$$\mathbf{A}_{LSE} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.112}$$

$(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ in Eq. (3.112) is also refereed to as the *pseudo-inverse* of $\mathbf{X}$ and is sometimes denoted as $\mathbf{X}^{\perp}$.

When $\mathbf{X}'\mathbf{X}$ is singular or some boundary conditions cause the LSE estimation in Eq. (3.112) to be unattainable, some numeric methods can be used to find an approximate solution. Instead of minimizing the quantity in Eq. (3.109), one can minimize the following quantity:

$$e(\mathbf{A}) = \| \mathbf{XA} - \mathbf{Y} \|^2 + \alpha \| \mathbf{A} \|^2 \tag{3.113}$$

Following a similar procedure, one can obtain the LSE estimate to minimize the quantity above in the following form.

$$\mathbf{A}^{*}_{LSE} = (\mathbf{X}'\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y} \tag{3.114}$$

The LSE solution in Eq. (3.112) can be used for polynomial functions too. In the problem of polynomial curve fitting using the least square criterion, we are aiming to find the coefficients $\mathbf{A} = (a_0, a_1, a_2, \cdots, a_d)'$ that minimize the following quantity:

$$\min_{a_0, a_1, a_2, \cdots, a_d} E(Y - \hat{Y})^2 \tag{3.115}$$

where $\hat{Y} = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$

To obtain the LSE estimate of coefficients $\mathbf{A} = (a_0, a_1, a_2, \cdots, a_d)'$, simply change the formation of matrix $\mathbf{X}$ in Eq. (3.108) to the following:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^d \\ 1 & x_2 & \cdots & x_2^d \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^d \end{pmatrix} \tag{3.116}$$

Note that x_i^j in Eq. (3.108) means the j-th dimension of sample $\mathbf{x}_i$, while x_i^j in Eq. (3.116) means j-th order of value x_i. Therefore, the LSE estimate of polynomial coefficients $\mathbf{A}_{LSE} = (a_0, a_1, a_2, \cdots, a_d)'$ has the same form as Eq. (3.112).

3.2.1.3. MMSE/LSE for Nonlinear Functions

As the most general case, consider solving the following minimization problem:

$$\min_{g(\bullet) \in G_{all}} E\left[Y - g(X)\right]^2 \tag{3.117}$$

Since we need to deal with all possible nonlinear functions, taking a derivative does not work here. Instead, we use the property of conditional expectation to solve this minimization problem. By applying Eq. (3.38) to (3.117), we get

$$\begin{aligned} E_{X,Y}\left[Y-g(X)\right]^2 &= E_X\left\{E_{Y|X}\left[\left[Y-g(X)\right]^2 \mid X=x\right]\right\} \\ &= \int_{-\infty}^{\infty} E_{Y|X}\left[\left[Y-g(X)\right]^2 \mid X=x\right] f_X(x)dx \\ &= \int_{-\infty}^{\infty} E_{Y|X}\left[\left[Y-g(x)\right]^2 \mid X=x\right] f_X(x)dx \end{aligned} \tag{3.118}$$

Since the integrand is nonnegative in Eq. (3.118), the quantity in Eq. (3.117) will be minimized at the same time the following equation is minimized.

$$\min_{g(x)\in R} E_{Y|X}\left[\left[Y-g(x)\right]^2 \mid X=x\right] \tag{3.119}$$

Since $g(x)$ is a constant in the calculation of the conditional expectation above, the MMSE estimate can be obtained in the same way as the constant functions in Section 3.2.1.1. Thus, the MMSE estimate should take the following form:

$$\hat{Y} = g_{MMSE}(X) = E_{Y|X}(Y \mid X) \tag{3.120}$$

If the value $X=x$ is observed and the value $E(Y \mid X=x)$ is used to predict Y, the mean squared error (MSE) is minimized and specified as follows:

$$E_{Y|X}\left[\left[Y-E_{Y|X}(Y \mid X=x)\right]^2 \mid X=x\right] = Var_{Y|X}(Y \mid X=x) \tag{3.121}$$

The overall MSE, averaged over all the possible values of X, is:

$$E_X\left[Y-E_{Y|X}(Y \mid X)\right]^2 = E_X\left\{E_{Y|X}\left[\left[Y-E_{Y|X}(Y \mid X)\right]^2 \mid X\right]\right\} = E_X\left[{}_{Y|X}Var(Y \mid X=x)\right] \tag{3.122}$$

It is important to distinguish between the overall MSE $E_X\left[Var_{Y|X}(Y \mid X)\right]$ and the MSE of the particular estimate when $X=x$, which is $Var_{Y|X}(Y \mid X=x)$. Before the value of X is observed, the expected MSE for the process of observing X and predicting Y is $E_X\left[Var_{Y|X}(Y \mid X)\right]$. On the other hand, after a particular value x of X has been observed and the prediction $E_{Y|X}(Y \mid X=x)$ has been made, the appropriate measure of MSE of the prediction is $Var_{Y|X}(Y \mid X=x)$.

In general, the form of the MMSE estimator for nonlinear functions depends on the form of the joint distribution of X and Y. There is no mathematical closed-form solution. To get the conditional expectation in Eq. (3.120), we have to perform the following integral:

$$\hat{Y}(x) = \int_{-\infty}^{\infty} y f_Y(y \mid X=x)dy \tag{3.123}$$

It is difficult to solve this integral calculation. First, different measures of x could determine different conditional pdf for the integral. Exact information about the pdf is often impossible to obtain. Second, there could be no analytic solution for the integral. Those difficulties reduce the interest of the MMSE estimation of nonlinear functions to theoretical aspects only. The same difficulties also exist for LSE estimation for nonlinear functions. Some certain classes of well-behaved nonlinear functions are typically assumed for LSE problems and numeric methods are used to obtain LSE estimate from sample data.

3.2.2. Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is the most widely used parametric estimation method, largely because of its efficiency. Suppose that a set of random samples $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is to be drawn independently according to a discrete or continuous distribution with the pf or the pdf $p(x|\Phi)$, where the parameter vector Φ belongs to some parameter space Ω. Given an observed vector $\mathbf{x} = (x_1, \cdots, x_n)$, the *likelihood* of the set of sample data vectors $\mathbf{x}$ with respect to Φ is defined as the joint pf or joint pdf $p_n(\mathbf{x}|\Phi)$; $p_n(\mathbf{x}|\Phi)$ is also referred to as the *likelihood function*.

MLE assumes the parameters of pdfs are fixed but unknown and aims to find the set of parameters that maximizes the likelihood of generating the observed data. For example, if the pdf $p_n(\mathbf{x}|\Phi)$ is assumed to be a Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the components of Φ will then include exactly the components of mean-vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Since $X_1, X_2, \ldots, X_n$ are independent random variables, the likelihood can be rewritten as follows:

$$p_n(\mathbf{x}|\Phi) = \prod_{k=1}^{n} p(x_k|\Phi) \tag{3.124}$$

The likelihood $p_n(\mathbf{x}|\Phi)$ can be viewed as the probability of generating the sample data set $\mathbf{x}$ based on parameter set Φ. The *maximum likelihood estimator* of Φ is denoted as Φ_{MLE} that maximizes the likelihood $p_n(\mathbf{x}|\Phi)$. That is,

$$\Phi_{MLE} = \underset{\Phi}{\operatorname{argmax}}\, p_n(\mathbf{x}|\Phi) \tag{3.125}$$

This estimation method is called the *maximum likelihood estimation* method and is often abbreviated as MLE. Since the logarithm function is a monotonically increasing function, the parameter set Φ_{MLE} that maximizes the log-likelihood should also maximize the likelihood. If $p_n(\mathbf{x}|\Phi)$ is differentiable function of Φ, Φ_{MLE} can be attained by taking the partial derivative with respect to Φ and setting it to zero. Specifically, let Φ be a k-component parameter vector $\Phi = (\Phi_1, \Phi_2, \ldots, \Phi_k)^t$ and ∇_Φ be the gradient operator:

$$\nabla_{\Phi} = \begin{bmatrix} \dfrac{\partial}{\partial \Phi_1} \\ \vdots \\ \dfrac{\partial}{\partial \Phi_k} \end{bmatrix} \tag{3.126}$$

The log-likelihood becomes:

$$l(\Phi) = \log p_n(\mathbf{x} \mid \Phi) = \sum_{k=1}^{n} \log p(x_k \mid \Phi) \tag{3.127}$$

and its partial derivative is:

$$\nabla_{\Phi}\, l(\Phi) = \sum_{k=1}^{n} \nabla_{\Phi} \log p(x_k \mid \Phi) \tag{3.128}$$

Thus, the maximum likelihood estimate of Φ can be obtained by solving the following set of k equations:

$$\nabla_{\Phi}\, l(\Phi) = 0 \tag{3.129}$$

Example 3.1

Let's take a look at the maximum likelihood estimator of a univariate Gaussian pdf, given as the following equation:

$$p(x \mid \Phi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2} \right] \tag{3.130}$$

where μ and σ^2 are the mean and the variance respectively. The parameter vector Φ denotes (μ, σ^2). The log-likelihood is:

$$\begin{aligned} \log p_n(\mathbf{x} \mid \Phi) &= \sum_{k=1}^{n} \log p(x_k \mid \Phi) \\ &= \sum_{k=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x_k-\mu)^2}{2\sigma^2} \right] \right) \\ &= -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{k=1}^{n} (x_k - \mu)^2 \end{aligned} \tag{3.131}$$

and the partial derivative of the above expression is:

$$\frac{\partial}{\partial \mu} \log p_n(x \mid \Phi) = \sum_{k=1}^{n} \frac{1}{\sigma^2}(x_k - \mu)$$
$$\frac{\partial}{\partial \sigma^2} \log p_n(x \mid \Phi) = -\frac{n}{2\sigma^2} + \sum_{k=1}^{n} \frac{(x_k - \mu)^2}{2\sigma^4} \tag{3.132}$$

We set the two partial differential derivatives to zero,

$$\sum_{k=1}^{n} \frac{1}{\sigma^2}(x_k - \mu) = 0$$
$$-\frac{n}{\sigma^2} + \sum_{k=1}^{n} \frac{(x_k - \mu)^2}{\sigma^4} = 0 \tag{3.133}$$

The maximum likelihood estimates for μ and σ^2 are obtained by solving the above equations:

$$\mu_{MLE} = \frac{1}{n}\sum_{k=1}^{n} x_k = E(x)$$
$$\sigma^2_{MLE} = \frac{1}{n}\sum_{k=1}^{n} (x_k - \mu_{MLE})^2 = E\left[(x - \mu_{MLE})^2\right] \tag{3.134}$$

Equation (3.134) indicates that the maximum likelihood estimation for mean and variance is just the sample mean and variance.

Example 3.2

For the multivariate Gaussian pdf $p(\mathbf{x})$

$$p(\mathbf{x} \mid \Phi) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right] \tag{3.135}$$

The maximum likelihood estimates of $\boldsymbol{\mu}$ and Σ can be obtained by a similar procedure.

$$\hat{\boldsymbol{\mu}}_{MLE} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k$$
$$\hat{\Sigma}_{MLE} = \frac{1}{n}\sum_{k=1}^{n} (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{MLE})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{MLE})^t = E\left[(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{MLE})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{MLE})^t\right] \tag{3.136}$$

Once again, the maximum likelihood estimation for mean vector and covariance matrix is the sample mean vector and sample covariance matrix.

In some situations, maximum likelihood estimation of Φ may not exist, or the maximum likelihood estimator may not be uniquely defined, i.e., there may be more than one MLE of Φ for a specific set of sample values. Fortunately, according to Fisher's theorem, for most practical problems with a well-behaved family of distributions, the MLE exists and is uniquely defined [4, 25, 26].

In fact, the maximum likelihood estimator can be proven to be sound under certain conditions. As mentioned before, the estimator $\theta(\mathbf{X})$ is a function of the vector of random variables $\mathbf{X}$ that represent the sample data. $\theta(\mathbf{X})$ itself is also a random variable, with a distribution determined by joint distributions of $\mathbf{X}$. Let $\tilde{\Phi}$ be the parameter vector of true distribution $p(x|\Phi)$ from which the samples are drawn. If the following three conditions hold:

1. The sample $\mathbf{x}$ is a drawn from the assumed family of distribution,
2. The family of distributions is well behaved,
3. The sample $\mathbf{x}$ is large enough,

then maximum likelihood estimator, Φ_{MLE}, has a Gaussian distribution with a mean $\tilde{\Phi}$ and a variance of the form $1/nB_x^2$ [26], where n is the size of sample and B_x is the *Fisher information*, which is determined solely by $\tilde{\Phi}$ and $\mathbf{x}$. An estimator is said to be *consistent*, iff the estimate will converge to the true distribution when there is infinite number of training samples.

$$\lim_{n\to\infty} \Phi_{MLE} = \tilde{\Phi} \tag{3.137}$$

Φ_{MLE} is a consistent estimator based on the analysis above. In addition, it can be shown that no consistent estimator has a lower variance than Φ_{MLE}. In other words, no estimator provides a closer estimate of the true parameters than the maximum likelihood estimator.

3.2.3. Bayesian Estimation and MAP Estimation

Bayesian estimation has a different philosophy than maximum likelihood estimation. While MLE assumes that the parameter Φ[3] is fixed but unknown, Bayesian estimation assumes that the parameter Φ itself is a random variable with a prior distribution $p(\Phi)$. Suppose we observe a sequence of random samples $\mathbf{x}=\{x_1, x_2, \ldots, x_n\}$, which are i.i.d. with a pdf $p(x|\Phi)$. According to Bayes' rule, we have the posterior distribution of Φ as:

$$p(\Phi|\mathbf{x}) = \frac{p(\mathbf{x}|\Phi)p(\Phi)}{p(\mathbf{x})} \propto p(\mathbf{x}|\Phi)p(\Phi) \tag{3.138}$$

[3] For simplicity, we assume the parameter Φ is a scalar instead of a vector here. However, the extension to a parameter vector Φ can be derived according to a similar procedure.

In Eq. (3.138), we dropped the denominator $p(\mathbf{x})$ here because it is independent of the parameter Φ. The distribution in Eq. (3.138) is called the posterior distribution of Φ because it is the distribution of Φ after we observed the values of random variables $X_1, X_2, \ldots, X_n$.

3.2.3.1. Prior and Posterior Distributions

For mathematical tractability, conjugate priors are often used in Bayesian estimation. Suppose a random sample is taken of a known distribution with pdf $p(\mathbf{x} \mid \Phi)$. A conjugate prior for the random variable (or vector) is defined as the prior distribution for the parameters of the probability density function of the random variable (or vector), such that the class-conditional pdf $p(\mathbf{x} \mid \Phi)$, the posterior distribution $p(\Phi \mid \mathbf{x})$, and the prior distribution $p(\Phi)$ belong to the same distribution family. For example, it is well known that the conjugate prior for the mean of a Gaussian pdf is also a Gaussian pdf [4]. Now, let's derive such a posterior distribution $p(\Phi \mid \mathbf{x})$ from the widely used Gaussian conjugate prior.

Example 3.3

Suppose $X_1, X_2, \ldots, X_n$ are drawn from a Gaussian distribution for which the mean Φ is a random variable and the variance σ^2 is known. The likelihood function $p(\mathbf{x} \mid \Phi)$ can be written as:

$$p(\mathbf{x} \mid \Phi) = \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i - \Phi}{\sigma}\right)^2\right] \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(\frac{x_i - \Phi}{\sigma}\right)^2\right] \tag{3.139}$$

To further simply Eq. (3.139), we could use Eq. (3.140)

$$\sum_{i=1}^{n}(x_i - \Phi)^2 = n(\Phi - \bar{x}_n)^2 + \sum_{i=1}^{n}(x_i - \bar{x}_n)^2 \tag{3.140}$$

where $\bar{x}_n = \frac{1}{n}\sum_{i=1}^{n} x_i =$ the sample mean of $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$.

Let's rewrite $p(\mathbf{x} \mid \Phi)$ in Eq. (3.139) into Eq. (3.141):

$$p(\mathbf{x} \mid \Phi) \propto \exp\left[-\frac{n}{2\sigma^2}(\Phi - \bar{x}_n)^2\right] \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \bar{x}_n)^2\right] \tag{3.141}$$

Now suppose the prior distribution of Φ is also a Gaussian distribution with mean μ and variance ν^2, i.e., the prior distribution $p(\Phi)$ is given as follows:

$$p(\Phi) = \frac{1}{(2\pi)^{1/2}\nu} \exp\left[-\frac{1}{2}\left(\frac{\Phi - \mu}{\nu}\right)^2\right] \propto \exp\left[-\frac{1}{2}\left(\frac{\Phi - \mu}{\nu}\right)^2\right] \tag{3.142}$$

By combining Eqs. (3.141) and (3.142) while dropping the second term in Eq. (3.141) we could attain the posterior pdf $p(\Phi \mid \mathbf{x})$ in the following equation:

$$p(\Phi \mid \mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left[\frac{n}{\sigma^2}(\Phi - \bar{x}_n)^2 + \frac{1}{\nu^2}(\Phi - \mu)^2\right]\right\} \tag{3.143}$$

Now if we define ρ and τ as follows:

$$\rho = \frac{\sigma^2\mu + n\nu^2\bar{x}_n}{\sigma^2 + n\nu^2} \tag{3.144}$$

$$\tau^2 = \frac{\sigma^2\nu^2}{\sigma^2 + n\nu^2} \tag{3.145}$$

We can rewrite Eq. (3.143) as:

$$p(\Phi \mid \mathbf{x}) \propto \exp\left\{-\frac{1}{2}\left[\frac{1}{\tau^2}(\Phi - \rho)^2 + \frac{n}{\sigma^2 + n\nu^2}(\bar{x}_n - \mu)^2\right]\right\} \tag{3.146}$$

Since the second term in Eq. (3.146) does not depend on Φ, it can be absorbed in the constant factor. Finally, we have the posterior pdf in the following form:

$$p(\Phi \mid \mathbf{x}) = \frac{1}{\sqrt{2\pi}\tau}\exp\left[\frac{-1}{2\tau^2}(\Phi - \rho)^2\right] \tag{3.147}$$

Equation (3.147) shows that the posterior pdf $p(\Phi \mid \mathbf{x})$ is a Gaussian distribution with mean ρ and variance τ^2 as defined in Eqs. (3.144) and (3.145). The Gaussian prior distribution defined in Eq. (3.142) is a conjugate prior.

3.2.3.2. General Bayesian Estimation

The foremost requirement of a good estimator θ is that it can yield an estimate of Φ ($\theta(\mathbf{X})$) which is close to the real value Φ. In other words, a good estimator is one for which it is highly probable that the error $\theta(\mathbf{X}) - \Phi$ is close to 0. In general, we can define a loss function[4] $R(\Phi, \bar{\Phi})$. It measures the loss or cost associated with the fact that the true value of the parameter is Φ while the estimate is $\bar{\Phi}$. When only the prior distribution

[4] Bayesian estimation and loss functions are based on Bayes' decision theory, described in Chapter 4.

