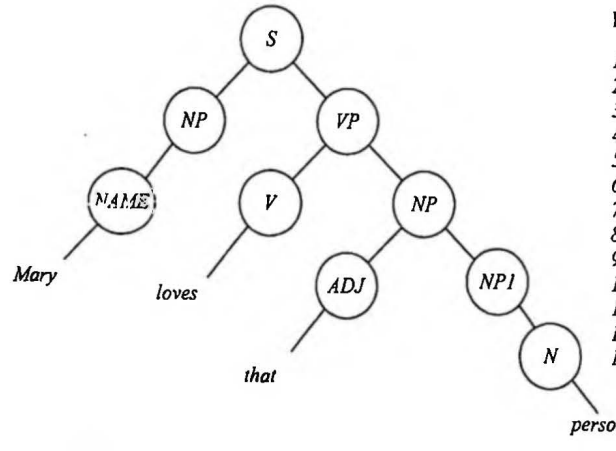


Figure 11.1 A tree representation of a sentence and its corresponding grammar.

Amazon/VB Assets
Exhibit 1012
Page 572

in the sentence to be analyzed. To construct such a tree for a sentence, we must know the structure of the language so that a set of rewrite rules can be used to describe what tree structures are allowable. These rules, as illustrated in Figure 11.1, determine that a certain symbol may be expanded in the tree by a sequence of symbols. The grammatical structure helps in determining the meaning of the sentence. It tells us that *that* in the sentence modifies *person*. "Mary loves *that* person."

11.1.1. Chomsky Hierarchy

In Chomsky's *formal language theory* [1, 14, 15], a grammar is defined as $G = (V, T, P, S)$, where V and T are finite sets of *non-terminals* and *terminals*, respectively. V contains all the *non-terminal* symbols. We often use upper-case symbols to denote them. In the example discussed here, *S, NP, NP1, VP, NAME, ADJ, N,* and V are non-terminal symbols. The *terminal* set T contains *Mary, loves, that,* and *person*, which are often denoted with lower-case symbols. P is a finite set of *production (rewrite) rules*, as illustrated in the rewrite rules in Figure 11.1. S is a special non-terminal, called the *start symbol.*

The language to be analyzed is essentially a string of terminal symbols, such as "*Mary loves that person.*" It is produced by applying production rules sequentially to the start symbol. The production rule is of the form $\alpha \rightarrow \beta$, where α and β are arbitrary strings of grammar symbols V and T, and the α must not be empty. In formal language theory, four major languages and their associated grammars are hierarchically structured. They are referred to as the Chomsky hierarchy [1] as defined in Table 11.1. There are four kinds of automata that can accept the languages produced by these four types of grammars. Among these automata, the finite-state automaton is not only the mathematical device used to implement the regular grammar but also one of the most significant tools in computational linguistics. Variations of automata such as finite-state transducers, hidden Markov models, and n-gram models are important examples in spoken language processing.

These grammatical formulations can be compared according to their generative capacity, i.e., the range that the formalism can cover. While there is evidence that natural languages are at least weakly context sensitive, the context-sensitive requirements are rare in practice. The context-free grammar (CFG) is a very important structure for dealing with both machine language and natural language. CFGs are not only powerful enough to describe most of the structure in spoken language,[1] but also restrictive enough to have efficient parsers to analyze natural sentences. Since CFGs offer a good compromise between parsing efficiency and power in representing the structure of the language, they have been widely applied to natural language processing. Alternatively, regular grammars, as represented with a finite-state machine, can be applied to more restricted applications. Since finite-state grammars are a subset of the more general context-free grammar, we focus our discussion on context-free grammars only, although the parsing algorithm for finite-state grammars can be more efficient.

[1] The effort to prove natural languages are not context-free is summarized in Pullum and Gazdar [54].

Amazon/VB Assets
Exhibit 1012
Page 573

Table 11.1 Chomsky hierarchy and the corresponding machine that accepts the language.