$p(\Phi)$ is available and no sample data has been observed, if we choose one particular estimate $\bar{\Phi}$, the expected loss is:

$$E\left[R(\Phi,\bar{\Phi})\right] = \int R(\Phi,\bar{\Phi})p(\Phi)d\Phi \tag{3.148}$$

The fact that we could derive posterior distribution from the likelihood function and the prior distribution [as shown in the derivation of Eq. (3.147)] is very important here because it allows us to compute the expected posterior loss after sample vector $\mathbf{x}$ is observed. The expected posterior loss associated with estimate $\bar{\Phi}$ is:

$$E\left[R(\Phi,\bar{\Phi}) \mid \mathbf{x}\right] = \int R(\Phi,\bar{\Phi})p(\Phi \mid \mathbf{x})d\Phi \tag{3.149}$$

The Bayesian estimator of Φ is defined as the estimator that attains minimum Bayes risk, that is, minimizes the expected posterior loss function (3.149). Formally, the Bayesian estimator is chosen according to:

$$\theta_{Bayes}(\mathbf{x}) = \underset{\theta}{\operatorname{argmin}}\ E\left[R(\Phi,\theta(\mathbf{x})) \mid \mathbf{x}\right] \tag{3.150}$$

The Bayesian estimator of Φ is the estimator θ_{Bayes} for which Eq. (3.150) is satisfied for every possible value of $\mathbf{x}$ of random vector $\mathbf{X}$. Therefore, the form of the Bayesian estimator θ_{Bayes} should depend only on the loss function and the prior distribution, but not the sample value $\mathbf{x}$.

One of the most common loss functions used in statistical estimation is the mean squared error function [20]. The mean squared error function for Bayesian estimation should have the following form:

$$R(\Phi,\theta(\mathbf{x})) = (\Phi - \theta(\mathbf{x}))^2 \tag{3.151}$$

In order to find the Bayesian estimator, we are seeking θ_{Bayes} to minimize the expected posterior loss function:

$$E\left[R(\Phi,\theta(\mathbf{x})) \mid \mathbf{x}\right] = E\left[(\Phi - \theta(\mathbf{x}))^2 \mid \mathbf{x}\right] = E(\Phi^2 \mid \mathbf{x}) - 2\theta(\mathbf{x})E(\Phi \mid \mathbf{x}) - \theta(\mathbf{x})^2 \tag{3.152}$$

The minimum value of this function can be obtained by taking the partial derivative of Eq. (3.152) with respect to $\theta(\mathbf{x})$. Since the above equation is simply a quadratic function of $\theta(\mathbf{x})$, it can be shown that the minimum loss can be achieved when θ_{Bayes} is chosen based on the following equation:

$$\theta_{Bayes}(\mathbf{x}) = E(\Phi \mid \mathbf{x}) \tag{3.153}$$

Equation (3.153) translates into the fact that the Bayesian estimate of the parameter Φ for mean squared error function is equal to the mean of the posterior distribution of Φ. In the following section, we discuss another popular loss function (MAP estimation) that also generates the same estimate for certain distribution functions.

3.2.3.3. MAP Estimation

One intuitive interpretation of Eq. (3.138) is that a prior pdf $p(\Phi)$ represents the relative likelihood before the values of $X_1, X_2, \ldots, X_n$ have been observed; while the posterior pdf $p(\Phi \mid \mathbf{x})$ represents the relative likelihood after the values of $X_1, X_2, \ldots, X_n$ have been observed. Therefore, choosing an estimate $\bar{\Phi}$ that maximizes the posterior probability is consistent without intuition. This estimator is in fact the *maximum posterior probability* (MAP) estimator and is the most popular Bayesian estimator.

The loss function associated with the MAP estimator is the so-called uniform loss function [20]:

$$R(\Phi, \theta(\mathbf{x})) = \begin{cases} 0, & \text{if } \mid \theta(\mathbf{x}) - \Phi \mid \leq \Delta \\ 1, & \text{if } \mid \theta(\mathbf{x}) - \Phi \mid > \Delta \end{cases} \quad \text{where } \Delta > 0 \tag{3.154}$$

Now let's see how this uniform loss function results in MAP estimation. Based on the loss function defined above, the expected posterior loss function is:

$$\begin{aligned} E(R(\Phi, \theta(\mathbf{x})) \mid \mathbf{x}) &= P(\mid \theta(\mathbf{x}) - \Phi \mid > \Delta \mid \mathbf{x}) \\ &= 1 - P(\mid \theta(\mathbf{x}) - \Phi \mid \leq \Delta \mid \mathbf{x}) = 1 - \int_{\theta(\mathbf{x})-\Delta}^{\theta(\mathbf{x})+\Delta} p(\Phi \mid \mathbf{x}) \end{aligned} \tag{3.155}$$

The quantity in Eq. (3.155) is minimized by maximizing the shaded area under $p(\Phi \mid \mathbf{x})$ over the interval $[\theta(\mathbf{x}) - \Delta, \theta(\mathbf{x}) + \Delta]$ in Figure 3.16. If $p(\Phi \mid \mathbf{x})$ is a smooth curve and Δ is small enough, the shaded area can be computed roughly as:

$$\int_{\theta(\mathbf{x})-\Delta}^{\theta(\mathbf{x})+\Delta} p(\Phi \mid \mathbf{x}) \cong 2\Delta p(\Phi \mid \mathbf{x})\Big|_{\Phi=\theta(\mathbf{x})} \tag{3.156}$$

Thus, the shaded area can be approximately maximized by choosing $\theta(\mathbf{x})$ to be the maximum point of $p(\Phi \mid \mathbf{x})$. This concludes our proof the using the error function in Eq. (3.154) indeed will generate MAP estimator.

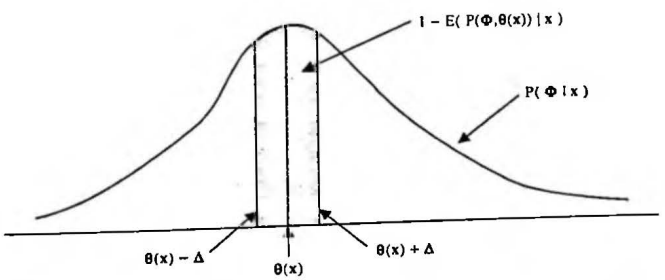

Figure 3.16 Illustration of the minimum expected posterior loss function for MAP estimation [20].

MAP estimation is to find the parameter estimate Φ_{MAP} or estimator $\theta_{MAP}(\mathbf{x})$ that maximizes the posterior probability,

$$\Phi_{MAP} = \theta_{MAP}(\mathbf{x}) = \underset{\Phi}{\text{argmax}}\ p(\Phi \mid \mathbf{x}) = \underset{\Phi}{\text{argmax}}\ p(\mathbf{x} \mid \Phi)p(\Phi) \tag{3.157}$$

Φ_{MAP} can also be specified in the logarithm form as follows:

$$\Phi_{MAP} = \underset{\Phi}{\text{argmax}}\ \left[\log p(\mathbf{x} \mid \Phi) + \log p(\Phi)\right] \tag{3.158}$$

Φ_{MAP} can be attained by solving the following partial differential equation:

$$\frac{\partial \log p(\mathbf{x} \mid \Phi)}{\partial \Phi} + \frac{\partial \log p(\Phi)}{\partial \Phi} = 0 \tag{3.159}$$

Thus the MAP equation for finding Φ_{MAP} can be established.

$$\left.\frac{\partial \log p(\mathbf{x} \mid \Phi)}{\partial \Phi}\right|_{\Phi=\Phi_{MAP}} = \left.\frac{-\partial \log p(\Phi)}{\partial \Phi}\right|_{\Phi=\Phi_{MAP}} \tag{3.160}$$

There are interesting relationships between MAP estimation and MLE estimation. The prior distribution is viewed as the knowledge of the statistics of the parameters of interest before any sample data is observed. For the case of MLE, the parameter is assumed to be fixed but unknown. That is, there is no preference (knowledge) of what the values of parameters should be. The prior distribution $p(\Phi)$ can only be set to constant for the entire parameter space, and this type of prior information is often referred to as *non-informative prior* or *uniform prior*. By substituting $p(\Phi)$ with a uniform distribution in Eq. (3.157), MAP estimation is identical to MLE. In this case, the parameter estimation is solely determined by the observed data. A sufficient amount of training data is often a requirement for MLE. On the other hand, when the size of the training data is limited, the use of the prior density becomes valuable. If some prior knowledge of the distribution of the parameters can be obtained, MAP estimation provides a way of incorporating prior information in the parameter learning process.

Example 3.4

Now, let's formulate MAP estimation for Gaussian densities. As described in Section 3.2.3.1, the conjugate prior distribution for a Gaussian density is also a Gaussian distribution. Similarly, we assumed random variables $X_1, X_2, \ldots, X_n$ drawn from a Gaussian distribution for which the mean Φ is unknown and the variance σ^2 is known, while the conjugate prior distribution of Φ is a Gaussian distribution with mean μ and variance ν^2. It is shown in Section 3.2.3.1 that the posterior pdf can be formulated as in Eq. (3.147). The MAP estimation for Φ can be solved by taking the derivative of Eq. (3.147) with respect to Φ:

$$\Phi_{MAP} = \rho = \frac{\sigma^2 \mu + n\nu^2 \bar{x}_n}{\sigma^2 + n\nu^2} \tag{3.161}$$

where n is the total number of training samples and $\bar{x}_n$ the sample mean.

The MAP estimate of the mean Φ is a weighted average of the sample mean $\bar{x}_n$ and the prior mean. When n is zero (when there is no training data at all), the MAP estimate is simply the prior mean μ. On the other hand, when n is large ($n \to \infty$), the MAP estimate will converge to the maximum likelihood estimate. This phenomenon is consistent with our intuition and is often referred to as *asymptotic equivalence* or *asymptotic convergence*. Therefore, in practice, the difference between MAP estimation and MLE is often insignificant when a large amount of training data is available. When the prior variance ν^2 is very large (e.g., $\nu^2 >> \sigma^2 / n$), the MAP estimate will converge to the ML estimate because a very large ν^2 translates into a non-informative prior.

It is important to note that the requirement of learning prior distributions for MAP estimation is critical. In some cases, the prior distribution is very difficult to estimate and MLE is still an attractive estimation method. As mentioned before, the MAP estimation framework is particularly useful for dealing with sparse data, such as parameter adaptation. For example, in speaker adaptation, the speaker-independent (or multiple speakers) database can be used to first estimate the prior distribution [9]. The model parameters are adapted to a target speaker through a MAP framework by using limited speaker-specific training data as discussed in Chapter 9.

3.3. SIGNIFICANCE TESTING

Significance testing is one of the most important theories and methods of statistical *inference*. A problem of statistical inference, or, more simply, a statistics problem, is one in which data that have been generated in accordance with some unknown probability distribution must be analyzed, and some type of inference about the unknown distribution must be made. Hundreds of test procedures have developed in statistics for various kinds of hypotheses testing. We focus only on tests that are used in spoken language systems.

The selection of appropriate models for the data or systems is essential for spoken language systems. When the distribution of certain sample data is unknown, it is usually appropriate to make some assumptions about the distribution of the data with a distribution function whose properties are well known. For example, people often use Gaussian distributions to model the distribution of background noise in spoken language systems. One important issue is how good our assumptions are, and what the appropriate values of the parameters for the distributions are, even when we can use the methods in Section 3.2 to estimate parameters from sample data. Statistical tests are often applied to determine if the distribution with specific parameters is appropriate to model the sample data. In this section, we describe the most popular testing method for the goodness of distribution fitting – the χ^2 goodness-of-fit test.

Another important type of statistical tests is designed to evaluate the excellence of two different methods or algorithms for the same tasks when there is uncertainty regarding the results. To assure that the two systems are evaluated on the same or similar conditions, experimenters often carefully choose similar or even the exactly same data sets for testing. This is why we refer to this type of statistical test as a *paired observations* test. In both speech recognition and speech synthesis, the paired observations test is a very important tool for interpreting the comparison results.

3.3.1. Level of Significance

We now consider statistical problems involving a parameter ϕ whose value is unknown but must lie in a certain parameter space Ω. In statistical tests, we let H_0 denote the hypothesis that $\phi \in \Omega_0$ and let H_1 denote the hypothesis that $\phi \in \Omega_1$. The subsets Ω_0 and Ω_1 are disjoint and $\Omega_0 \cup \Omega_1 = \Omega$, so exactly one of the hypotheses H_0 and H_1 must be true. We must now decide whether to accept H_0 or H_1 by observing a random sample $X_1, \cdots, X_n$ drawn from a distribution involving the unknown parameter ϕ. A problem like this is called hypotheses testing. A procedure for deciding whether to accept H_0 or H_1 is called a *test procedure* or simply a *test*. The hypothesis H_0 is often referred to as the *null hypothesis* and the hypothesis H_1 as the *alternative hypothesis*. Since there are only two possible decisions, accepting H_0 is equivalent to rejecting H_1 and rejecting H_0 is equivalent to accepting H_1. Therefore, in testing hypotheses, we often use the terms *accepting or rejecting the null hypothesis* H_0 as the only decision choices.

Usually we are presented with a random sample $\mathbf{X} = (X_1, \cdots, X_n)$ to help us in making the test decision. Let S denote the sample space of n-dimensional random vector $\mathbf{X}$. The testing procedure is equivalent to partitioning the sample space S into two subsets. One subset specifies the values of $\mathbf{X}$ for which one will accept H_0 and the other subset specifies the values of $\mathbf{X}$ for which one will reject H_0. The second subset is called the *critical region* and is often denoted as C.

Since there is uncertainty associated with the test decision, for each value of $\phi \in \Omega$, we are interested in the probability $\rho(\phi)$ that the testing procedure rejects H_0. The function $\rho(\phi)$ is called the *power function* of the test and can be specified as follows:

$$\rho(\phi) = P(\mathbf{X} \in C \mid \phi) \tag{3.162}$$

For $\phi \in \Omega_0$, the decision to reject H_0 is incorrect. Therefore, if $\phi \in \Omega_0$, $\rho(\phi)$ is the probability that the statistician will make an incorrect decision (false rejection). In statistical tests, an upper bound α_0 $(0 < \alpha_0 < 1)$ is specified, and we only consider tests for which $\rho(\phi) \le \alpha_0$ for every value of $\phi \in \Omega_0$. The upper bound α_0 is called the *level of significance*. The smaller α_0 is, the less likely it is that the test procedure will reject H_0. Since α_0 specifies the upper bound for false rejection, once a hypothesis is rejected by the test procedure, we can be $(1-\alpha_0)$ confident the decision is correct. In most applications, α_0 is set to be 0.05 and the test is said to be carried out at the 0.05 level of significance or 0.95 level of confidence.

We define the size α of a given test as the maximum probability, among all the values of ϕ which satisfy the null hypothesis, of making an incorrect decision.

$$\alpha = \max_{\theta \in \Omega_0} \rho(\phi) \tag{3.163}$$

Once we obtain the value of α, the test procedure is straightforward. First, the statistician specifies a certain level of significance α_0 in a given problem of testing hypotheses, then he or she rejects the null hypothesis if the size α is such that $\alpha \leq \alpha_0$.

The size α of a given test is also called the *tail area* or the *p-value* corresponding to the observed value of data sample **X** because it corresponds to tail area of the distribution. The hypothesis will be rejected if the level of significance α_0 is such that $\alpha_0 > \alpha$ and should be accepted for any value of $\alpha_0 < \alpha$. Alternatively, we can say the observed value of **X** is *just significant* at the level of significance α without using the level of significance α_0. Therefore, if we had found that the observed value of one data sample **X** was just significant at the level of 0.0001, while the other observed value of data sample **Y** was just significant at the level of 0.001, then we can conclude the sample **X** provides much stronger evidence against H_0. In statistics, an observed value of one data sample **X** is generally said to be statistically significant if the corresponding tail area is smaller than the traditional value 0.05. For cases requiring more significance (confidence), 0.01 can be used.

A statistically significant observed data sample **X** that provides strong evidence against H_0 does not necessary provide strong evidence that the actual value of ϕ is significantly far away from parameter set Ω_0. This situation can arise, particularly when the size of random data sample is large, because a test with larger sample size will in general reject hypotheses with more confidence, unless the hypothesis is indeed the true one.

3.3.2. Normal Test (Z-Test)

Suppose we need to find whether a coin toss is fair or not. Let p be the probability of heads. The hypotheses to be tested are as follows:

$H_0 : p = ½$

$H_1 : p \neq ½$

We assume that a random sample size n is taken, and let random variable M denote the number of times we observe heads as the result. The random variable M has a binomial distribution $B(n, ½)$. Because of the shape of binomial distribution, M can lie on either side of the mean. This is why it is called a typical *two-tailed test*. The tail area or p-value for the observed value k can be computed as:

$$p = \begin{cases} 2P(k \leq M \leq n) & \text{for } k > n/2 \\ 2P(0 \leq M \leq k) & \text{for } k < n/2 \\ 1.0 & \text{for } k = n/2 \end{cases} \tag{3.164}$$

The p-value in Eq. (3.164) can be computed directly using the binomial distribution. The test procedure will reject H_0 when p is less than the significance level α_0.

In many situations, the p-value for the distribution of data sample $\mathbf{X}$ is difficult to obtain due to the complexity of the distribution. Fortunately, if some statistic Z of the data sample $\mathbf{X}$ has some well-known distribution, the test can then be done in the Z domain instead. If n is large enough ($n > 50$), a *normal test* (or *Z-test*) can be used to approximate a binomial probability. Under H_0, the mean and variance for M are $E(M) = n/2$ and $Var(M) = n/4$. The new random variable Z is defined as,

$$Z = \frac{|M - n/2| - 1/2}{\sqrt{n/4}} \tag{3.165}$$

which can be approximated as standard Gaussian distribution $N(0,1)$ under H_0. The p-value can now be computed as $p = 2P(Z \geq z)$ where z is the realized value of Z after M is observed. Thus, H_0 is rejected if $p < \alpha_0$, where α_0 is the level of significance.

3.3.3. χ^2 Goodness-of-Fit Test

The normal test (Z-test) can be extended to test the hypothesis that a given set of data came from a certain distribution with all parameters specified. First let's look at the case of discrete distribution fitting.