Types	Constraints	Automata
Phrase structure grammar	$\alpha \rightarrow \beta$. This is the most general grammar.	Turing machine
Context-sensitive grammar	A subset of the phrase structure grammar. $\lvert\alpha\rvert \leq \lvert\beta\rvert$, where |.| indicates the length of the string.	Linear bounded automata
Context-free grammar (CFG)	A subset of the context sensitive grammar. The production rule is $A \rightarrow \beta$, where A is a non-terminal. This production rule is shown to be equivalent to Chomsky normal form: $A \rightarrow w$ and $A \rightarrow BC$, where w is a terminal and B, C are non-terminals.	Push down automata
Regular grammar	A subset of the CFG. The production rule is expressed as: $A \rightarrow w$ and $A \rightarrow wB$.	Finite-state automata

As discussed in Section 11.1.2, a parsing algorithm offers a procedure that searches through various ways of combining grammatical rules to find a combination that generates a tree to illustrate the structure of the input sentence, which is similar to the search problem in speech recognition. The result of the parsing algorithm is a parse tree,[2] which can be regarded as a record of the CFG rules that account for the structure of the sentence. In other words, if we parse the sentence, working either top-down from S or bottom-up from each word, we automatically derive something that is similar to the tree representation, as illustrated in Figure 11.1.

A push-down automaton is also called a *recursive transition network* (RTN), which is an alternative formalism to describe context-free grammars. A transition network consists of nodes and labeled arcs. One of the nodes is specified as the initial state S. Starting at the initial state, we traverse an arc if the current word in the sentence is in the category on the arc. If the arc is followed, the current word is updated to the next word. A phrase can be parsed if there is a path from the starting node to a *pop* arc that indicates a complete parse for all the words in the phrase. Simple transition networks without recursion are often called *finite-state machines* (FSM). Finite-state machines are equivalent in expressive power to regular grammars and, thus, are not powerful enough to describe all languages that can be described by CFGs. Chapter 12 has a more detailed discussion on RTNs and FSMs used in speech recognition.

[2] The result can be more than one parse tree since natural language sentences are often ambiguous. In practice, a parsing algorithm should not only consider all the possible parse trees but also provide a ranking among them, as discussed in Chapter 17.

Amazon/VB Assets
Exhibit 1012
Page 574

11.1.2. Chart Parsing for Context-Free Grammars

Since Chomsky introduced the notion of context-free grammars in the 1950s, a vast literature has arisen on the parsing algorithms. Most parsing algorithms were developed in computer science to analyze programming languages that are not ambiguous in the way that spoken language is [1, 32]. We discuss only the most relevant materials that are fundamental to building spoken language systems, namely the chart parser for the context-free grammar. This algorithm has been widely used in state-of-the-art spoken language understanding systems.

11.1.2.1. Top Down or Bottom Up?

Parsing is a special case of the search problem generally encountered in speech recognition. A parsing algorithm offers a procedure that searches through various ways of combining grammatical rules to find a combination that generates a tree to describe the structure of the input sentence, as illustrated in Figure 11.1. The search procedure can start from the root of the tree with the *S* symbol, attempting to rewrite it into a sequence of terminal symbols that matches the words in the input sentence, which is based on *goal-directed search*. Alternatively, the search procedure can start from the words in the input sentence and identify a word sequence that matches some non-terminal symbol. The bottom-up procedure can be repeated with partially parsed symbols until the root of the tree or the start symbol *S* is identified. This *data-directed search* has been widely used in practical SLU systems.

A top-down approach starts with the *S* symbol, then searches through different ways to rewrite the symbols until the input sentence is generated, or until all possibilities have been examined. A grammar is said to accept a sentence if there is a sequence of rules that allow us to rewrite the start symbol into the sentence. For the grammar in Figure 11.1, a sequence of rewrite rules can be illustrated as follows:

```
S
→ NP VP (rewriting S using S→NP)
→NAME VP (rewriting NP using NP→NAME)
→Mary VP (rewriting NAME using NAME→Mary)
...
→Mary loves that person (rewriting N using N→person)
```

Alternatively, we can take a bottom-up approach to start with the words in the input sentence and use the rewrite rules backward to reduce the sequence of symbols until it becomes S. The left-hand side of each rule is used to rewrite the symbol on the right-hand side as follows:

```
→NAME loves that person (rewriting Mary using NAME→Mary)
→NAME V that person (rewriting loves using V→loves)
...
→NP VP
→S (rewriting NP using S→NP VP)
```

Amazon/VB Assets
Exhibit 1012
Page 575

A parsing algorithm must systematically explore every possible state that represents the intermediate node in the parsing tree. If a mistake occurs early on in choosing the rule that rewrites *S*, the intermediate parser results can be quite wasteful if the number of rules becomes large.

The main difference between top-down and bottom-up parsers is the way the grammar rules are used. For example, consider the rule *NP→ADJ NP1*. In a top-down approach, the rule is used to identify an *NP* by looking for the sequence *ADJ NP1*. Top-down parsing can be very predictive. A phrase or a word may be ambiguous in isolation. The top-down approach may prevent some ungrammatical combinations from consideration. It never wastes time exploring trees that cannot result in an *S*. On the other hand, it may predict many different constituents that do not have a match to the input sentence and rebuild large constituents again and again. For example, when the grammar is *left-recursive* (i.e., it contains a non-terminal category that has a derivation that includes itself anywhere along its leftmost branch), the top-down approach can lead a top-down, depth-first left-to-right parser to recursively expand the same non-terminal over again in exactly the same way. This causes an infinite expansion of trees. In contrast, a bottom-up parser takes a sequence *ADJ NP1* and identifies it as an *NP* according to the rule. The basic operation in bottom-up parsing is to take a sequence of symbols and match it to the right-hand side of the rules. It checks the input only once, and only builds each constituent exactly once. However, it may build up trees that have no hope of leading to *S* since it never suggests trees that are not at least locally grounded in the actual input. Since bottom-up parsing is similar to top-down parsing in terms of overall performance and is particularly suitable for robust spoken language processing as described in Chapter 17, we use the bottom-up method as our example to understand the key concept in the next section.

11.1.2.2. Bottom-Up Chart Parsing

As a standard search procedure, the state of the search consists of a symbol list, starting with the words in the sentence. Successor states can be generated by exploring all possible ways to replace a sequence of symbols that matches the right-hand side of a grammar rule with its left-hand side symbol. A simple-minded solution enumerates all the possible matches, leading to prohibitively expensive computational complexity. To avoid this problem, it is necessary to store partially parsed results of the matching, thereby eliminating duplicate work. This is the same technique that has been widely used in dynamic programming, as described in Chapter 8. Since chart parsing does not need to be from left to right, it is more efficient than the graph search algorithm discussed in Chapter 12, which can be used to parse the input sentence from left to right.

A data structure, called a *chart,* is used to allow the parser to store the partial results of the matching. The chart data structure maintains not only the records of all the constituents derived from the sentence so far in the parse tree, but also the records of rules that have matched partially but are still incomplete. These are called *active arcs*. Here, matches are always considered from the point of view of some *active constituents*, which represent the

Amazon/VB Assets
Exhibit 1012
Page 576

subparts that the input sentence can be divided into according to the rewrite rules. Active constituents are stored in a data structure called an *agenda*. To find grammar rules that match a string involving the active constituent, we need to identify rules that start with the active constituent or rules that have already been started by earlier active constituents and require the current constituent to complete the rule or to extend the rule. The basic operation of a chart-based parser involves combining these partially matched records (active arcs) with a completed constituent to form either a new completed constituent or a new partially matched (but incomplete) constituent that is an extension of the original partially matched constituent. Just like the graph search algorithm, we can use either a depth-first or breadth-first search strategy, depending on how the agenda is implemented. If we use probabilities or other heuristics, we take the best-first strategy discussed in Chapter 12 to select constituents from the agenda. The chart-parser process is defined more precisely in Algorithm 11.1. It is possible to combine both top-down and bottom-up. The major difference is how the constituents are used.