Suppose that a large population consists of items of k different types and let p_i be the probability that a random selected item belongs to type i. Now, let $q_1,\ldots,q_k$ be a set of specific numbers satisfying the probabilistic constraint ($q_i \geq 0$ for $i = 1,\ldots,k$ and $\sum_{i=1}^{k} q_i = 1$). Finally, suppose that the following hypotheses are to be tested:

$H_0 : p_i = q_i$ for $i = 1,\ldots,k$

$H_1 : p_i \neq q_i$ for at least one value of i

Assume that a random sample of size n is to be taken from the given population. For $i = 1,\ldots,k$, let N_i denote the number of observations in the random sample which are of type i. Here, $N_1,\ldots,N_k$ are nonnegative numbers and $\sum_{i=1}^{k} N_i = n$. Random variables $N_1,\ldots,N_k$ have a multinomial distribution. Since the p-value for the multinomial distribution is hard to obtain, instead we use another statistic about $N_1,\ldots,N_k$. When H_0 is true, the expected number of observations of type i is nq_i. In other words, the difference between the actual number of observations N_i and the expected number nq_i should be small when H_0 is true. It seems reasonable to base the test on the differences $N_i - nq_i$ and to reject H_0 when the differences are large. It can be proved [14] that the following random variable λ

$$\lambda = \sum_{i=1}^{k} \frac{(N_i - nq_i)^2}{nq_i} \tag{3.166}$$

converges to the χ^2 distribution with $k-1$ degrees of freedom as the sample size $n \to \infty$.

A χ^2 test of goodness-of-fit can be carried out in the following way. Once a level of significance α_0 is specified, we can use the following p-value function to find critical point c:[5]

$$P(\lambda > c) = 1 - F_{\chi^2}(x = c) = \alpha_0 \tag{3.167}$$

where $F_{\chi^2}(x)$ is the distribution function for χ^2 distribution. The test procedure simply rejects H_0 when the realized value λ is such that $\lambda > c$. Empirical results show that the χ^2 distribution will be a good approximation to the actual distribution of λ as long as the value of each expectation nq_i is not too small (≥ 5). The approximation should still be satisfactory if $nq_i \geq 1.5$ for $i = 1,\ldots,k$.

For continuous distributions, a modified χ^2 goodness-of-fit test procedure can be applied. Suppose that we would like to hypothesize a null hypothesis H_0 in which continuous random sample data $X_1,\ldots,X_k$ are drawn from a certain continuous distribution with all parameters specified or estimated. Also, suppose the observed values of random sample $x_1,\ldots,x_k$ are bounded within interval Ω. First, we divide the range of the hypothesized distribution into m subintervals within interval Ω such that the expected number of values, say E_i, in each interval is at least 5. For $i = 1,\ldots,k$, we let N_i denote the number of observations in the i^{th} subintervals. As in Eq. (3.166), one can prove that the following random variable λ

$$\lambda = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i} \tag{3.168}$$

converges to the χ^2 distribution with $m-k-1$ degrees of freedom as the sample size $n \to \infty$, where k is the number of parameters that must be estimated from the sample data in order to calculate the expected number of values, E_i. Once the χ^2 distribution is established, the same procedure can be used to find the critical c in Eq. (3.167) to make test decisions.

Example 3.5

Suppose we are given a random variable X of sample size 100 points and we want to determine whether we can reject the following hypothesis:

$$H_0 : X \sim N(0,1) \tag{3.169}$$

To perform χ^2 goodness-of-fit test, we first divide the range of X into 10 subintervals. The corresponding probability falling in each subinterval, the expected number of points falling in each subinterval and the actual number of points falling in each subintervals [10] are illustrated in Table 3.1.

[5] Since χ^2 pdf is a monotonic function, the test is a one-tail test. Thus, we only need to calculate one tail area.

Table 3.1 The probability falling in each subinterval of an $N(0,1)$, and 100 sample points, the expected number of points falling in each subinterval, and the actual number of points falling in each subinterval [10].

Subinterval I_i	$P(X \in I_i)$	$E_i = 100P(X \in I_i)$	N_i
$[-\infty, -1.6]$	0.0548	5.48	2
$[-1.6, -1.2]$	0.0603	6.03	9
$[-1.2, -0.8]$	0.0968	9.68	6
$[-0.8, -0.4]$	0.1327	13.27	11
$[-0.4, 0.0]$	0.1554	15.54	19
$[0.0, 0.4]$	0.1554	15.54	25
$[0.4, 0.8]$	0.1327	13.27	17
$[0.8, 1.2]$	0.0968	9.68	2
$[1.2, 1.6]$	0.0603	6.03	6
$[-1.6, \infty]$	0.0548	5.48	3

The value for λ can then be calculated as follows:

$$\lambda = \sum_{i=1}^{m} \frac{(N_i - E_i)^2}{E_i} = 18.286$$

Since λ can be approximated as a χ^2 distribution with $m-k-1=10-0-1=9$ degrees of freedom, the critical point c at the 0.05 level of significance is calculated[6] to be 16.919 according to Eq. (3.167). Thus, we should reject the hypothesis H_0 because the calculated λ is greater than the critical point c.

The χ^2 goodness-of-fit test at the 0.05 significance level is in general used to determine when a hypothesized distribution is not an adequate distribution to use. To accept the diswtribution as a good fit, one needs to make sure the hypothesized distribution cannot be rejected at the 0.4 to 0.5 level-of-significance. The alternative is to use the χ^2 goodness-of-fit test for a number of potential distributions and select the one with smallest calculated χ^2.

When all the parameters are specified (instead of estimated), the Kolmogorov-Smirnov test [5] can also be used for the goodness-of-fit test. The Kolmogorov-Smirnov test in general is a more powerful test procedure when the sample size is relatively small.

3.3.4. Matched-Pairs Test

In this section, we discuss experiments in which two different methods (or systems) are to be compared to learn which one is better. To assure the two methods are evaluated under

[6] In general, we use a cumulative distribution function table to find the point with specific desired cumulative probability for complicated distributions, like the χ^2 distribution.

similar conditions, two closely resemble data samples or ideally the same data sample should be used to evaluate both methods. This type of hypotheses test is called *matched-paired* test [5].

3.3.4.1. The Sign Test

For $i = 1,\ldots,n$, let p_i denote the probability that method A is better than method B when testing on the i^{th} paired data sample. We shall assume that the probability p_i has the same value p for each of the n pairs. Suppose we wish to test the null hypothesis that method A is no better than method B. That is, the hypotheses to be tested have the following form:

$H_0 : p \leq 1/2$

$H_1 : p > 1/2$

Suppose that, for each pair of data samples, either one method or the other will appear to be better, and the two methods cannot tie. Under these assumptions, the n pairs represent n Bernoulli trials, for each of which there is probability p that method A yields better performance. Thus the number of pairs M in which method A yields better performance will have a binomial distribution $B(n, p)$. For the simple sign test where one needs to decide which method is better, p will be set to $1/2$. Hence a reasonable procedure is to reject H_0 if $M > c$, where c is a critical point. This procedure is called a signed test. The critical point can be found according to

$$P(M > c) = 1 - F_B(x = c) = \alpha_0 \tag{3.170}$$

where $F_B(x)$ is the distribution for binomial distribution. Thus, for observed values $M > c$, we will reject H_0.

3.3.4.2. Magnitude-Difference Test

The only information that the sign test utilizes from each pair of data samples, is the sign of the difference between two performances. To do a sign test, one does not need to obtain a numeric measurement of the magnitude of the difference between the two performances. However, if the measurement of magnitude of the difference for each pair is available, a test procedure based on the relative magnitudes of the differences can be used [11].

We assume now that the performance of each method can be measured for any data samples. For $i = 1,\ldots,n$, let A_i denote the performance of the method A on the i^{th} pair of data samples and B_i denote the performance of the method B on the i^{th} pair of data sample. Moreover, we shall let $D_i = A_i - B_i$. Since $D_1,\ldots,D_n$ are generated on n different pairs of data samples, they should be independent random variables. We also assume that $D_1,\ldots,D_n$ have the same distribution. Suppose now we are interested in testing the null hypothesis that method A and method B have on the average the same performance on the n pairs of data samples.

Let μ_D be the mean of D_i. The MLE estimate of μ_D is:

$$\mu_D = \sum_{i=1}^{n} \frac{D_i}{n} \tag{3.171}$$

The test hypotheses are:

$H_0 : \mu_D = 0$

$H_1 : \mu_D \neq 0$

The MLE estimate of the variance of D_i is

$$\sigma_D^2 = \frac{1}{n} \sum_{i=1}^{n} (D_i - \mu_D)^2 \tag{3.172}$$

We define a new random variable Z as follows:

$$Z = \frac{\mu_D}{\sigma_D / \sqrt{n}} \tag{3.173}$$

If n is large enough (> 50), Z is proved to have a standard Gaussian distribution $N(0,1)$. The normal test procedure described in Section 3.3.2 can be used to test H_0. This type of matched-paired tests usually depends on having enough pairs of data samples for the assumption that Z can be approximated with a Gaussian distribution. It also requires enough data samples to estimate the mean and variance of the D_i.

3.4. INFORMATION THEORY

Transmission of information is a general definition of what we call communication. Claude Shannon's classic paper of 1948 gave birth to a new field in information theory that has become the cornerstone for coding and digital communication. In the paper titled "A Mathematical Theory of Communication," he wrote [21]:

> *The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.*

Information theory is a mathematical framework for approaching a large class of problems related to encoding, transmission, and decoding information in a systematic and disciplined way. Since speech is a form of communication, information theory has served as the underlying mathematical foundation for spoken language processing.

3.4.1. Entropy

Three interpretations can be used to describe the quantity of *information*: (1) the amount of uncertainty before seeing an event, (2) the amount of surprise when seeing an event, and (3)

the amount of information after seeing an event. Although these three interpretations seem slightly different, they are virtually the same under the framework of information theory.

According to information theory, the information derivable from outcome x_i depends on its probability. If the probability $P(x_i)$ is small, we can derive a large degree of information, because the outcome that it has occurred is very rare. On the other hand, if the probability is large, the information derived will be small, because the outcome is well expected. Thus, the amount of information is defined as follows:

$$I(x_i) = \log \frac{1}{P(x_i)} \tag{3.174}$$

The reason to use a logarithm can be interpreted as follows. The information for two independent events to occur (where the joint probability is the multiplication of both individual probabilities) can be simply carried out by the addition of the individual information of each event. When the logarithm base is 2, the unit of information is called a *bit*. This means that one bit of information is required to specify the outcome. In this probabilistic framework, the amount of information represents uncertainty. Suppose X is a discrete random variable taking value x_i (referred to as a symbol) from a finite or countable infinite sample space $S = \{x_1, x_2, \ldots, x_i, \ldots\}$ (referred to as an alphabet). The symbol x_i is produced from an information source with alphabet S, according to the probability distribution of the random variable X. One of the most important properties of an information source is the entropy $H(S)$ of the random variable X, defined as the average amount of information (expected information):

$$H(X) = E[I(X)] = \sum_S P(x_i) I(x_i) = \sum_S P(x_i) \log \frac{1}{P(x_i)} = E\left[-\log P(X)\right] \tag{3.175}$$

This entropy $H(X)$ is the amount of information required to specify what kind of symbol has occurred on average. It is also the average uncertainty for the symbol. Suppose that the sample space S has an alphabet size $\|S\| = N$. The entropy $H(X)$ attains the maximum value when the pf has a uniform distribution, i.e.:

$$P(x_i) = P(x_j) = \frac{1}{N} \quad \text{for all } i \text{ and } j \tag{3.176}$$

Equation (3.176) can be interpreted to mean that *uncertainty* reaches its maximum level when no outcome is more probable than any other. It can be proved that the entropy $H(X)$ is nonnegative and becomes zero only if the probability function is a deterministic one, i.e.,

$$H(X) \geq 0 \text{ with equality i.f.f. } P(x_i) = 1 \text{ for some } x_i \in S \tag{3.177}$$

There is another very interesting property for the entropy. If we replace the pf of generating symbol x_i in Eq. (3.175) with any other arbitrary pf, the new value is no smaller than the original entropy. That is,

$$H(X) \le E[-\log Q(X)] = -\sum_S P(x_i)\log Q(x_i) \tag{3.178}$$

Equation (3.178) has a very important meaning. It shows that we are more uncertain about the data if we misestimate the distribution governing the data source. The equality for Eq. (3.178) occurs if and only if $P(x_i) = Q(x_i) \quad 1 \le i \le N$. Equation (3.178), often referred to as *Jensen's inequality*, is the basis for the proof of EM algorithm in Chapter 4. Similarly, Jensen's inequality can be extended to continuous pdf:

$$-\int f_x(x)\log f_x(x)dx \le -\int g_x(x)\log f_x(x)dx \tag{3.179}$$

with equality occurring if and only if $f_x(x) = g_x(x) \ \forall x$.

The proof of Eq. (3.178) follows from the fact $\log(x) \le x-1, \ \forall x$, so the following quantity must have an non-positive value.

$$\sum_S P(x_i)\log\frac{Q(x_i)}{P(x_i)} \le \sum_S P(x_i)\left[\frac{Q(x_i)}{P(x_i)} - 1\right] = 0 \tag{3.180}$$

Based on this property, the negation of the quantity in Eq. (3.180) can be used for the measurement of the distance of two probability distributions. Specifically, the *Kullback-Leibler* (KL) *distance* (*relative entropy*, *discrimination*, or *divergence*) is defined as:

$$KL(P \| Q) = E\left[\log\frac{P(X)}{Q(X)}\right] = \sum_S P(x_i)\log\frac{P(x_i)}{Q(x_i)} \tag{3.181}$$

As discussed in Chapter 11, the branching factor of a grammar or language is an important measure of degree of difficulty of a particular task in spoken language systems. This relates to the size of the word list from which a speech recognizer or a natural language processor needs to disambiguate in a given context. According to the entropy definition above, this branching factor estimate (or average choices for an alphabet) is defined as follows:

$$PP(X) = 2^{H(X)} \tag{3.182}$$

$PP(X)$ is called the *perplexity* of source X, since it describes how confusing the grammar (or language) is. The value of perplexity is equivalent to the size of an imaginary equivalent list, whose words are equally probable. The bigger the perplexity, the higher branching factor. To find out the perplexity of English, Shannon devised an ingenious way [22] to estimate the entropy and perplexity of English words and letters. His method is similar to a guessing game where a human subject guesses sequentially the words of a text hidden from him, using the relative frequencies of her/his guesses as the estimates of the probability distribution underlying the source of the text. Shannon's perplexity estimate of English comes out to be about 2.39 for English letters and 130 for English words. Chapter 11 has a detailed description on the use of perplexity for language modeling.

3.4.2. Conditional Entropy

Now let us consider transmission of symbols through an information channel. Suppose the input alphabet is $X=(x_1,x_2,\ldots,x_s)$, the output alphabet is $Y=(y_1,y_2,\ldots,y_t)$, and the information channel is defined by the channel matrix $M_{ij}=P(y_j\mid x_i)$, where $P(y_j\mid x_i)$ is the conditional probability of receiving output symbol y_j when input symbol x_i is sent. Figure 3.17 shows an example of an information channel.

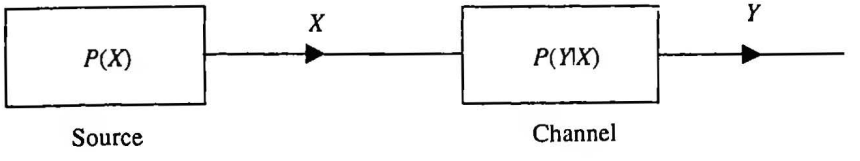

Figure 3.17 Example of information channel. The source is described by source pf $P(X)$ and the channel is characterized by the conditional pf $P(Y|X)$.

Before transmission, the average amount of information, or the uncertainty of the input alphabet X, is the prior entropy $H(X)$.

$$H(X)=\sum_X P(X=x_i)\log\frac{1}{P(X=x_i)} \tag{3.183}$$

where $P(x_i)$ is the prior probability. After transmission, suppose y_j is received; then the average amount of information, or the uncertainty of the input alphabet A, is reduced to the following *posterior* entropy:

$$H(X\mid Y=y_j)=-\sum_X P(X=x_i\mid Y=y_j)\log P(X=x_i\mid Y=y_j) \tag{3.184}$$

where the $P(x_i\mid y_j)$ are the posterior probabilities. Averaging the posterior entropy $H(X\mid y_j)$ over all output symbols y_j leads to the following equation:

$$\begin{aligned} H(X\mid Y)&=\sum_Y P(Y=y_j)H(X\mid Y=y_j)\\ &=-\sum_Y P(Y=y_j)\sum_X P(X=x_i\mid Y=y_j)\log P(X=x_i\mid Y=y_j)\\ &=-\sum_X\sum_Y P(X=x_i,Y=y_j)\log P(X=x_i\mid Y=y_j) \end{aligned} \tag{3.185}$$

This *conditional entropy*, defined in Eq. (3.185), is the average amount of information or the uncertainty of the input alphabet X given the outcome of the output event Y. Based on the definition of conditional entropy, we derive the following equation:

$$\begin{aligned} H(X,Y) &= -\sum_X \sum_Y P(X=x_i, Y=y_i) \log P(X=x_i, Y=y_i) \\ &= -\sum_X \sum_Y P(X=x_i, Y=y_i) \{\log P(X=x_i) + \log P(Y=y_i \mid X=x_i)\} \\ &= H(X) + H(Y \mid X) \end{aligned} \tag{3.186}$$

Equation (3.186) has an intuitive meaning – the uncertainty about two random variables equals the sum of uncertainty about the first variable and the conditional entropy for the second variable given the first variable is known. Equations (3.185) and (3.186) can be generalized to random vectors **X** and **Y** where each contains several random variables.

It can be proved that the chain rule [Eq. (3.16)] applies to entropy.

$$H(X_1, \cdots, X_n) = H(X_n \mid X_1, \cdots, X_{n-1}) + \cdots + H(X_2 \mid X_1) + H(X_1) \tag{3.187}$$

Finally, the following inequality can also be proved:

$$H(X \mid Y, Z) \le H(X \mid Y) \tag{3.188}$$

with equality i.f.f. X and Z being independent when conditioned on Y. Equation (3.188) basically confirms the intuitive belief that uncertainty decreases when more information is known.

3.4.3. The Source Coding Theorem

Information theory is the foundation for data compressing. In this section we describe *Shannon's source coding theorem*, also known as the *first coding theorem*. In source coding, we are interested in *lossless* compression, which means the compressed information (or symbols) can be recovered (decoded) perfectly. The entropy serves as the upper bound for a source lossless compression.