ALGORITHM 11.1: *A BOTTOM-UP CHART PARSER*

Step1: Initialization: Define a list called chart to store active arcs, and a list called an agenda to store active constituents until they are added to the chart.
Step 2: Repeat: Repeat Step 2 to 7 until there is no input left.
Step 3: Push and pop the agenda: If the agenda is empty, look up the interpretations of the next word in the input and push them to the agenda. Pop a constituent *C* from the agenda. If *C* corresponds to position from w_i to w_j of the input sentence, we denote it *C[i,j]*.
Step 4: Add *C* to the chart: Insert *C[i,j]* into the chart.
Step 5: Add key-marked active arcs to the chart: For each rule in the grammar of the form $X \rightarrow C\ Y$, add to the chart an active arc (partially matched constituent) of the form $X[i,j] \rightarrow °CY$, where ° denotes the critical position called the key that indicates that everything before ° has been seen, but things after ° are yet to be matched (incomplete constituent).
Step 6: Move ° forward: For any active arc of the form $X[1,j] \rightarrow Y...°C...Z$ (everything before w_j) in the chart, add a new active arc of the form $X[1,j] \rightarrow Y...C°...Z$ to the chart.
Step 7: Add new constituents to the agenda: For any active arc of the form $X[1,l] \rightarrow Y...°C$, add a new constituent of type *X[1,j]* to the agenda.
Step 8: Exit: If *S*[*1,n*] is in the chart, where *n* is the length of the input sentence, we can exit successfully unless we want to find all possible interpretations of the sentence. The chart may contain many *S* structures covering the entire set of positions.

Let us look at an example to see how the chart parser parses the sentence *Mary loves that person* using the grammar specified in Figure 11.1. We first create the chart and agenda data structure as illustrated in Figure 11.2 (a), in which the leaves of the tree-like chart data structure corresponds to the position of each input word. The parent of each block in the chart covers from the position of the left child's corresponding starting word position to the right child's corresponding ending word position. Thus, the root block in the chart covers the whole sentence from the first word *Mary* to the last word *person*. The chart parser scans

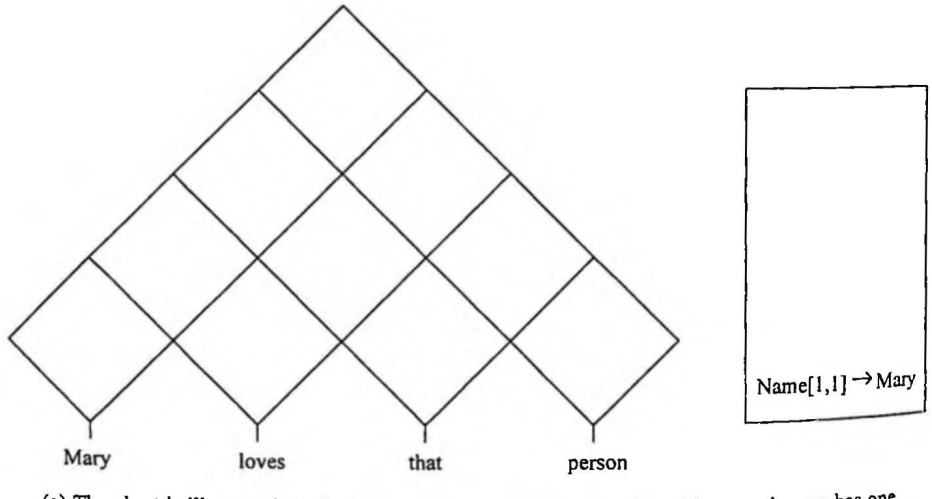
Amazon/VB Assets
Exhibit 1012
Page 577

through the input words to match against possible rewrite rules in the grammar. For the first word, the rule *Name→Mary* can be matched, so it is added to the agenda according to Step 3 in Algorithm 11.1. In Step 4, *Name→Mary* is added to the chart from the agenda. After the word Mary is processed, we have *Name→Mary, NP→Name*, and *S→NP°VP* in the chart, as illustrated in Figure 11.2 (b). *NP°VP* in the chart indicates that ° has reached the point at which everything before ° has been matched (in this case *Mary* matched *NP*) but everything after ° is yet to be parsed. The completed parsed chart is illustrated in Figure 11.2 (c).

A parser may assign one or more parsed structures to the sentence in the language it defines. If any sentence is assigned more than one such structure, the grammar is said to be ambiguous. Spoken language is, of course, ambiguous by nature.[3] For example, we can have a sentence like *Mary sold the student bags*. It is unclear whether *student* should be the modifier for *bags* or whether it means that *Mary* sold the *bags* to *the student*.

Chart parsers can be fairly efficient simply because the same constituent is never constructed more than once. In the worst case, the chart parser builds every possible constituent between every possible pair of positions, leading to the worst-case computational complexity of $O(n^3)$, where n is the length of the input sentence. This is still far more efficient than a straightforward brute-force search.

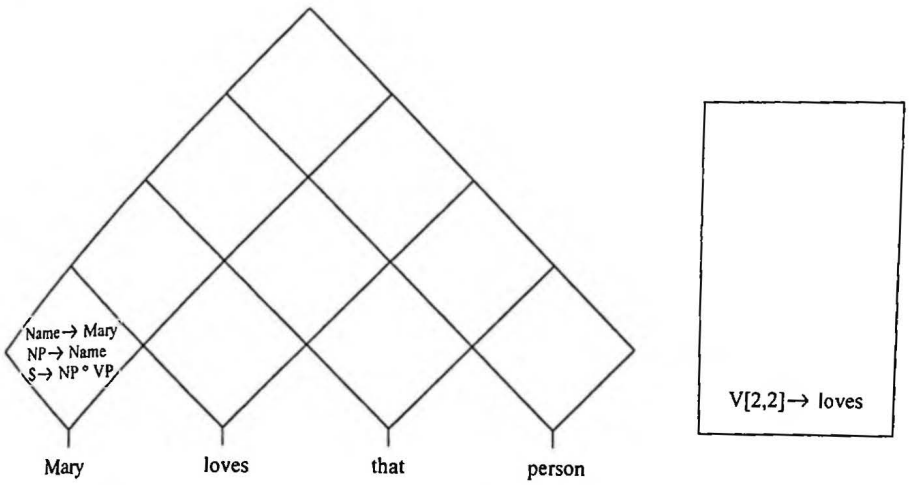
In many practical tasks, we need only a partial parse or shallow parse of the input sentence. You can use cascades of finite-state automata instead of CFGs. Relying on simple finite-state automata rather than full parsing makes such systems more efficient, although finite-state systems cannot model certain kinds of recursive rules, so that efficiency is traded for a certain lack of coverage.



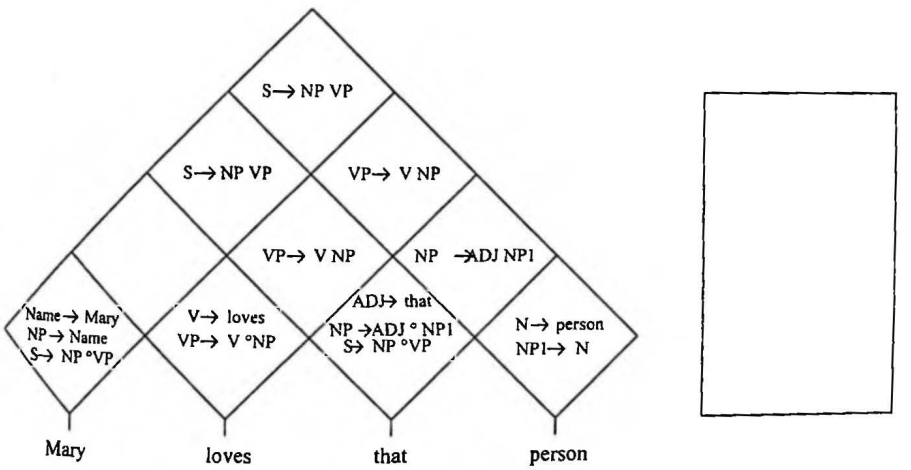
(a) The chart is illustrated on the left, and the agenda is on the right. The agenda now has one rule in it according to Step 3, since the agenda is empty.

[3] The same parse tree can also mean multiple things, so a parse tree itself does not define meaning. "*Mary loves that person*" could be sarcastic and mean something different.