Consider an information source with alphabet $S = \{0, 1, \ldots, N-1\}$. The goal of data compression is to *encode* the output symbols into a string of binary symbols. An interesting question arises: *What is the minimum number of bits required, on the average, to encode the output symbols of the information source?*

Let's assume we have a source that can emit four symbols {0,1,2,3} with equal probability $P(0) = P(1) = P(2) = P(3) = 1/4$. Its entropy is 2 bits as illustrated in Eq. (3.189):

$$H(S) = \sum_{i=0}^{3} P(i) \log_2 \frac{1}{P(i)} = 2 \tag{3.189}$$

It is obvious that 2 bits per symbol is good enough to encode this source. A possible binary code for this source is {00, 01, 10, 11}. It could happen, though some symbols are more likely than others, for example, $P(0) = 1/2$, $P(1) = 1/4$, $P(2) = 1/8$, $P(3) = 1/8$. In this case the entropy is only 1.75 bits. One obvious idea is to use fewer bits for lower values that are frequently used and more bits for larger values that are rarely used. To represent this

source we can use a variable-length code {0,10,110,111}, where no codeword is a prefix for the rest and thus a string of 0s and 1s can be uniquely broken into those symbols. The encoding scheme with such a property is called *uniquely decipherable* (or instantaneous) coding, because as soon as the decoder observes a sequence of codes, it can decisively determine the sequence of the original symbols. If we let $r(x)$ be the number of bits (length) used to encode symbol x, the average rate R of bits per symbol used for encoding the information source is:

$$R = \sum_x r(x)P(x) \tag{3.190}$$

In our case, R is 1.75 bits as shown in Eq. (3.191):

$$R = 0.5\times1 + 0.25\times2 + 0.125\times3 + 0.125\times3 = 1.75 \tag{3.191}$$

Such variable-length coding strategy is called *Huffman coding*. Huffman coding belongs to *entropy coding* because it matches the entropy of the source. In general, *Shannon's source coding* theorem says that a source cannot be coded with fewer bits than its entropy. We will skip the proof here. Interested readers can refer to [3, 15, 17] for the detailed proof. This theorem is consistent with our intuition because the entropy measure is exactly the information content of the information measured in bits. If the entropy increases, then uncertainty increases, resulting in a large amount of information. Therefore, it takes more bits to encode the symbols. In the case above, we are able to match this rate, but, in general, this is impossible, though we can get arbitrarily close to it. The Huffman code for this source offers a compression rate of 12.5% relative to the code designed for the uniform distribution.

Shannon's source coding theorem establishes not only the lower bound for lossless compression but also the upper bound. Let $\lceil x \rceil$ denote the smallest integer that greater or equal to x. As in the similar procedure above, we can make the code length assigned to source output x equal to

$$l(x) = \lceil -\log P(x) \rceil \tag{3.192}$$

The average length L satisfies the following inequality:

$$L = \sum_x l(x)P(x) < \sum_x [1 - \log P(x)]P(x) = 1 + H(X) \tag{3.193}$$

Equation (3.193) means that the average rate R only exceeds the value of entropy by less than one bit.

L can be made arbitrarily close to the entropy by block coding. Instead of encoding single output symbols of the information source, one can encode each block of length n. Let's assume the source is memoryless, so $X_1, X_2, \ldots, X_n$ are independent. According to Eq. (3.193), the average rate R for this block code satisfies:

$$L < 1 + H(X_1, X_2, \ldots, X_n) = 1 + nH(X) \tag{3.194}$$

This makes the average number of bits per output symbol, L/n, satisfy

$$\lim_{n\to\infty}\frac{1}{n}L \le H(X) \tag{3.195}$$

In general, Huffman coding arranges the symbols in order of decreasing probability, assigns the bit 0 to the symbol of highest probability and the bit 1 to what is left, and proceeds the same way for the second highest probability value (which now has a code 10) and iterate. This results in 2.25 bits for the uniform distribution case, which is higher than the 2 bits we obtain with equal-length codes.

Lempel-Ziv coding is a coding strategy that uses correlation to encode *strings* of symbols that occur frequently. Although it can be proved to converge to the entropy, its convergence rate is much slower [27]. Unlike Huffman coding, Lempel-Ziv coding is independent of the distribution of the source; i.e., it needs not be aware of the distribution of the source before encoding. This type of coding scheme is often referred to as *universal* encoding scheme.

3.4.4. Mutual Information and Channel Coding

Let's review the information channel illustrated in Figure 3.17. An intuitively plausible measure of the average amount of information provided by the random event Y about the random event X is the average difference between the number of bits it takes to specify the outcome of X when the outcome of Y is not known and the outcome of Y is known. *Mutual information* is defined as the difference in the entropy of X and the conditional entropy of X given Y:

$$\begin{aligned} I(X;Y) &= H(X) - H(X \mid Y) \\ &= \sum_X P(x_i)\log\frac{1}{P(x_i)} - \sum_X\sum_Y P(x_i,y_j)\log\frac{1}{P(x_i \mid y_j)} \\ &= \sum_X\sum_Y P(x_i,y_j)\log\frac{P(x_i \mid y_j)}{P(x_i)} = \sum_X\sum_Y P(x_i,y_j)\log\frac{P(x_i,y_j)}{P(x_i)P(y_i)} \\ &= E\left[\log\frac{P(X,Y)}{P(X)P(Y)}\right] \end{aligned} \tag{3.196}$$

$I(X;Y)$ is referred to as the mutual information between X and Y. $I(X;Y)$ is symmetrical; i.e., $I(X;Y) = I(Y;X)$. The quantity $P(x,y)/P(x)P(y)$ is often referred to as the *mutual information between symbol x and y*. $I(X;Y)$ is bounded:

$$0 \le I(X;Y) \le \min\left[H(X), H(Y)\right] \tag{3.197}$$

$I(X;Y)$ reaches the minimum value (zero) when the random variables X and Y are independent.

Mutual information represents the information obtained (or the reduction in uncertainty) through a channel by observing an output symbol. If the information channel is noiseless, the input symbol can be determined definitely by observing an output symbol. In this case, the conditional entropy $H(X|Y)$ equals zero and it is called a *noiseless channel*. We obtain the maximum mutual information $I(X; Y) = H(X)$. However, the information channel is generally *noisy* so that the conditional entropy $H(X|Y)$ is not zero. Therefore, maximizing the mutual information is equivalent to obtaining a low-noise information channel, which offers a closer relationship between input and output symbols.

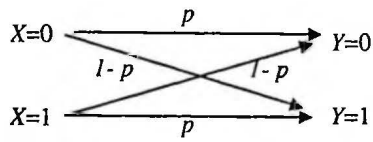

Figure 3.18 A binary channel with two symbols.

Let's assume that we have a binary channel, a channel with a binary input and output as shown in Figure 3.18. Associated with each output are a probability p that the output is correct, and a probability $(1-p)$ that it is not, so that the channel is *symmetric*.

If we observe a symbol $Y = 1$ at the output, we don't know for sure what symbol X was transmitted, though we know $P(X=1|Y=1)=p$ and $P(X=0|Y=1)=(1-p)$, so that we can measure our uncertainty about X by its conditional entropy:

$$H(X|Y=1)=-p\log p-(1-p)\log(1-p) \tag{3.198}$$

If we assume that our source X has a uniform distribution, $H(X|Y)=H(X|Y=1)$ as shown in Eq. (3.198) and $H(X) = 1$. The mutual information between X and Y is given by

$$I(X,Y)=H(X)-H(X|Y)=1+p\log p+(1-p)\log(1-p) \tag{3.199}$$

It measures the information that Y carries by about X. The channel capacity C is the maximum of the mutual information over all distributions of X. That is,

$$C=\max_{P(x)} I(X;Y) \tag{3.200}$$

The channel capacity C can be attained by varying the distribution of the information source until the mutual information is maximized for the channel. The channel capacity C can be regarded as a channel that can transmit at most C bits of information per unit of time. *Shannon's channel coding* theorem says that for a given channel there exists a code that permits error-free transmission across the channel, provided that $R \le C$, where R is the rate of the communication system, which is defined as the number of bits per unit of time being transmitted by the communication system. Shannon's channel coding theorem states the fact that *arbitrarily reliable communication is possible at any rate below channel capacity*.

Figure 3.19 illustrates a transmission channel with the source encoder and destination decoder. The source encoder will encode the source symbol sequence $\mathbf{x}=x_1,x_2,\ldots,x_n$ into

channel input sequence $y_1, y_2, \ldots, y_k$. The destination decoder takes the output sequence $z_1, z_2, \ldots, z_k$ from the channel and converts it into the estimates of the source output $\overline{\mathbf{x}} = \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n$. The goal of this transmission is to make the probability of correct decoding $P(\overline{\mathbf{x}} = \mathbf{x})$ asymptotically close to 1 while keeping the compression ratio $\Re = n/k$ as large as possible. *Shannon's source-channel coding* theorem (also referred to as *Shannon's second coding* theorem) says that it is possible to find an encoder-decoder pair of rate $\Re$ for a noisy information channel, provided that $\Re \times H(X) \leq C$.

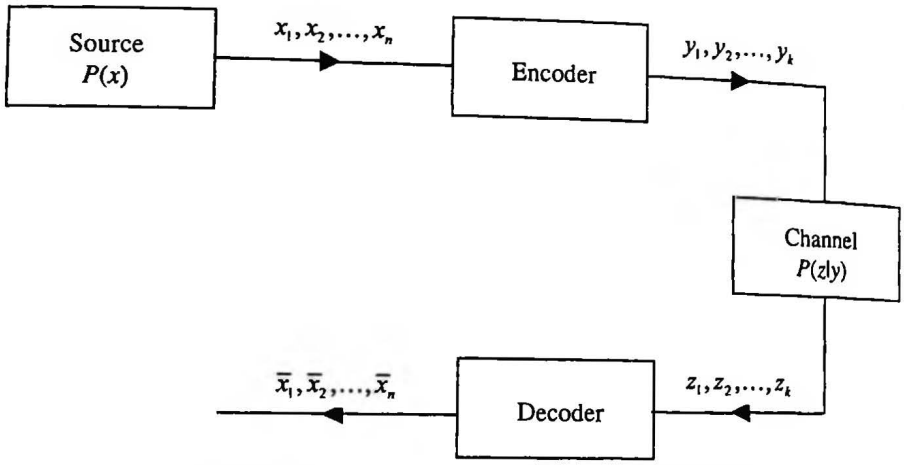

Figure 3.19 Transmission of information through a noisy channel [15].

Because of channel errors, speech coders need to provide error correction codes that will decrease the bit rate allocated to the speech. In practice, there is a tradeoff between the bit rate used for source coding and the bit rate for channel coding. In Chapter 7 we will describe speech coding in great detail.

3.5. HISTORICAL PERSPECTIVE AND FURTHER READING

The idea of uncertainty and probability can be traced all the way back to about 3500 B.C., when games of chance played with bone objects were developed in Egypt. Cubical dice with markings virtually identical to modern dice have been found in Egyptian tombs dating around 2000 B.C. Gambling with dice played an important part in the early development of probability theory. Modern mathematical theory of probability is believed to have been started by the French mathematicians Blaise Pascal (1623-1662) and Pierre Fermat (1601-1665) when they worked on certain gambling problems involving dice. English mathematician Thomas Bayes (1702-1761) was first to use probability inductively and established a mathematical basis for probability inference, leading to what is now known as Bayes' theorem. The theory of probability has developed steadily since then and has been widely ap-

plied in diverse fields of study. There are many good textbooks on probability theory. The book by DeGroot [6] is an excellent textbook for both probability and statistics which covers all the necessary elements for engineering majors. The authors also recommend [14], [19], or [24] for interested readers.

Estimation theory is a basic subject in statistics covered in textbooks. The books by DeGroot [6], Wilks [26] and Hoel [13] offer excellent discussions of estimation theory. They all include comprehensive treatments for maximum likelihood estimation and Bayesian estimation. Maximum likelihood estimation was introduced in 1912 by R. A. Fisher (1890-1962) and has been applied to various domains. It is arguably the most popular parameter estimation method due to its intuitive appeal and excellent performance with large training samples. The EM algorithm in Chapter 4 and the estimation of hidden Markov models in Chapter 8 are based on the principle of MLE. The use of prior distribution in Bayesian estimation is very controversial in statistics. Some statisticians adhere to the Bayesian philosophy of statistics by taking the Bayesian estimation view of the parameter Φ having a probability distribution. Others, however, believe that in many problems Φ is not a random variable but rather a fixed number whose value is unknown. Those statisticians believe that a prior distribution can be assigned to a parameter Φ only when there is extensive prior knowledge of the past; thus the non-informative priors are completely ruled out. Both groups of statisticians agree that whenever a meaningful prior distribution can be obtained, the theory of Bayesian estimation is applicable and useful. The books by DeGroot [6] and Poor [20] are excellent for learning the basics of Bayesian and MAP estimations. Bayesian and MAP adaptation are particularly powerful when the training samples are sparse. Therefore, they are often used for adaptation where the knowledge of prior distribution can help to adapt the model to a new but limited training set. The speaker adaptation work done by Brown et al. [2] first applied Bayesian estimation to speech recognition and [9] is another good paper on using MAP for hidden Markov models. References [4], [16] and [14] have extensive studies of different conjugate prior distributions for various standard distributions. Finally, [1] has an extensive reference for Bayesian estimation.

Significance testing is an essential tool for statisticians to interpret all the statistical experiments. Neyman and Pearson provided some of the most important pioneering work in hypotheses testing [18]. There are many different testing methods presented in most statistics books. The χ^2 test, invented in 1900 by Karl Pearson, is arguably the most widely used testing method. Again, the textbook by DeGroot [6] is an excellent source for the basics of testing and various testing methods. The authors recommend [7] as an interesting book that uses many real-world examples to explain statistical theories and methods, particularly the significance testing.

Information theory first appeared in Claude Shannon's historical paper: *A Mathematical Theory of Communication* [21]. In it, Shannon, analyzed communication as the transmission of a message from a source through a channel to a receiver. In order to solve the problem he created a new branch of applied mathematics – *information and coding theory*. IEEE published a collection of Shannon's papers [23] containing all of his published works, as well as many that have never been published. Those published include his classic papers on information theory and switching theory. Among the unpublished works are his once-

secret wartime reports, his Ph.D. thesis on population genetics, unpublished Bell Labs memoranda, and a paper on the theory of juggling. The textbook by McEliece [17] is excellent for learning all theoretical aspects of information and coding theory. However, it might be out of print now. Instead, the books by Hamming [12] and Cover [3] are two current great references for information and coding theory. Finally, F. Jelinek's *Statistical Methods for Speech Recognition* [15] approaches the speech recognition problem from an information-theoretic aspect. It is a useful book for people interested in both topics.

REFERENCES

[1] Bernardo, J.M. and A.F.M. Smith, *Bayesian Theory*, 1996, New York, John Wiley.

[2] Brown, P., C.-H. Lee, and J. Spohrer, "Bayesian Adaptation in Speech Recognition," *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1983, Boston, MA, pp. 761-764.

[3] Cover, T.M. and J.A. Thomas, *Elements of Information Theory*, 1991, New York, John Wiley and Sons.

[4] DeGroot, M.H., *Optimal Statistical Decisions*, 1970, New York, NY, McGraw-Hill.

[5] DeGroot, M.H., *Probability and Statistics*, Addison-Wesley Series in Behavioral Science: Quantitive Methods, F. Mosteller, ed., 1975, Reading, MA, Addison-Wesley Publishing Company.

[6] DeGroot, M.H., *Probability and Statistics*, 2nd ed, Addison-Wesley Series in Behavioral Science: Quantitive Methods, F. Mosteller, ed., 1986, Reading, MA, Addison-Wesley Publishing Company.

[7] Freedman, D., *et al.*, *Statistics*, 2nd ed, 1991, New York, W. W. Norton & Company, Inc.

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

[9] Gauvain, J.L. and C.H. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains," *IEEE Trans. on Speech and Audio Processing*, 1994, **2**(2), pp. 291-298.

[10] Gillett, G.E., *Introduction to Operations Research: A Computer-Oriented Algorithmic Approach*, McGraw-Hill Series in Industrial Engineering and Management Science, J. Riggs, ed., 1976, New York, McGraw-Hill.

[11] Gillick, L. and S.J. Cox, "Some Statistical Issues in the Comparison of Speech Recognition Algorithms," *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1989, Glasgow, Scotland, UK, IEEE, pp. 532-535.

[12] Hamming, R.W., *Coding and Information Theory*, 1986, Englewood Cliffs, NJ, Prentice-Hall.

[13] Hoel, P.G., *Introduction to Mathematical Statistics*, 5th ed., 1984, John Wiley & Sons.

[14] Jeffreys, H., *Theory of Probability*, 1961, Oxford University Press.

[15] Jelinek, F., *Statistical Methods for Speech Recognition*, Language, Speech, and Communication, 1998, Cambridge, MA, MIT Press.

[16] Lindley, D.V., "The Use of Prior Probability Distributions in Statistical Inference and Decision," *Fourth Berkeley Symposium on Mathematical Statistics and Probability*, 1961, Berkeley, CA, Univ. of California Press.

[17] McEliece, R., *The Theory of Information and Coding*, Encyclopedia of Mathematics and Its Applications, eds. R. Gian-Carlo. Vol. 3, 1977, Reading, MA, Addison-Wesley Publishing Company.

[18] Neyman, J. and E.S. Pearson, "On the Problem of the Most Efficient Tests of Statistical Hypotheses," *Philosophical Trans. of Royal Society*, 1928, **231**, pp. 289-337.

[19] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, 3rd ed., 1991, New York, McGraw-Hill.

[20] Poor, H.V., *An Introduction to Signal Detection and Estimation*, Springer Tests in Electrical Engineering, J.B. Thomas, ed., 1988, New York, Springer-Verlag.

[21] Shannon, C., "A Mathematical Theory of Communication," *Bell System Technical Journal*, 1948, **27**, pp. 379-423, 623-526.

[22] Shannon, C.E., "Prediction and Entropy of Printed English," *Bell System Technical Journal*, 1951, pp. 50-62.

[23] Shannon, C.E., *Claude Elwood Shannon: Collected Papers*, 1993, IEEE.

[24] Viniotis, Y., *Probability and Random Processes for Electrical Engineering*, Outline Series in Electronics & Electrical Engineering, Schaum, ed., 1998, New York, WCB McGraw-Hill.

[25] Wald, A., "Note of Consistency of Maximum Likelihood Estimate," *Ann. Mathematical Statistics*, 1949(20), pp. 595-601.

[26] Wilks, S.S., *Mathematical Statistics*, 1962, New York, John Wiley and Sons.

[27] Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory*, 1997, **IT-23**, pp. 337-343.

CHAPTER 4

Pattern Recognition

Spoken language processing relies heavily on pattern recognition, one of the most challenging problems for machines. In a broader sense, the ability to recognize patterns forms the core of our intelligence. If we can incorporate the ability to reliably recognize patterns in our work and life, we can make machines much easier to use. The process of human pattern recognition is not well understood.

Due to the inherent variability of spoken language patterns, we emphasize the use of statistical approaches in this book. The decision for pattern recognition is based on appropriate probabilistic models of the patterns. This chapter presents several mathematical fundamentals for statistical pattern recognition and classification. In particular, Bayes' decision theory and estimation techniques for parameters of classifiers are introduced. Bayes' decision theory, which plays a central role for statistical pattern recognition, introduces the concept of decision-making based on both *posterior* knowledge obtained from specific observation data, and *prior* knowledge of the categories. To build such a classifier or predictor, it is critical to estimate *prior* class probabilities and the class-conditional probabilities for a Bayes' classifier.

Supervised learning has class information for the data. Only the probabilistic structure needs to be learned. Maximum likelihood estimation (MLE) and maximum posterior probability estimation (MAP) that we discussed in Chapter 3 are two most powerful methods. Both MLE and MAP aim to maximize the likelihood function. The MLE criterion does not necessarily minimize the recognition error rate. Various discriminant estimation methods are introduced for that purpose. *Maximum mutual information estimation* (MMIE) is based on criteria to achieve maximum model separation (the model for the correct class is well separated from other competing models) instead of likelihood criteria. The MMIE criterion is one step closer but still is not directly related to minimizing the error rate. Other discriminant estimation methods, such as *minimum error-rate estimation*, use the ultimate goal of pattern recognition – minimizing the classification errors. *Neural networks* are one class of discriminant estimation methods.

The EM algorithm is an iterative algorithm for unsupervised learning in which class information is unavailable or only partially available. The EM algorithm forms the theoretical basis for training hidden Markov models (HMM) as described in Chapter 8. To better understand the relationship between MLE and EM algorithms, we first introduce *vector quantization* (VQ), a widely used source-coding technique in speech analysis. The well-known *k-means* clustering algorithm best illustrates the relationship between MLE and the EM algorithm. We close this chapter by introducing a powerful binary prediction and regression technique, classification and regression trees (CART). CART represents an important technique that combines rule-based expert knowledge and statistical learning.

4.1. BAYES' DECISION THEORY

Bayes' decision theory forms the basis of statistical pattern recognition. The theory is based on the assumption that the decision problem can be specified in probabilistic terms and that all of the relevant probability values are known. Bayes' decision theory can be viewed as a formalization of a common-sense procedure, i.e., the aim to achieve minimum-error-rate classification. This common-sense procedure can be best observed in the following real-world decision examples.

Consider the problem of making predictions for the stock market. We use the Dow Jones Industrial average index to formulate our example, where we have to decide tomorrow's Dow Jones Industrial average index in one of the three categories (events): *Up*, *Down*, or *Unchanged*. The available information is the probability function $P(\omega)$ of the three categories. The variable ω is a discrete random variable with the value $\omega = \omega_i$ $(i = 1,2,3)$. We call the probability $P(\omega_i)$ a *prior* probability, since it reflects prior knowledge of tomorrow's Dow Jones Industrial index. If we have to make a decision based only on the *prior* probability, the most plausible decision may be made by selecting the class ω_i with the highest prior probability $P(\omega_i)$. This decision is unreasonable, in that we always make the same decision even though we know that all three categories of Dow Jones Industrial index changes will possibly appear. If we are given further observable data, such as the federal funds interest rate or the jobless rate, we can make a more informed decision. Let x be a con-

tinuous random variable whose value is the federal-fund interest rate, and $f_{x|\omega}(x|\omega)$ be a class-conditional pdf For simplicity, we denote the pdf $f_{x|\omega}(x|\omega)$ as $p(x|\omega_i)$, where $i = 1$, 2, 3 unless there is ambiguity. The class-conditional probability density function is often referred to as the *likelihood* function as well, since it measures how likely it is that the underlying parametric model of class ω_i will generate the data sample x. Since we know the prior probability $P(\omega_i)$ and class-conditional pdf $p(x|\omega_i)$, we can compute the conditional probability $P(\omega_i|x)$ using Bayes' rule:

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} \tag{4.1}$$

where $p(x) = \sum_{i=1}^{3} p(x|\omega_i)P(\omega_i)$.

The probability term in the left-hand side of Eq. (4.1) is called the *posterior* probability as it is the probability of class ω_i after observing the federal-funds interest rate x. An intuitive decision rule would be choosing the class ω_k with the greatest posterior probability. That is,

$$k = \arg\max_i P(\omega_i|x) \tag{4.2}$$

In general, the denominator $p(x)$ in Eq. (4.1) is unnecessary because it is a constant term for all classes. Therefore, Eq. (4.2) becomes

$$k = \arg\max_i P(\omega_i|x) = \arg\max_i p(x|\omega_i)P(\omega_i) \tag{4.3}$$

The rule in Eq. (4.3) is referred to as Bayes' decision rule. It shows how the observed data x changes the decision based on the *prior* probability $P(\omega_i)$ to one based on the posterior probability $P(\omega_i|x)$. Decision making based on the *posterior* probability is more reliable, because it employs prior knowledge together with the present observed data. As a matter of fact, when the prior knowledge is non-informative ($P(\omega_1) = P(\omega_2) = P(\omega_3) = 1/3$), the observed data fully control the decision. On the other hand, when observed data are ambiguous, then prior knowledge controls the decision. There are many kinds of decision rules based on posterior probability. Our interest is to find the decision rule that leads to minimum overall risk, or minimum error rate in decision.

4.1.1. Minimum-Error-Rate Decision Rules

Bayes' decision rule is designed to minimize the overall risk involved in making a decision. Bayes' decision based on posterior probability $P(\omega_i|x)$ instead of prior probability $P(\omega_i)$ is a natural choice. Given an observation x, if $P(\omega_k|x) \geq P(\omega_i|x)$ for all $i \neq k$, we can decide that the true class is ω_k. To justify this procedure, we show such a decision results in minimum decision error.

Let $\Omega=\{\omega_1,...,\omega_s\}$ be the finite set of s possible categories to be predicted and $\Delta=\{\delta_1,...,\delta_t\}$ be a finite set of t possible decisions. Let $l(\delta_i \mid \omega_j)$ be the loss function incurred for making decision δ_i when the true class is ω_j. Using the *prior* probability $P(\omega_j)$ and class-conditional pdf $p(x \mid \omega_j)$, the posterior probability $P(\omega_i \mid x)$ is computed by Bayes' rule as shown in Eq. (4.1). Since the posterior probability $P(\omega_j \mid x)$ is the probability that the true class is ω_j after observing the data x, the expected loss associated with making decision δ_i is:

$$R(\delta_i \mid x)=\sum_{j=1}^{s} l(\delta_i \mid \omega_j)P(\omega_j \mid x) \tag{4.4}$$

In decision-theoretic terminology, the above expression is called *conditional risks*. The overall risk R is the expected loss associated with a given decision rule. The decision rule is employed as a decision function $\delta(x)$ that maps the data x to one of the decisions $\Delta=\{\delta_1,...,\delta_t\}$. Since $R(\delta_i \mid x)$ is the conditional risk associated with decision δ_i, the overall risk is given by:

$$R=\int_{-\infty}^{\infty} R(\delta(x) \mid x)p(x)dx \tag{4.5}$$

If the decision function $\delta(x)$ is chosen so that the conditional risk $R(\delta(x) \mid x)$ is minimized for every x, the overall risk is minimized. This leads to the Bayes' decision rule: To minimize the overall risk, we compute the conditional risk shown in Eq. (4.4) for $i=1,...,t$ and select the decision δ_i for which the conditional risk $R(\delta_i \mid x)$ is minimum. The resulting minimum overall risk is known as *Bayes' risk* that has the best performance possible.

The loss function $l(\delta_i \mid \omega_j)$ in the Bayes' decision rule can be defined as:

$$l(\delta_i \mid \omega_j)=\begin{cases} 0 & i=j \\ & i,\ j=1,\ldots,s \\ 1 & i\neq j \end{cases} \tag{4.6}$$

This loss function assigns no loss to a correct decision where the true class is ω_i and the decision is δ_i, which implies that the true class must be ω_i. It assigns a unit loss to any error where $i\neq j$; i.e., all errors are equally costly. This type of loss function is known as a *symmetrical* or *zero-one* loss function. The risk corresponding to this loss function equals the classification error rate, as shown in the following equation.

$$\begin{aligned} R(\delta_i \mid x)&=\sum_{j=1}^{s} l(\delta_i \mid \omega_j)P(\omega_j \mid x)=\sum_{j\neq i} P(\omega_j \mid x) \\ &=\sum_{j=1}^{s} P(\omega_j \mid x)-P(\omega_i \mid x)=1-P(\omega_i \mid x) \end{aligned} \tag{4.7}$$

Here $P(\omega_i \mid x)$ is the conditional probability that decision δ_i is correct after observing the data x. Therefore, in order to minimize classification error rate, we have to choose the decision of class i that maximizes the posterior probability $P(\omega_i \mid x)$. Furthermore, since $p(x)$ is a constant, the decision is equivalent to picking the class i that maximizes $p(x \mid \omega_i)P(\omega_i)$. The Bayes' decision rule can be formulated as follows:

$$\delta(x) = \arg\max_i P(\omega_i \mid x) = \arg\max_i P(x \mid \omega_i)P(\omega_i) \tag{4.8}$$

This decision rule, which is based on the maximum of the posterior probability $P(\omega_i \mid x)$, is called the *minimum-error-rate decision rule*. It minimizes the classification error rate. Although our description is for random variable x, Bayes' decision rule is applicable to multivariate random vector $\mathbf{x}$ without loss of generality.

A pattern classifier can be regarded as a device for partitioning the feature space into decision regions. Without loss of generality, we consider a two-class case. Assume that the classifier divides the space $\Re$ into two regions, $\Re_1$ and $\Re_2$. To compute the likelihood of errors, we need to consider two cases. In the first case, x falls in $\Re_1$, but the true class is ω_2. In the other case, x falls in $\Re_2$, but the true class is ω_1. Since these two cases are mutually exclusive, we have

$$\begin{aligned} P(error) &= P(x \in \Re_1, \omega_2) + P(x \in \Re_2, \omega_1) \\ &= P(x \in \Re_1 \mid \omega_2)P(\omega_2) + P(x \in \Re_2 \mid \omega_1)P(\omega_1) \\ &= \int_{\Re_1} P(x \mid \omega_2)P(\omega_2)dx + \int_{\Re_2} P(x \mid \omega_1)P(\omega_1)dx \end{aligned} \tag{4.9}$$

Figure 4.1 illustrates the calculation of the classification error in Eq. (4.9). The two terms in the summation are merely the tail areas of the function $P(x \mid \omega_i)P(\omega_i)$. It is clear

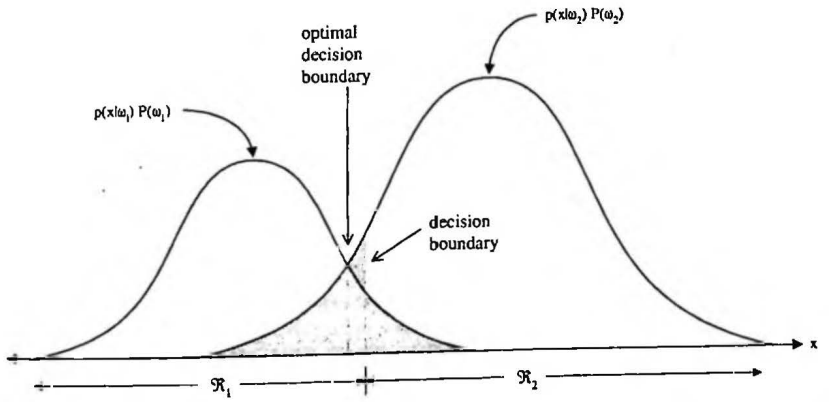

Figure 4.1 Calculation of the likelihood of classification error [22]. The shaded area represents the integral value in Eq. (4.9).

that this decision boundary is not optimal. If we move the decision boundary a little bit to the left, so that the decision is made to choose the class i based on the maximum value of $P(x \mid \omega_i)P(\omega_i)$, the tail integral area $P(error)$ becomes minimum, which is the Bayes' decision rule.

4.1.2. Discriminant Functions

The decision problem above can also be viewed as a pattern classification problem where unknown data $\mathbf{x}$[1] are classified into known categories, such as the classification of sounds into phonemes using spectral data $\mathbf{x}$. A classifier is designed to classify data $\mathbf{x}$ into s categories by using s discriminant functions, $d_i(\mathbf{x})$, computing the similarities between the unknown data $\mathbf{x}$ and each class ω_i and assigning $\mathbf{x}$ to class ω_j if

$$d_j(\mathbf{x}) > d_i(\mathbf{x}) \quad \forall i \neq j \tag{4.10}$$

This representation of a classifier is illustrated in Figure 4.2.

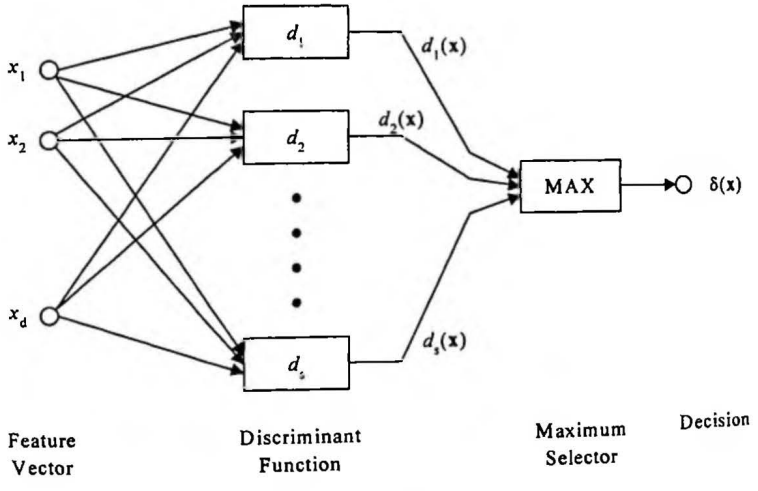

Figure 4.2 Block diagram of a classifier based on discriminant functions [22].

A Bayes' classifier can be represented in the same way. Based on the Bayes' classifier, unknown data $\mathbf{x}$ are classified on the basis of Bayes' decision rule, which minimizes the conditional risk $R(\delta_i \mid \mathbf{x})$. Since the classification decision of a pattern classifier is based on the maximum discriminant function shown in Eq. (4.10), we define our discriminant function as:

$$d_i(\mathbf{x}) = -R(\delta_i \mid \mathbf{x}) \tag{4.11}$$

[1] Assuming $\mathbf{x}$ is a d-dimensional vector.

As such, the maximum discriminant function corresponds to the minimum conditional risk. In the minimum-error-rate classifier, the decision rule is to maximize the posterior probability $P(\omega_i \mid \mathbf{x})$. Thus, the discriminant function can be written as follows:

$$d_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{\sum_{j=1}^{s} p(\mathbf{x} \mid \omega_j)P(\omega_j)} \tag{4.12}$$

There is a very interesting relationship between Bayes' decision rule and the hypotheses testing method described in Chapter 3. For a two-class pattern recognition problem, the Bayes' decision rule in Eq. (4.2) can be written as follows:

$$p(\mathbf{x} \mid \omega_1)P(\omega_1) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} p(\mathbf{x} \mid \omega_2)P(\omega_2) \tag{4.13}$$

Eq. (4.13) can be rewritten as:

$$\ell(x) = \frac{p(\mathbf{x} \mid \omega_1)}{p(\mathbf{x} \mid \omega_2)} \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \frac{P(\omega_2)}{P(\omega_1)} \tag{4.14}$$

The term $\ell(x)$ is called *likelihood ratio* and is the basic quantity in hypothesis testing [73]. The term $P(\omega_2)/P(\omega_1)$ is called the *threshold value* of the likelihood ratio for the decision. Often it is convenient to use the log-likelihood ratio instead of the likelihood ratio for the decision rule. Namely, the following single discriminant function can be used instead of $d_1(x)$ and $d_2(x)$ for:

$$d(\mathbf{x}) = \log \ell(\mathbf{x}) = \log p(\mathbf{x} \mid \omega_1) - \log p(\mathbf{x} \mid \omega_2) \underset{\omega_2}{\overset{\omega_1}{\gtrless}} \log P(\omega_2) - \log P(\omega_1) \tag{4.15}$$

As the classifier assigns data $\mathbf{x}$ to class ω_i, the data space is divided into s regions, $\Re_1^d, \Re_2^d, \ldots, \Re_s^d$, called decision regions. The boundaries between decision regions are called decision boundaries and are represented as follows (if they are contiguous):

$$d_i(\mathbf{x}) = d_j(\mathbf{x}) \quad i \neq j \tag{4.16}$$

For points on the decision boundary, the classification can go either way. For a Bayes' classifier, the conditional risk associated with either decision is the same and how to break the tie does not matter. Figure 4.3 illustrates an example of decision boundaries and regions for a three-class classifier on a scalar data sample x.

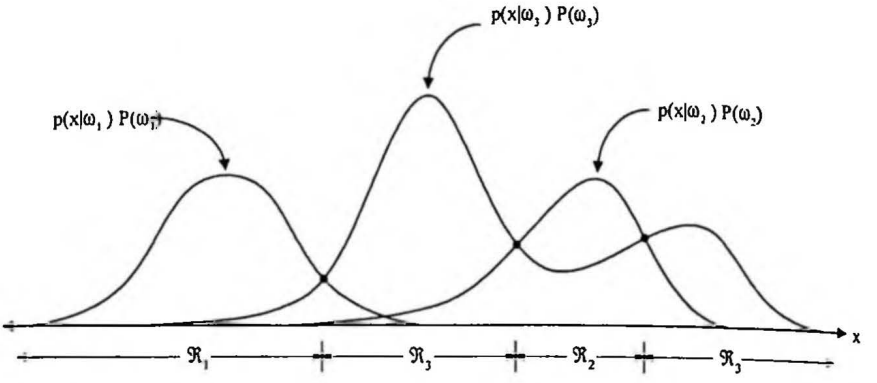

Figure 4.3 An example of decision boundaries and regions. For simplicity, we use scalar variable x instead of a multi-dimensional vector [22].

4.2. How To Construct Classifiers

In the Bayes' classifier, or the minimum-error-rate classifier, the prior probability $P(\omega_i)$ and class-conditional pdf $p(\mathbf{x}|\omega_i)$ are known. Unfortunately, in pattern recognition, we rarely have complete knowledge of class-conditional pdfs and/or prior probability. They often must be estimated or learned from the training data. In practice, the estimation of the prior probabilities is relatively easy. Estimation of the class-conditional pdf is more complicated. There is always concern to have sufficient training data relative to the tractability of the huge dimensionality of the sample data $\mathbf{x}$. In this chapter we focus on estimation methods for the class-conditional pdf.