Amazon/VB Assets
Exhibit 1012
Page 578



(b) After Mary, the chart now has rules *Name→Mary, NP→Name*, and *S→NP°VP*.



(c) The chart after the whole sentence is parsed. S→ NP VP covers the whole sentence, indicating that the sentence is parsed successfully by the grammar.

Figure 11.2 An example of a chart parser with the grammar illustrated in Figure 11.1. Parts (a) and (b) show the initial chart and agenda to parse the first word; part (c) shows the chart after the sentence is completely parsed.

Amazon/VB Assets
Exhibit 1012
Page 579

11.2. STOCHASTIC LANGUAGE MODELS

Stochastic language models (SLM) take a probabilistic viewpoint of language modeling. We need to accurately estimate the probability $P(\mathbf{W})$ for a given word sequence $\mathbf{W} = w_1 w_2 ... w_n$. In the formal language theory discussed in Section 11.1, $P(\mathbf{W})$ can be regarded as 1 or 0 if the word sequence is accepted or rejected, respectively, by the grammar. This may be inappropriate for spoken language systems, since the grammar itself is unlikely to have a complete coverage, not to mention that spoken language is often ungrammatical in real conversational applications.

The key goal of SLM is to provide adequate probabilistic information so that likely word sequences should have a higher probability. This not only makes speech recognition more accurate but also helps to dramatically constrain the search space for speech recognition (see Chapters 12 and 13). Notice that SLM can have a wide coverage on all the possible word sequences, since probabilities are used to differentiate different word sequences. The most widely used SLM is the so call *n*-gram model discussed in this chapter. In fact, the CFG can be augmented as the bridge between the *n*-gram and the formal grammar if we can incorporate probabilities into the production rules, as discussed in the next section.

11.2.1. Probabilistic Context-Free Grammars

The CFG can be augmented with probability for each production rule. The advantages of probabilistic CFGs (PCFGs) lie in their ability to more accurately capture the embedded usage structure of spoken language to minimize syntactic ambiguity. The use of probability becomes increasingly important to discriminate many competing choices when the number of rules is large.

In the PCFG, we have to address the parallel problems we discussed for HMMs in Chapter 8. The *recognition problem* is concerned with the computation of the probability of the start symbol S generating the word sequence $\mathbf{W} = w_1, w_2, ... w_T$, given the grammar G:

$$P(S \Rightarrow \mathbf{W}|G) \tag{11.1}$$

where $\Rightarrow$ denotes a derivation sequence consisting of one or more steps. This is equivalent to the chart parser augmented with probabilities, as discussed in Section 11.1.2.2.

The *training problem* is concerned with determining a set of rules G based on the training corpus and estimating the probability of each rule. If the set of rules is fixed, the simplest approach to deriving these probabilities is to count the number of times each rule is used in a corpus containing parsed sentences. We denote the probability of a rule $A \rightarrow \alpha$ by $P(A \rightarrow \alpha|G)$. For instance, if there are m rules for left-hand side non-terminal node $A: A \rightarrow \alpha_1, A \rightarrow \alpha_2, ... A \rightarrow \alpha_m$, we can estimate the probability of these rules as follows:

$$P(A \rightarrow \alpha_j \mid G) = C(A \rightarrow \alpha_j) / \sum_{i=1}^{m} C(A \rightarrow \alpha_i) \tag{11.2}$$

where $C(.)$ denotes the number of times each rule is used.

Amazon/VB Assets
Exhibit 1012
Page 580

When you have hand-annotated corpora, you can use the maximum likelihood estimation as illustrated by Eq. (11.2) to derive the probabilities. When you don't have hand-annotated corpora, you can extend the EM algorithm (see Chapter 4) to derive these probabilities. The algorithm is also known as the *inside-outside* algorithm. As we discussed in Chapter 8, you can develop algorithms similar to the Viterbi algorithm to find the most likely parse tree that could have generated the sequence of words $P(\mathbf{W})$ after these probabilities are estimated.