The estimation of the class-conditional pdfs can be nonparametric or parametric. In nonparametric estimation, no model structure is assumed and the pdf is directly estimated from the training data. When large amounts of sample data are available, nonparametric learning can accurately reflect the underlying probabilistic structure of the training data. However, available sample data are normally limited in practice, and parametric learning can achieve better estimates if valid model assumptions are made. In parametric learning, some general knowledge about the problem space allows one to parameterize the class-conditional pdf, so the severity of sparse training data can be reduced significantly. Suppose the pdf $p(\mathbf{x}|\omega_i)$ is assumed to have a certain probabilistic structure, such as the Gaussian pdf In such cases, only the mean vector $\boldsymbol{\mu}_i$ (or mean μ_i) and covariance matrix Σ_i (or variance σ^2) need to be estimated.

When the observed data x only takes discrete values from a finite set of N values, the class-conditional pdf is often assumed nonparametric, so there will be $N-1$ free parameters in the probability function $p(\mathbf{x}|\omega_i)$.[2] When the observed data $\mathbf{x}$ takes continuous values, parametric approaches are usually necessary. In many systems, the continuous class-conditional pdf (likelihood) $p(\mathbf{x}|\omega_i)$ is assumed to be a Gaussian distribution or a mixture of Gaussian distributions.

In pattern recognition, the set of data samples, which is often collected to estimate the parameters of the recognizer (including the prior and class-conditional pdf), is referred to as the *training set*. In contrast to the training set, the *testing set* is referred to the independent set of data samples, which is used to evaluate the recognition performance of the recognizer.

For parameter estimation or learning, it is also important to distinguish between *supervised learning* and *unsupervised learning*. Let's denote the pair $(\mathbf{x},\omega)$ as a sample, where $\mathbf{x}$ is the observed data and ω is the class from which the data $\mathbf{x}$ comes. From the definition, it is clear that $(\mathbf{x},\omega)$ are jointly distributed random variables. In supervised learning, ω, information about the class of the sample data $\mathbf{x}$ is given. Such sample data are usually called labeled data or complete data, in contrast to incomplete data where the class information ω is missing for unsupervised learning. Techniques for parametric unsupervised learning are discussed in Section 4.4.

In Chapter 3 we introduced two most popular parameter estimation techniques – *maximum likelihood estimation* (MLE) and *maximum a posteriori probability estimation* (MAP). Both MLE and MAP are supervised learning methods since the class information is required. MLE is the most widely used because of its efficiency. The goal of MLE is to find the set of parameters that maximizes the probability of generating the training data actually observed. The class-conditional pdf is typically parameterized. Let Φ_i denote the parameter vector for class i. We can represent the class-conditional pdf as a function of Φ_i as $p(\mathbf{x}|\omega_i,\Phi_i)$. As stated earlier, in supervised learning, the class name ω_i is given for each sample data in training set $\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}$. We need to make an assumption[3] that samples in class ω_i give no information about the parameter vector Φ_j of the other class ω_j. This assumption allows us to deal with each class independently, since the parameter vectors for different categories are functionally independent. The class-conditional pdf can be rewritten as $p(\mathbf{x}|\Phi)$, where $\Phi=\{\Phi_1,\Phi_i,\ldots,\Phi_n\}$. If a set of random samples $\{\mathbf{X}_1,\mathbf{X}_2,\ldots,\mathbf{X}_n\}$ is drawn independently according to a pdf $p(\mathbf{x}|\Phi)$, where the value of the parameter Φ is unknown, the MLE method described in Chapter 3 can be directly applied to estimate Φ.

Similarly, MAP estimation can be applied to estimate Φ if knowledge about a prior distribution is available. In general, MLE is used for estimating parameters from scratch without any prior knowledge, and MAP estimation is used for parameter adaptation where the behavior of a prior distribution is known and only a small amount of adaptation data is available. When the amount of adaptation data increases, MAP estimation converges to MLE.

[2] Since all the probabilities need to add up to one.

[3] This assumption is only true for non-discriminative estimation. Samples in class ω_i may affect parameter vector Φ_i of the other classes in discriminative estimation methods as described in Section 4.3

4.2.1. Gaussian Classifiers

A *Gaussian classifier* is a Bayes' classifier where class-conditional probability density $p(\mathbf{x}\,|\,\omega_i)$ for each class ω_i is assumed to have a Gaussian distribution:[4]

$$p(\mathbf{x}\,|\,\omega_i) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)'\,\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right] \tag{4.17}$$

As discussed in Chapter 3, the parameter estimation techniques are well suited for the Gaussian family. The MLE of the Gaussian parameter is just its sample mean and variance (or co-variance matrix). A Gaussian classifier is equivalent to the one using a quadratic discriminant function. As noted in Eq. (4.12), the discriminant function for a Bayes' decision rule is the posterior probability $p(\omega_i\,|\,\mathbf{x})$ or $p(\mathbf{x}\,|\,\omega_i)P(\omega_i)$. Assuming $p(\mathbf{x}\,|\,\omega_i)$ is a multivariate Gaussian density as shown in Eq. (4.17), a discriminant function can be written as follows:

$$\begin{aligned} d_i(\mathbf{x}) &= \log p(\mathbf{x}\,|\,\omega_i)P(\omega_i) \\ &= -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)'\,\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) + \log P(\omega_i) - \frac{1}{2}\log|\Sigma| - \frac{d}{2}\log 2\pi \end{aligned} \tag{4.18}$$

If we have a uniform prior $P(\omega_i)$, it is clear that the above discriminant function $d_i(\mathbf{x})$ is a quadratic function. Once we have the s Gaussian discriminant functions, the decision process simply assigns data $\mathbf{x}$ to class ω_j if

$$j = \arg\max_i d_i(\mathbf{x}) \tag{4.19}$$

When all the Gaussian pdfs have the same covariance matrix ($\Sigma_i = \Sigma$ for $i = 1,2,\ldots,s$), the quadratic term $\mathbf{x}'\Sigma^{-1}\mathbf{x}$ is independent of the class and can be treated as a constant. Thus the following new discriminant function $d_i(\mathbf{x})$ can be used [22]:

$$d_i(\mathbf{x}) = \mathbf{a}_i'\mathbf{x} + \mathbf{c}_i \tag{4.20}$$

where $\mathbf{a}_i = \Sigma^{-1}\boldsymbol{\mu}_i$ and $\mathbf{c}_i = -\frac{1}{2}\boldsymbol{\mu}_i'\Sigma_i^{-1}\boldsymbol{\mu}_i + \log P(\omega_i)$. $d_i(\mathbf{x})$ in Eq. (4.20) is a linear discriminant function. For linear discriminant functions, the decision boundaries are hyperplanes. For the two-class case (ω_1 and ω_2), and assuming that data sample $\mathbf{x}$ is a real random vector, the decision boundary can be shown to be the following hyperplane:

[4] The Gaussian distribution may include a mixture of Gaussian pdfs.

$$\mathbf{A}^t(\mathbf{x}-\mathbf{b})=0 \tag{4.21}$$

where

$$\mathbf{A}=\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) \tag{4.22}$$

and

$$\mathbf{b}=\frac{1}{2}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)-\frac{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\log[P(\omega_1/\omega_2]}{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^t\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)} \tag{4.23}$$

Figure 4.4 shows a two dimensional decision boundary for a two-class Gaussian classifier with the same covariance matrix. Please note that the decision hyperplane is generally not orthogonal to the line between the means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, although it does intersect that line at the point **b**, which is halfway between $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$. The analysis above is based on the case of uniform priors ($p(\omega_1)=p(\omega_2)$). For nonuniform priors, the decision hyperplane moves away from the more likely mean.

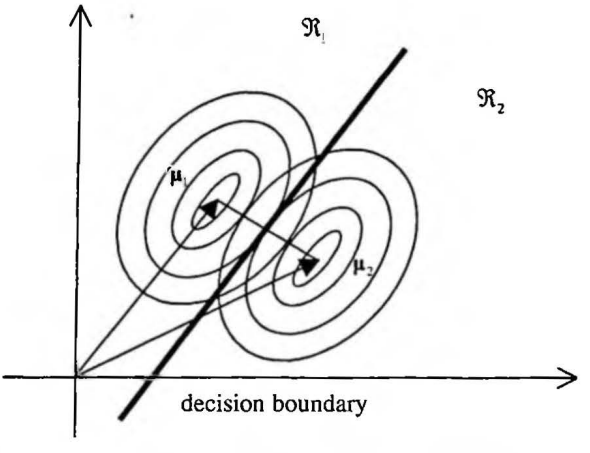

Figure 4.4 Decision boundary for a two-class Gaussian classifier. Gaussian distributions for the two categories have the same covariance matrix Σ. Each ellipse represents the region with the same likelihood probability [22].

Finally, if each dimension of random vector **x** is statistically independent and has the same variance σ^2, i.e., $\Sigma_1=\Sigma_2=\sigma^2\mathbf{I}$, Figure 4.4 becomes Figure 4.5. The ellipse in Figure 4.4 becomes a circle because the variance σ^2 is the same for all dimensions [22].

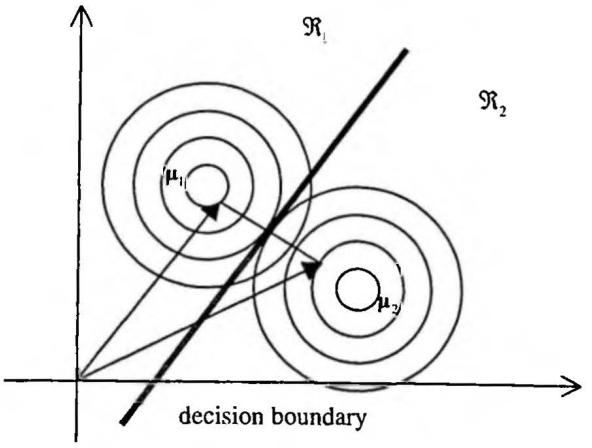

Figure 4.5 Decision boundary for a two-class Gaussian classifier. Gaussian distributions for the two categories have the same covariance matrix $\sigma^2\mathbf{I}$. Each circle represents the region with the same likelihood probability [22].

4.2.2. The Curse of Dimensionality

More features (higher dimensions for sample $\mathbf{x}$) and more parameters for the class-conditional pdf $p(\mathbf{x}|\Phi)$ may lead to lower classification error rate. If the features are statistically independent, there are theoretical arguments that support better classification performance with more features. Let us consider a simple two-class Gaussian classifier. Suppose the prior probabilities $p(\omega_i)$ are equal and the class-conditional Gaussian pdfs $p(\mathbf{x}|\boldsymbol{\mu}_i,\Sigma)$ share the same covariance matrix Σ. According to Eqs. (4.9) and (4.21), the Bayes' classification error rate is given by:

$$\begin{aligned} P(error) &= 2\int_{\Re_2} P(\mathbf{x}|\omega_1)P(\omega_1)d\mathbf{x} \\ &= 2\int_{\mathbf{A}'(\mathbf{x}-\mathbf{b})=0}^{\infty} \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)'\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)\right] d\mathbf{x} \\ &= \frac{1}{\sqrt{2\pi}}\int_{\frac{r}{2}}^{\infty} e^{\frac{-z^2}{2}} dz \end{aligned} \tag{4.24}$$

where $r = \sqrt{(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)'\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)}$. When features are independent, the covariance matrix becomes a diagonal one. The following equation shows that each independent feature helps to reduce the error rate:[5]

[5] When the means of a feature for the two classes are exactly the same, adding such a feature does not reduce the Bayes' error. Nonetheless, according to Eq. (4.25), the Bayes' error cannot possibly be increased by incorporating an additional independent feature.

$$r = \sqrt{\sum_{i=1}^{d} \left(\frac{\mu_{1i} - \mu_{2i}}{\sigma_i} \right)^2} \tag{4.25}$$

where μ_{1i} and μ_{2i} are the i^{th} -dimension of mean vectors μ_1 and μ_2 respectively.

Unfortunately, in practice, the inclusion of additional features may lead to worse classification results. This paradox is called *the curse of dimensionality*. The fundamental issue, called *trainability*, refers to how well the parameters of the classifier are trained from the limited training samples. Trainability can be illustrated by a typical curve-fitting (or regression) problem. Figure 4.6 shows a set of eleven data points and several polynomial fitting curves with different orders. Both the first-order (linear) and second-order (quadratic) polynomials shown provide fairly good fittings for these data points. Although the tenth-order polynomial fits the data points perfectly, no one would expect such an under-determined solution to fit the new data well. In general, many more data samples would be necessary to get a good estimate of a tenth-order polynomial than of a second-order polynomial, because reliable interpolation or extrapolation can be attained only with an over-determined solution.

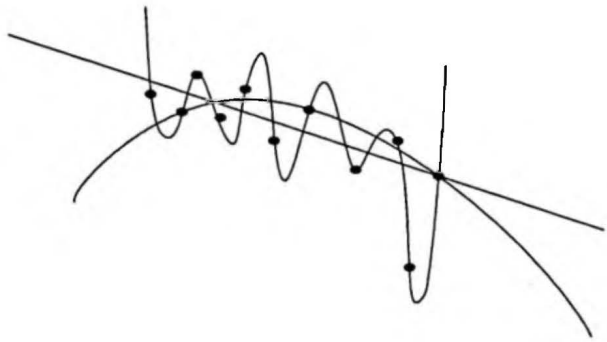

Figure 4.6 Fitting eleven data points with polynomial functions of different orders [22].

Figure 4.7 shows the error rates for two-phonemes (*/ae/* and */ih/*) classification where two phonemes are modeled by mixtures of Gaussian distributions. The parameters of mixtures of Gaussian are trained from a varied set of training samples via maximum likelihood estimation. The curve illustrates the classification error rate as a function of the number of training samples and the number of mixtures. For every curve associated with a finite number of samples, there are an optimal number of mixtures. This illustrates the importance of trainability: it is critical to assure there are enough samples for training an increased number of features or parameters. When the size of training data is fixed, increasing the number of features or parameters beyond a certain point is likely to be counterproductive.

When you have an insufficient amount of data to estimate the parameters, some simplification can be made to the structure of your models. In general, the estimation for higher-order statistics, like variances or co-variance matrices, requires more data than that for lower- order statistics, like mean vectors. Thus more attention often is paid to dealing with

the estimation of covariance matrices. Some frequently used heuristics for Gaussian distributions include the use of the same covariance matrix for all mixture components [77], diagonal covariance matrix, and shrinkage (also referred to as regularized discriminant analysis), where the covariance matrix is interpolated with the constant covariance matrix [23, 50].

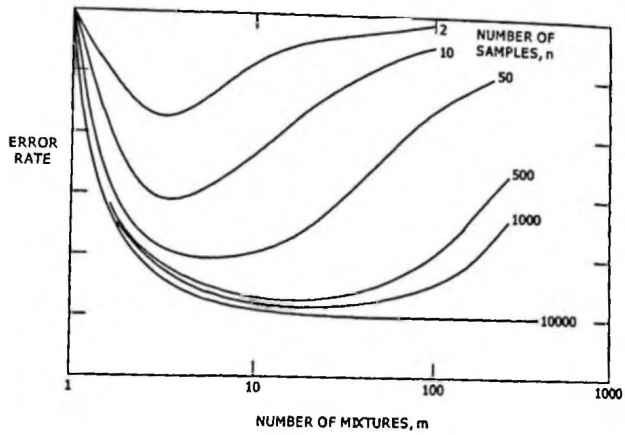

Figure 4.7 Two-phoneme (*/ae/* and */ih/*) classification results as a function of the number of Gaussian mixtures and the number of training samples.

4.2.3. Estimating the Error Rate

Estimating the error rate of a classifier is important. We want to see whether the classifier is good enough to be useful for our task. For example, telephone applications show that some minimum accuracy is required before users would switch from using the touch-tone to the speech recognizer. It is also critical to compare the performance of a classifier (algorithm) against an alternative. In this section we deal with how to estimate the true classification error rate.

One approach is to compute the theoretic error rate from the parametric model as shown in Eq. (4.24). However, there are several problems with this approach. First, such an approach almost always under-estimates, because the parameters estimated from the training samples might not be realistic unless the training samples are representative and sufficient. Second, all the assumptions about models and distributions might be severely wrong. Finally, it is very difficult to compute the exact error rate, as in the simple case illustrated in Eq. (4.24).

Instead, you can estimate the error rate empirically. In general, the recognition error rate on the training set should be viewed only as a lower bound, because the estimate can be made to minimize the error rate on the training data. Therefore, a better estimate of the rec-

ognition performance should be obtained on an independent test set. The question now is how representative is the error rate computed from an arbitrary independent test set. The common process of using some of the data samples for design and reserving the rest for test is called the *holdout* or *H* method.

Suppose the true but unknown classification error rate of the classifier is p, and one observes that k out of n independent randomly drawn test samples are misclassified. The random variable K should have a binomial distribution $B(n, p)$. The maximum likelihood estimation for p should be

$$\hat{p} = \frac{k}{n} \tag{4.26}$$

The statistical test for binomial distribution is discussed in Chapter 3. For a 0.05 significance level, we can compute the following equations to get the range (p_1, p_2):

$$2P(k \le m \le n) = 2\sum_{m=k}^{n} \binom{n}{m} (p_1)^m (1-p_1)^{n-m} = 0.05 \quad \text{when } k > np_1 \tag{4.27}$$

$$2P(0 \le m \le k) = 2\sum_{m=0}^{k} \binom{n}{m} (p_2)^m (1-p_2)^{n-m} = 0.05 \quad \text{when } k < np_2 \tag{4.28}$$

Equations (4.27) and (4.28) are cumbersome to solve, so the normal test described in Chapter 3 can be used instead. The null hypothesis H_0 is

$H_0 : p = \hat{p}$

We can use the normal test to find the two boundary points p_1 and p_2 at which we would not reject the null hypothesis H_0.

The range (p_1, p_2) is called the 0.95 confidence intervals because one can be 95% confident that the true error rate p falls in the range (p_1, p_2). Figure 4.8 illustrates 95% confidence intervals as a function of $\hat{p}$ and n. The curve certainly agrees with our intuition – the larger the number of test samples n, the more confidence we have in the MLE estimated error rate $\hat{p}$; otherwise, the $\hat{p}$ can be used only with caution.

Based on the description in the previous paragraph, the larger the test set is, the better it represents the recognition performance of possible data. On one hand, we need more training data to build a reliable and consistent estimate. On the other hand, we need a large independent test set to derive a good estimate of the true recognition performance. This creates a contradictory situation for dividing the available data set into training and independent test set. One way to effectively use the available database is *V-fold cross validation*. It first splits the entire database into V equal parts. Each part is used in turn as an independent test set while the remaining $(V - 1)$ parts are used for training. The error rate can then be better estimated by averaging the error rates evaluated on the V different testing sets. Thus, each part can contribute to both training and test sets during V-fold cross validation. This procedure, also called the *leave-one-out* or *U* method [53], is particularly attractive when the number of available samples are limited.

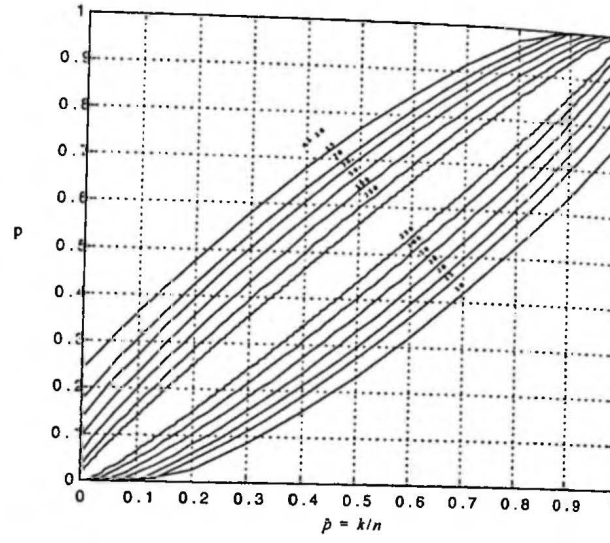

Figure 4.8 95% confidence intervals for classification error rate estimation with normal test.

4.2.4. Comparing Classifiers

Given so many design alternatives, it is critical to compare the performance of different classifiers so that the best classifier can be used for real-world applications. It is common for designers to test two classifiers on some test samples and decide if one is superior to the other. Relative efficacy can be claimed only if the difference in performance is statistically significant. In other words, we establish the null hypothesis H_0 that the two classifiers have the same error rates. Based on the observed error patterns, we decide whether we could reject H_0 at the 0.05 level of significance. The test for different classifiers falls into the category of *matched-pairs* tests described in Chapter 3. Classifiers are compared with the same test samples.

We present an effective matched-pairs test – McNemar's test [66] which is particularly suitable for comparing classification results. Suppose there are two classifiers: Q_1 and Q_2. The estimated classification error rates on the same test set for these two classifiers are p_1 and p_2 respectively. The null hypothesis H_0 is $p_1 = p_2$. The classification performance of the two classifiers can be summarized as in Table 4.1. We define q_{ij} as follows:

$$q_{00} = P(Q_1 \text{ and } Q_2 \text{ classify data sample correctly})$$

$$q_{01} = P(Q_1 \text{ classifies data sample correctly, but } Q_2 \text{ incorrectly})$$

q_{10} = $P(Q_2$ classifies data sample correctly, but Q_1 incorrectly)
q_{11} = $P(Q_1$ and Q_2 classify data sample incorrectly)

Table 4.1 Classification performance table for classifiers Q_1 and Q_2. N_{00} is the number of samples which Q_1 and Q_2 classify correctly, N_{01} is the number of samples which Q_1 classifies correctly, but Q_2 incorrectly, N_{10} is the number of samples which Q_2 classifies correctly, but Q_1 incorrectly, and N_{11} is the number of samples which Q_1 and Q_2 classify incorrectly [30].

		Q_2	
		Correct	Incorrect
Q_1	Correct	N_{00}	N_{01}
	Incorrect	N_{10}	N_{11}

The null hypothesis H_0 is equivalent to $H_0^1 : q_{01} = q_{10}$. If we define $q = q_{10}/(q_{01}+q_{10})$, H_0 is equivalent to $H_0^2 : q = 1/2$. H_0^2 represents the hypothesis that, given only one of the classifiers makes an error, it is equally likely to be either one. We can test H_0^2 based on the data samples on which only one of the classifiers made an error. Let $n = N_{01} + N_{10}$. The observed random variable N_{01} should have a binomial distribution $B(n, 1/2)$. Therefore, the normal test (z-test) described in Chapter 3 can be applied directly to test the null hypothesis H_0^2.

The above procedure is called the *McNemar's test* [66]. If we view the classification results as N (the total number of test samples) independent matched pairs, the sign test as described in Chapter 3 can be directly applied to test the null hypothesis that classifier Q_1 is not better than classifier Q_2, that is, the probability that classifier Q_1 performs better than classifier Q_2, p, is smaller than or equal to ½.

McNemar's test is applicable when the errors made by a classifier are independent among different test samples. Although this condition is true for most static pattern recognition problems, it is not the case for most speech recognition problems. In speech recognition, the errors are highly inter-dependent because of the use of higher-order language models (described in Chapter 11).

The solution is to divide the test data stream into segments in such a way that errors in one segment are statistically independent of errors in any other segment [30]. A natural candidate for such a segment is a sentence or a phrase after which the speaker pauses. Let N_1^i be the number of errors[6] made on the i^{th} segment by classifier Q_1 and N_2^i be the number of errors made on the i^{th} segment by classifier Q_2. Under this formulation, the magnitude-difference test described in Chapter 3 can be applied directly to test the null hypothesis that classifiers Q_1 and Q_2 have on the average the same error rate on the pairs of n independent segments.

[6] The errors for speech recognition include substitutions, insertions and deletions as discussed in Chapter 9.

4.3. DISCRIMINATIVE TRAINING

Both MLE and MAP criteria maximize the probability of the model associated with the corresponding data. Only data labeled as belonging to class ω_i are used to train the parameters. There is no guarantee that the observed data $\mathbf{x}$ from class ω_i actually have a higher likelihood $P(\mathbf{x}|\omega_i)$ than the likelihood $P(\mathbf{x}|\omega_j)$ associated with class j, given $j \neq i$. Models generated by MLE or MAP have a loose discriminant nature. Several estimation methods aim for maximum discrimination among models to achieve best pattern recognition performance.

4.3.1. Maximum Mutual Information Estimation

The pattern recognition problem can be formalized as an information channel, as illustrated in Figure 4.9. The source symbol ω is encoded into data $\mathbf{x}$ and transmitted through an information channel to the observer. The observer utilizes pattern recognition techniques to decode $\mathbf{x}$ into source symbol $\hat{\omega}$. Consistent with the goal of communication channels, the observer hopes the decoded symbol $\hat{\omega}$ is the same as the original source symbol ω. Maximum mutual information estimation tries to improve channel quality between input and output symbols.

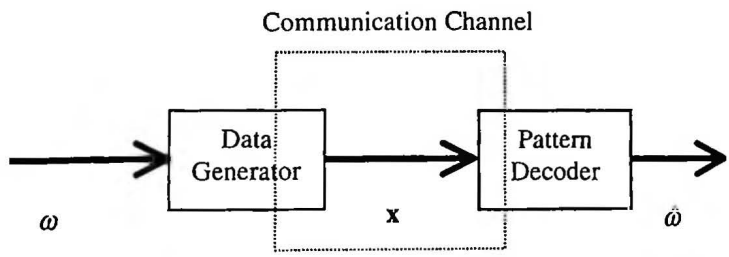

Figure 4.9 An information channel framework for pattern recognition.

As described in Section 4.1.1, the decision rule for the minimum-error-rate classifier selects the class ω_i with maximum posterior probability $P(\omega_i|\mathbf{x})$. It is a good criterion to maximize the posterior probability $P(\omega_i|\mathbf{x})$ for parameter estimation. Recalling Bayes' rule in Section 4.1, the posterior probability $p(\omega_i|\mathbf{x})$ (assuming $\mathbf{x}$ belongs to class ω_i) is:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \tag{4.29}$$

and $p(\mathbf{x})$ can be expressed as follows:

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\omega_k)p(\omega_k) \tag{4.30}$$

In the classification stage, $p(\mathbf{x})$ can be considered as a constant. However, during training, the value of $p(\mathbf{x})$ depends on the parameters of all models and is different for different $\mathbf{x}$. Equation (4.29) is referred to as *conditional likelihood*. A conditional maximum likelihood estimator (CMLE) θ_{CMLE} is defined as follows:

$$\theta_{CMLE}(\mathbf{x}) = \Phi_{MAP} = \underset{\Phi}{\operatorname{argmax}}\ p_{\Phi}(\omega_i \mid \mathbf{x}) \tag{4.31}$$

The summation in Eq. (4.30) extends over all possible classes that include the correct model and all the possible competing models. The parameter vector Φ in Eq. (4.31) includes not only the parameter Φ_i corresponding to class ω_i, but also those for all other classes.

Note that in Chapter 3, the mutual information between random variable $\mathbf{X}$ (observed data) and Ω (class label) is defined as:

$$I(\mathbf{X},\Omega) = E\left[\log\frac{p(\mathbf{X},\Omega)}{p(\mathbf{X})P(\Omega)}\right] = E\left[\log\frac{p(\mathbf{X}\mid\Omega)P(\Omega)}{p(\mathbf{X})P(\Omega)}\right] \tag{4.32}$$

Since we don't know the probability distribution for $p(\mathbf{X},\Omega)$, we assume our sample $(\mathbf{x},\omega_i)$ is representative and define the following *instantaneous mutual information*:

$$I(\mathbf{x},\omega_i) = \log\frac{p(\mathbf{x},\omega_i)}{p(\mathbf{x})P(\omega_i)} \tag{4.33}$$

If equal prior $p(\omega_i)$ is assumed for all classes, maximizing the conditional likelihood in Eq. (4.29) is equivalent to maximizing the mutual information defined in Eq.(4.33). CMLE becomes *maximum mutual information estimation* (MMIE). It is important to note that, in contrast to MLE, MMIE is concerned with distributions over all possible classes. Equation (4.30) can be rewritten as two terms, one corresponding to the correct one, and the other corresponding to the competing models:

$$p(\mathbf{x}) = p(\mathbf{x}\mid\omega_i)P(\omega_i) + \sum_{k\neq i} p(\mathbf{x}\mid\omega_k)P(\omega_k) \tag{4.34}$$

Based on the new expression of $p(\mathbf{x})$ shown in Eq. (4.34), the posterior probability $p(\omega_i \mid \mathbf{x})$ in Eq. (4.29) can be rewritten as:

$$P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x}\mid\omega_i)P(\omega_i)}{p(\mathbf{x}\mid\omega_i)P(\omega_i) + \sum_{k\neq i} p(\mathbf{x}\mid\omega_k)P(\omega_k)} \tag{4.35}$$

Now, maximization of the posterior probability $p(\omega_i \mid \mathbf{x})$ with respect to all models leads to a discriminant model.[7] It implies that the contribution of $p(\mathbf{x}\mid\omega_i)P(\omega_i)$ from the true model needs to be enforced, while the contribution of all the competing models, specified by

[7] General minimum-error-rate estimation is described in Section 4.3.2.

$\sum_{k\neq i} p(\mathbf{x}\mid\omega_k)P(\omega_k)$, needs to be minimized. Maximization of Eq. (4.35) can be further rewritten as:

$$P(\omega_i\mid\mathbf{x}) = \frac{1}{1+\frac{\sum_{k\neq i} p(\mathbf{x}\mid\omega_i)p(\omega_i)}{p(\mathbf{x}\mid\omega_i)p(\omega_i)}} \tag{4.36}$$

Maximization is thus equivalent to maximization of the following term, which is clearly a discriminant criterion between model ω_i and the sum of all other competing models.

$$\frac{p(\mathbf{x}\mid\omega_i)p(\omega_i)}{\sum_{k\neq i} p(\mathbf{x}\mid\omega_k)p(\omega_k)} \tag{4.37}$$

Equation (4.37) also illustrates a fundamental difference between MLE and MMIE. In MLE, only the correct model needs to be updated during training. However, every MMIE model is updated even with one training sample. Furthermore, the greater the prior probability $p(\omega_k)$ for class ω_k, the more effect it has on the maximum mutual information estimator θ_{MMIE}. This makes sense, since the greater the prior probability $p(\omega_k)$, the greater the chance for the recognition system to mis-recognize ω_i as ω_k. MLE is a simplified version of MMIE by restricting the training of model using the data for the model only. This simplification allows the denominator term in Eq. (4.29) to contain the correct model so that it can be dropped as a constant term. Thus, maximization of the posterior probability $p(\omega_i\mid\mathbf{x})$ can be transformed into maximization of the likelihood $p(\mathbf{x}\mid\omega_i)$.

Although likelihood and posterior probability are transformable based on Bayes' rule, MLE and MMIE often generate different results. Discriminative criteria like MMIE attempt to achieve minimum error rate. It might actually produce lower likelihood for the underlying probability density $p(\mathbf{x}\mid\omega_k)$. However, if the assumption of the underlying distributions is correct and there are enough (or infinite) training data, the estimates should converge to the true underlying distributions. Therefore, Bayes' rule should be satisfied and MLE and MMIE should produce the same estimate.

Arthur Nadas [71] showed that if the prior distribution (language model) and the assumed likelihood distribution family are correct, both MLE and MMIE are consistent estimators, but MMIE has a greater variance. However, when some of those premises are not valid, it is desirable to use MMIE to find the estimate that maximizes the mutual information (instead of likelihood) between sample data and its class information. The difference between these two estimation techniques is that MMIE not only aims to increase the likelihood for the correct class, but also tries to decrease the likelihood for the incorrect classes. Thus, MMIE in general possesses more discriminating power among different categories. Although MMIE is theoretically appealing, computationally it is very expensive. Comparing with MLE, every data sample needs to train all the possible models instead of the corresponding model. It also lacks an efficient maximization algorithm. You need to use a gradient descent algorithm.

4.3.1.1. Gradient Descent

To maximize Eq. (4.37) over the entire parameter space $\Phi = \{\Phi_1, \Phi_2, \ldots, \Phi_S\}$ with S classes, we define the mutual information term in Eq. (4.37) to be a function of Φ. To fit into the traditional gradient descent framework, we take the inverse of Eq. (4.37) as our optimization function to minimize the following function:[8]

$$F(\Phi) = \frac{\sum_{k \neq i} p_{i_k}(\mathbf{x} \mid \omega_k) p(\omega_k)}{p_{i_i}(\mathbf{x} \mid \omega_i) p(\omega_i)} \tag{4.38}$$

The gradient descent algorithm starts with some initial estimate Φ^0 and computes the gradient vector $\nabla F(\Phi)$ (∇ is defined in Chapter 3). It obtains a new estimate Φ^1 by moving Φ^0 in the direction of the steepest descent, i.e., along the negative of the gradient. Once it obtains the new estimate, it can perform the same gradient descent procedure iteratively until $F(\Phi)$ converges to the local minimum. In summary, it obtains Φ^{t+1} from Φ^t by the following formula:

$$\Phi^{t+1} = \Phi^t - \varepsilon_t \nabla F(\Phi)\big|_{\Phi=\Phi^t} \tag{4.39}$$

where ε_t is the learning rate (or step size) for the gradient descent.

Why can gradient descent lead $F(\Phi)$ to a local minimum? Based on the definition of gradient vector, $F(\Phi)$ can be approximated by the first order expansion if the correction term $\Delta\Phi$ is small enough.

$$F(\Phi^{t+1}) \approx F(\Phi^t) + \Delta\Phi * \nabla F(\Phi)\big|_{\Phi=\Phi^t} \tag{4.40}$$

$\Delta\Phi$ can be expressed as the following term based on Eq. (4.39)

$$\Delta\Phi = \Phi^{t+1} - \Phi^t = -\varepsilon_t \nabla F(\Phi)\big|_{\Phi=\Phi^t} \tag{4.41}$$

Thus, we can obtain the following equation:

$$\begin{aligned} F(\Phi^{t+1}) - F(\Phi^t) &= -\varepsilon_t \left\langle \nabla F(\Phi)\big|_{\Phi=\Phi^t}, \nabla F(\Phi)\big|_{\Phi=\Phi^t} \right\rangle \\ &= -\varepsilon_t \left\| \nabla F(\Phi)\big|_{\Phi=\Phi^t} \right\|^2 < 0 \end{aligned} \tag{4.42}$$

where $\langle x, y \rangle$ represents the inner product of two vectors, and $\|x\|$ represents the Euclidean norm of the vector. Equation (4.42) means that the gradient descent finds a new estimate Φ^{t+1} that makes the value of the function $F(\Phi)$ decrease.

The gradient descent algorithm needs to go through an iterative hill-climbing procedure to converge to the local minimum (estimate). Gradient descent usually requires many iterations to converge. The algorithm usually stops when the change of the parameter $\Delta\Phi$ becomes small enough. That is,

[8] You can use the logarithm of the object function to make it easier to compute the derivative in gradient descent.

$$\left|\varepsilon_t \nabla F(\Phi)\big|_{\Phi=\Phi'}\right| < \lambda \tag{4.43}$$

where λ is a preset threshold.

Based on the derivation above, the learning rate coefficient ε_t must be small enough for gradient descent to converge. However, if ε_t is too small, convergence is needlessly slow. Thus, it is very important to choose an appropriate ε_t. It is proved [47] [48] that gradient converges almost surely if the learning rate coefficient ε_t satisfies the following condition:

$$\sum_{t=0}^{\infty} \varepsilon_t = \infty, \quad \sum_{t=0}^{\infty} \varepsilon_t^2 < \infty, \quad \text{and } \varepsilon_t > 0 \tag{4.44}$$

One popular choice of ε_t satisfying the above condition is

$$\varepsilon_t = \frac{1}{t+1} \tag{4.45}$$

Another way to find an appropriate ε_t is through the second-order expansion:

$$F(\Phi^{t+1}) \approx F(\Phi^t) + \Delta\Phi \nabla F(\Phi)\big|_{\Phi=\Phi'} + \frac{1}{2}(\Delta\Phi)^t \mathbf{D} \Delta\Phi \tag{4.46}$$

where $\mathbf{D}$ is the *Hessian* matrix [23] of the second-order gradient operator where the i-th row and j-th element $D_{i,j}$ are given by the following partial derivative:

$$D_{i,j} = \frac{\partial^2 F(\Phi)}{\partial \Phi_i \partial \Phi_j} \tag{4.47}$$

By substituting $\Delta\Phi$ from Eq. (4.41) into Eq. (4.46), we can obtain

$$F(\Phi^{t+1}) \approx F(\Phi^t) - \varepsilon_t \|\nabla F\|^2 + \frac{1}{2}\varepsilon_t^2 \nabla F^t \mathbf{D} \nabla F \tag{4.48}$$

From this, it follows that ε_t can be chosen as follows to minimize $F(\Phi)$ [23]:

$$\varepsilon_t = \frac{\|\nabla F\|^2}{\nabla F^t \mathbf{D} \nabla F} \tag{4.49}$$

Sometimes it is desirable to impose a different learning rate for the correct model vs. competing models. Therefore re-estimation Eq. (4.39) can be generalized to the following form [19, 48]:

$$\Phi^{t+1} = \Phi^t - \varepsilon_t U_t \nabla F(\Phi)\big|_{\Phi=\Phi'} \tag{4.50}$$

where U_t is the learning bias matrix which is a positive definite matrix. One particular choice of U_t is $\mathbf{D}^{-1}$, where $\mathbf{D}$ is the Hessian matrix defined in Eq. (4.47). When the learning

rate is set to be 1.0, Eq. (4.50) becomes Newton's algorithm, where the gradient descent is chosen to minimize the second-order expansion. Equation (4.50) becomes:

$$\mathbf{\Phi}^{t+1} = \mathbf{\Phi}^{t} - \mathbf{D}^{-1}\nabla F(\mathbf{\Phi})|_{\mathbf{\Phi}=\mathbf{\Phi}^{t}} \tag{4.51}$$

When probabilistic parameters are iteratively re-estimated, probabilistic constraints must be satisfied in each iteration as probability measure, such as:

1. For discrete distributions, all the values of the probability function ought to be nonnegative. Moreover the sum of all discrete probability values needs to be one, i.e., $\sum_i a_i = 1$
2. For continuous distributions (assuming Gaussian density family), the variance needs to be nonnegative. For Gaussian mixtures, the diagonal covariance entries need to be nonnegative and the sum of mixture weights needs to be one, i.e., $\sum_i c_i = 1$

In general, gradient descent is an unconstrained minimization (or maximization) process that needs to be modified to accommodate constrained minimization (or maximization) problems. The tricks to use are parameter transformations that implicitly maintain these constraints during gradient descent. The original parameters are updated through the inverse transform from the transformed parameter space to the original parameter space. The transformation is done in such a way that constraints on the original parameter are always maintained. Some of these transformations are listed as follows [48]:

1. For probabilities which need to be nonnegative and sum to one, like discrete probability function and mixture weight, the following transformation can be performed:

$$a_i = \frac{\exp(\tilde{a}_i)}{\sum_k \exp(\tilde{a}_k)} \tag{4.52}$$

2. For mean μ and variance (or diagonal covariance entries) σ^2, the following transformation can be used.

$$\mu = \tilde{\mu}\sigma \tag{4.53}$$

$$\sigma = \exp(\tilde{\sigma}) \tag{4.54}$$

After the transformations, we can now compute the gradient with respect to the transformed parameters $(\tilde{a}_i, \tilde{\mu}, \tilde{\sigma})$ using the chain rule. Once the new estimate for the transformed parameters is obtained through gradient descent, one can easily transform them back to the original parameter domain.