We can make certain independence assumptions about rule usage. Namely, we assume that the probability of a constituent being derived by a rule is independent of how the constituent is used as a subconstituent. For instance, we assume that the probabilities of *NP* rules are the same no matter whether the *NP* is used for the subject or the object of a verb, although the assumptions are not valid in many cases. More specifically, let the word sequence $\mathbf{W}=w_1 w_2 \ldots w_T$ be generated by a PCFG G, with rules in Chomsky normal form as discussed in Section 11.1.1:

$$A_i \rightarrow A_m A_n \text{ and } A_i \rightarrow w_l \tag{11.3}$$

where A_m and A_n are two possible non-terminals that expand A_i at different locations. The probability for these rules must satisfy the following constraint:

$$\sum_{m,n} P(A_i \rightarrow A_m A_n \mid G) + \sum_{l} P(A_i \rightarrow w_l \mid G) = 1, \quad \text{for all } i \tag{11.4}$$

Equation (11.4) simply means that all non-terminals can generate either pairs of non-terminal symbols or a single terminal symbol, and all these production rules should satisfy the probability constraint. Analogous to the HMM forward and backward probabilities discussed in Chapter 8, we can define the inside and outside probabilities to facilitate the estimation of these probabilities from the training data.

A non-terminal symbol A_i can generate a sequence of words $w_j w_{j+1} \ldots w_k$; we define the probability of $Inside(j, A_i, k) = P(A_i \Rightarrow w_j w_{j+1} \ldots w_k \mid G)$ as the *inside constituent probability*, since it assigns a probability to the word sequence inside the constituent. The inside probability can be computed recursively. When only one word is emitted, the transition rule of the form $A_i \rightarrow w_m$ applies. When there is more than one word, rules of the form $A_i \rightarrow A_m A_n$ must apply. The inside probability of $inside(j, A_i, k)$ can be expressed recursively as follows:

$$\begin{aligned} inside(j, A_i, k) &= P(A_i \Rightarrow w_j w_{j+1} \ldots w_k) \\ &= \sum_{n,m} \sum_{l=j}^{k-1} P(A_i \rightarrow A_m A_n) P(A_m \Rightarrow w_j \ldots w_l) P(A_n \Rightarrow w_{l+1} \ldots w_k) \\ &= \sum_{n,m} \sum_{l=j}^{k-1} P(A_i \rightarrow A_m A_n) inside(j, A_m, l) inside(l+1, A_n, k) \end{aligned} \tag{11.5}$$

Amazon/VB Assets
Exhibit 1012
Page 581

The inside probability is the sum of the probabilities of all derivations for the section over the span of j to k. One possible derivation of the form can be drawn as a parse tree shown in Figure 11.3.

Another useful probability is the *outside* probability for a non-terminal node A_i covering w_s to w_t, in which they can be derived from the start symbol S, as illustrated in Figure 11.4, together with the rest of the words in the sentence:

$$outside(s, A_i, t) = P(S \Rightarrow w_1 \ldots w_{s-1}\ A_i\ w_{t+1} \ldots w_T) \tag{11.6}$$

After the inside probabilities are computed bottom-up, we can compute the outside probabilities top-down. For each non-terminal symbol A_i, there are one of two possible configurations $A_m \rightarrow A_n A_i$ or $A_m \rightarrow A_i A_n$ as illustrated in Figure 11.5. Thus, we need to consider all the possible derivations of these two forms as follows:

$$\begin{aligned} outside(s, A_i, t) &= P(S \Rightarrow w_1 \ldots w_{s-1}\ A_i\ w_{t+1} \ldots w_T) \\ &= \sum_{m,n} \left\{ \begin{array}{l} \sum_{l=1}^{s-1} P(A_m \rightarrow A_n A_i) P(A_n \Rightarrow w_l \ldots w_{s-1}) P(S \Rightarrow w_1 \ldots w_{l-1}\ A_m\ w_{t+1} \ldots w_T) + \\ + \sum_{l=t+1}^{T} P(A_m \rightarrow A_i A_n) P(A_n \Rightarrow w_{t+1} \ldots w_l) P(S \Rightarrow w_1 \ldots w_{s-1}\ A_m\ w_{l+1} \ldots w_T) \end{array} \right\} \\ &= \sum_{m,n} \left\{ \begin{array}{l} \sum_{l=1}^{s-1} P(A_m \rightarrow A_n A_i) inside(l, A_n, s-1) outside(l, A_m, t) + \\ + \sum_{l=t+1}^{T} P(A_m \rightarrow A_i A_n) inside(t+1, A_n, l) outside(s, A_m, l) \end{array} \right\} \end{aligned} \tag{11.7}$$

The inside and outside probabilities are used to compute the sentence probability as follows:

$$P(S \Rightarrow w_1 \ldots w_T) = \sum_i inside(s, A_i, t) outside(s, A_i, t) \qquad \text{for any } s \le t \tag{11.8}$$

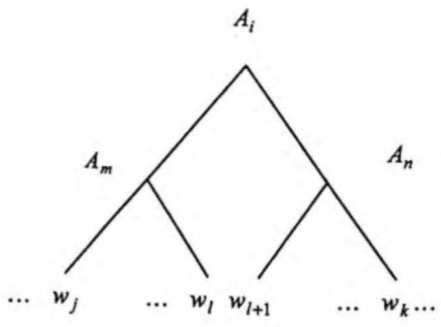


Figure 11.3 Inside probability is computed recursively as sum of all the derivations.

Amazon/VB Assets
Exhibit 1012
Page 582

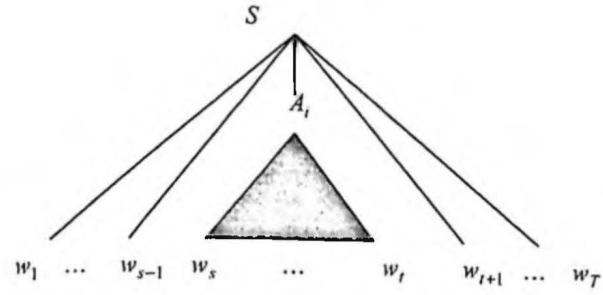


Figure 11.4 Definition of the outside probability.

Since $outside(1, A_j, T)$ is equal to 1 for the starting symbol only, the probability for the whole sentence can be conveniently computed using the inside probability alone as

$$P(S \Rightarrow \mathbf{W}|G) = inside(1, S, T) \tag{11.9}$$

We are interested in the probability that a particular rule, $A_i \rightarrow A_m A_n$ is used to cover a span $w_s ... w_t$, given the sentence and the grammar:

$$\xi(i,m,n,s,t) = P(A_i \Rightarrow w_s ... w_t, A_i \rightarrow A_m A_n \mid S \Rightarrow \mathbf{W}, G)$$
$$= \frac{1}{P(S \Rightarrow \mathbf{W} \mid G)} \sum_{k=s}^{t-1} P(A_i \rightarrow A_m A_n \mid G) inside(s, A_m, k) inside(k+1, A_n, t) outside(s, A_i, t) \tag{11.10}$$

These conditional probabilities form the basis of the inside-outside algorithm, which is similar to the forward-backward algorithm discussed in Chapter 8. We can start with some initial probability estimates. For each sentence of training data, we determine the inside and outside probabilities in order to compute, for each production rule, how likely it is that the production rule is used as part of the derivation of that sentence. This gives us the number of counts for each production rule in each sentence. Summing these counts across sentences gives us an estimate of the total number of times each production rule is used to produce the

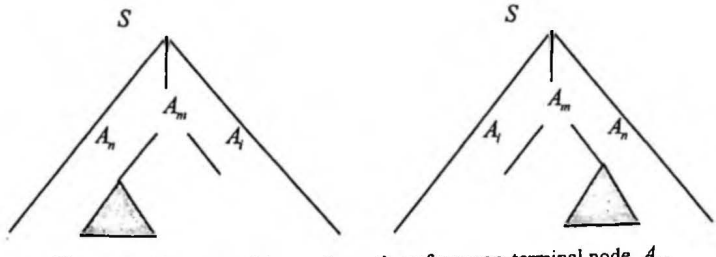


Figure 11.5 Two possible configurations for a non-terminal node A_j.

Amazon/VB Assets
Exhibit 1012
Page 583