4.3.2. Minimum-Error-Rate Estimation

Parameter estimation techniques described so far aim to maximize either the likelihood (class-conditional probability) (MLE and MAP) or the posterior probability (MMIE) in Bayes' equation, Eq. (4.1). Although the criteria used in those estimation methods all have their own merit and under some conditions should lead to satisfactory results, the ultimate parameter estimation criterion for pattern recognition should be to minimize the recognition error rate (or the Bayes' risk) directly. *Minimum-error-rate estimation* is also called *minimum-classification-error* (MCE) training, or discriminative training. Similar to MMIE, the algorithm generally tests the classifier using re-estimated models in the training procedure, and subsequently improves the correct models and suppresses mis-recognized or near-miss models.[9] Neural networks are in this class. Although minimum-error-rate estimation cannot be easily applied, it is still attractive that the criterion is identical to the goal of pattern recognition.

We have used the posterior probability $p(\omega_i \mid \mathbf{x})$ in Bayes' rule as the discriminant function. In fact, just about any discriminant function can be used for minimum-error-rate estimation. For example, as described in Section 4.2.1, a Bayes' Gaussian classifier is equivalent to a quadratic discriminant function. The goal now is to find the estimation of parameters for a discriminant function family $\{d_i(\mathbf{x})\}$ to achieve the minimum error rate. One such error measure is defined in Eq. (4.5). The difficulty associated with the discriminative training approach lies in the fact that the error function needs to be consistent with the true error rate measure and also suitable for optimization.[10] Unfortunately, the error function defined in Section 4.1.1 [Eq. (4.5)] is based on a finite set, which is a piecewise constant function of the parameter vector $\mathbf{\Phi}$. It is not suitable for optimization.

To find an alternative smooth error function for MCE, let us assume that the discriminant function family contains s discriminant functions $d_i(\mathbf{x}, \mathbf{\Phi})$, $i = 1, 2, \ldots, s$. $\mathbf{\Phi}$ denotes the entire parameter set for s discriminant functions. We also assume that all the discriminant functions are nonnegative. We define the following error (misclassification) measure:

$$e_i(\mathbf{x}) = -d_i(\mathbf{x}, \mathbf{\Phi}) + \left[\frac{1}{s-1} \sum_{j \neq i} d_j(\mathbf{x}, \mathbf{\Phi})^{\eta} \right]^{1/\eta} \tag{4.55}$$

where η is a positive number. The intuition behind the above measure is the attempt to enumerate the decision rule. For a ω_i class input $\mathbf{x}$, $e_i(\mathbf{x}) > 0$ implies recognition error; while $e_i(\mathbf{x}) \leq 0$ implies correct recognition. The number η can be thought to be a coefficient to select competing classes in Eq. (4.55). When $\eta = 1$, the competing class term is the average of all the competing discriminant function scores. When $\eta \to \infty$, the competing class term becomes $\max_{j \neq i} d_j(\mathbf{x}, \mathbf{\Phi})$ representing the discriminant function score for the top

[9] A near-miss model occurs when the incorrect model has higher likelihood than the correct model.
[10] In general, a function is optimizable if it is a smooth function and has a derivative.

competing class. By varying the value of η, one can take all the competing classes into account based on their individual significance.

To transform $e_i(\mathbf{x})$ into a normalized smooth function, we can use the sigmoid function to embed $e_i(\mathbf{x})$ in a smooth zero-one function. The loss function can be defined as follows:

$$l_i(\mathbf{x};\Phi) = sigmoid(e_i(\mathbf{x})) \tag{4.56}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$ (4.57)

When $e_i(\mathbf{x})$ is a big negative number, which indicates correct recognition, the loss function $l_i(\mathbf{x};\Phi)$ has a value close to zero, which implies no loss incurred. On the other hand, when $e_i(\mathbf{x})$ is a positive number, it leads to a value between zero and one that indicates the likelihood of an error. Thus $l_i(\mathbf{x};\Phi)$ essentially represents a soft recognition error count.

For any data sample $\mathbf{x}$, the recognizer's loss function can be defined as:

$$l(\mathbf{x},\Phi) = \sum_{i=1}^{s} l_i(\mathbf{x},\Phi)\delta(\omega = \omega_i) \tag{4.58}$$

where $\delta(\bullet)$ is a Boolean function which will return 1 if the argument is true and 0 if the argument is false. Since $\mathbf{x}$ is a random vector, the expected loss according to Eq. (4.58) can be defined as:

$$L(\Phi) = E_{\mathbf{x}}(l(\mathbf{x},\Phi)) = \sum_{i=1}^{s}\int_{\omega=\omega_i} l(\mathbf{x},\Phi)p(\mathbf{x})d\mathbf{x} \tag{4.59}$$

Since $\max_{\Phi}\left[\int f(\mathbf{x},\Phi)d\mathbf{x}\right] = \int\left[\max_{\Phi} f(\mathbf{x},\Phi)\right]d\mathbf{x}$, Φ can be estimated by gradient descent over $l(\mathbf{x},\Phi)$ instead of expected loss $L(\Phi)$. That is, minimum classification error training of parameter Φ can be estimated by first choosing an initial estimate Φ_0 and the following iterative estimation equation:

$$\Phi^{t+1} = \Phi^t - \varepsilon_t \nabla l(\mathbf{x},\Phi)|_{\Phi=\Phi^t} \tag{4.60}$$

You can follow the gradient descent procedure described in Section 4.3.1.1 to achieve the MCE estimate of Φ.

Both MMIE and MCE are much more computationally intensive than MLE, owing to the inefficiency of gradient descent algorithms. Therefore, discriminant estimation methods, like MMIE and MCE, are usually used for tasks containing few classes or data samples. A

more pragmatic approach is *corrective training* [6], which is based on a very simple error-correcting procedure. First, a labeled training set is used to train the parameters for each corresponding class by standard MLE. For each training sample, a list of confusable classes is created by running the recognizer and kept as its *near-miss* list. Then, the parameters of the correct class are moved in the direction of the data sample, while the parameters of the "near-miss" class are moved in the opposite direction of the data samples. After all training samples have been processed; the parameters of all classes are updated. This procedure is repeated until the parameters for all classes converge. Although there is no theoretical proof that such a process converges, some experimental results show that it outperforms both MLE and MMIE methods [4].

We have described various estimators: minimum mean square estimator, maximum likelihood estimator, maximum posterior estimator, maximum mutual information estimator, and minimum error estimator. Although based on different training criteria, they are all powerful estimators for various pattern recognition problems. Every estimator has its strengths and weaknesses. It is almost impossible always to favor one over the others. Instead, you should study their characteristics and assumptions and select the most suitable ones for the domains you are working on.

In the following section we discuss *neural networks*. Both neural networks and MCE estimations follow a very similar discriminant training framework.

4.3.3. Neural Networks

In the area of pattern recognition, the advent of new learning procedures and the availability of high-speed parallel supercomputers have given rise to a renewed interest in neural networks.[11] Neural networks are particularly interesting for speech recognition, which requires massive constraint satisfaction, i.e., the parallel evaluation of many clues and facts and their interpretation in the light of numerous interrelated constraints. The computational flexibility of the human brain comes from its large number of neurons in a mesh of axons and dendrites. The communication between neurons is via the synapse and afferent fibers. There are many billions of neural connections in the human brain. At a simple level it can be considered that nerve impulses are comparable to the phonemes of speech. or to letters, in that they do not themselves convey meaning but indicate different intensities [95, 101] that are interpreted as meaningful units by *the language of the brain*. Neural networks attempt to achieve real-time response and humanlike performance using many simple processing elements operating in parallel as in biological nervous systems. Models of neural networks use a particular topology for the interactions and interrelations of the connections of the *neural units*. In this section we describe the basics of neural networks, including the multi-layer perceptrons and the back-propagation algorithm for training neural networks.

[11] A neural network is sometimes called an artificial neural network (ANN), a neural net, or a connectionist model.

4.3.3.1. Single-Layer Perceptrons

Figure 4.10 shows a basic *single-layer perceptron*. Assuming there are N inputs, labeled as $x_1, x_2, \ldots, x_N$, we can form a linear function with weights $w_{0j}, w_{1j}, w_{2j}, \ldots, w_{Nj}$ to give the output y_j, defined as

$$y_j = w_{0j} + \sum_{i=1}^{N} w_{ij} x_i = \mathbf{w}_j \mathbf{x}^t = g_j(\mathbf{x}) \tag{4.61}$$

where $\mathbf{w}_j = (w_{0j}, w_{1j}, w_{2j}, \ldots, w_{Nj})$ and $\mathbf{x} = (1, x_1, x_2, \ldots, x_N)$.

For pattern recognition purposes, we associate each class ω_j out of s classes $(\omega_1, \omega_2, \ldots, \omega_s)$ with such a linear discriminant function $g_j(\mathbf{x})$. By collecting all the discriminant functions, we will have the following matrix representation:

$$\mathbf{y} = \mathbf{g}(\mathbf{x}) = \mathbf{W}^t \mathbf{x} \tag{4.62}$$

where $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_s(\mathbf{x}))^t$; $\mathbf{W} = (\mathbf{w}_1^t, \mathbf{w}_2^t, \ldots, \mathbf{w}_s^t)^t$ and $\mathbf{y} = (y_1, y_2, \ldots, y_s)^t$. The pattern recognition decision can then be based on these discriminant functions as in Bayes' decision theory. That is,

$$\mathbf{x} \in \omega_k \quad \text{iff} \quad k = \arg\max_i g_i(\mathbf{x}) \tag{4.63}$$

The *perceptron training algorithm* [68], guaranteed to converge for linearly separable classes, is often used for training the weight matrix $\mathbf{W}$. The algorithm basically divides the sample space $\Re^N$ into regions of corresponding classes. The decision boundary is characterized by hyper-planes of the following form:

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0 \quad \forall i \neq j \tag{4.64}$$

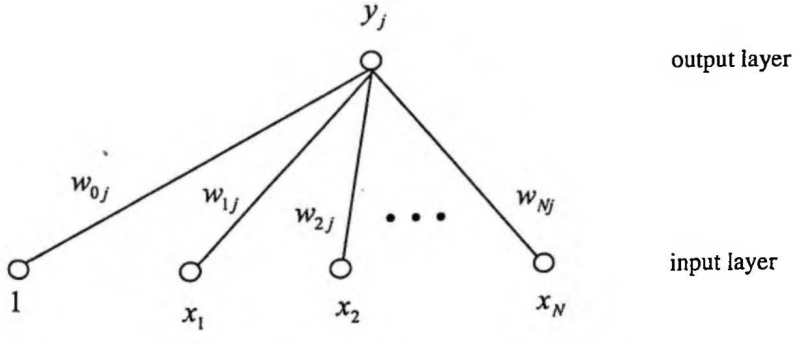

Figure 4.10 A single-layer perceptron.

Unfortunately, for data samples that are not linearly separable, the perceptron algorithm does not converge. However, if we can relax the definition of classification errors in this case, we can still use a powerful algorithm to train the weight matrix $\mathbf{W}$. This approach is the *least square error* (LSE) algorithm described in Chapter 3, which aims at minimizing *sum-of-squared-error* (SSE) criterion, instead of minimizing the classification errors. The sum-of-squared-error is defined as:

$$\text{SSE} = \sum_i \sum_{\mathbf{x}\in\omega_i} \| \mathbf{g}(\mathbf{x}) - \delta_i \|^2 \tag{4.65}$$

where δ_i is an M-dimensional *index vector* with all zero components except that the i^{th} one is 1.0, since the desired output for $\mathbf{g}(\mathbf{x})$ is typically equal to 1.0 if $\mathbf{x}\in\omega_i$ and 0 if $\mathbf{x}\notin\omega_i$.

The use of LSE leads to discriminant functions that have real outputs approximating the values 1 or 0. Suppose there are M input vectors $\mathbf{X} = (\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_M^t)$ in the training set. Similar to the LSE for linear functions described in Chapter 3 (cf. Section 3.2.1.2), the LSE estimate of weight matrix $\mathbf{W}$ will have the following closed form:

$$\mathbf{W} = ((\mathbf{X}\mathbf{X}^t))^{-1}\mathbf{L}\Sigma \tag{4.66}$$

where $\mathbf{L}$ is a $(N+1)\times s$ matrix where the k-th column is the mean vector $\boldsymbol{\mu}_k = (1, \mu_{k1}, \mu_{k2}, \ldots, \mu_{kN})^t$ of all the vectors classified into class ω_k, and Σ is an $s\times s$ diagonal matrix with diagonal entry $c_{k,k}$ representing the number of vectors classified into ω_k. LSE estimation using linear discriminant functions is equivalent to estimating Bayes' Gaussian densities where all the densities are assumed to share the same covariance matrix [98], as described in Section 4.2.1.

Although the use of LSE algorithm solves the convergence problems, it loses the power of nonlinear logical decision (i.e., minimizing the classification error rate), since it is only approximating the simple logical decision between alternatives. An alternative approach is to use a smooth and differential sigmoid function as the threshold function:

$$\begin{aligned}\mathbf{y} &= sigmoid(\mathbf{g}(\mathbf{x})) = sigmoid((g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_s(\mathbf{x}))^t) \\ &= (sigmoid(g_1(\mathbf{x})), sigmoid(g_2(\mathbf{x})), \ldots, sigmoid(g_s(\mathbf{x})))^t\end{aligned} \tag{4.67}$$

where $sigmoid(x)$ is the sigmoid function defined in Eq. (4.57). With the sigmoid function, the following new sum-of-squared-error term closely tracks the classification error:

$$\text{NSSE} = \sum_i \sum_{\mathbf{x}\in\omega_i} \| sigmoid(\mathbf{g}(\mathbf{x})) - \delta_i \|^2 \tag{4.68}$$

where δ_i is the same index vector defined above. Since there is no analytic way of minimizing a nonlinear function, the use of the sigmoid threshold function requires an iterative gradient descent algorithm, back-propagation, which will be described in the next section.

4.3.3.2. Multi-Layer Perceptron

One of the technical developments sparking the recent resurgence of interest in neural networks has been the popularization of *multi-layer perceptrons* (MLP) [37, 90]. Figure 4.11 shows a multi-layer perceptron. In contrast to a single-layer perceptron, it has two hidden layers. The hidden layers can be viewed as feature extractors. Each layer has the same computation models as the single-layer perceptron; i.e., the value of each node is computed as a linear weighted sum of the input nodes and passed to a sigmoid type of threshold function.

$$\begin{aligned}\mathbf{h}_1 &= sigmoid(\mathbf{g}_{h1}(\mathbf{x})) = sigmoid(\mathbf{W}_{h1}^t\mathbf{x}) \\ \mathbf{h}_2 &= sigmoid(\mathbf{g}_{h2}(\mathbf{h}_1)) = sigmoid(\mathbf{W}_{h2}^t\mathbf{h}_1) \\ \mathbf{y} &= sigmoid(\mathbf{g}_y(\mathbf{h}_2)) = sigmoid(\mathbf{W}_y^t\mathbf{h}_2)\end{aligned} \tag{4.69}$$

where $sigmoid(x)$ is the sigmoid function defined in Eq. (4.57).

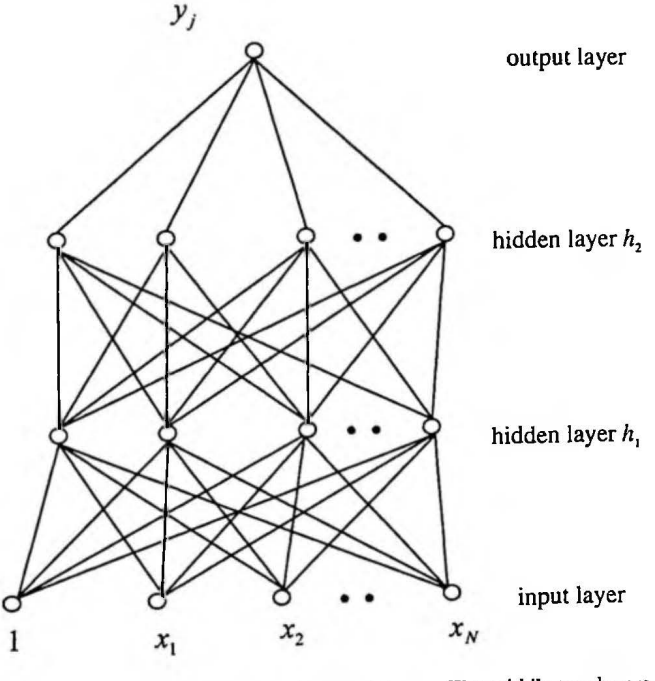

Figure 4.11 A multi-layer perceptron with four total layers. The middle two layers are hidden.

According to Eq. (4.69), we can propagate the computation from input layer to output layer and denote the output layer as a nonlinear function of the input layer.

$$\mathbf{Y} = MLP(\mathbf{x}) \tag{4.70}$$

Let's denote $O(\mathbf{x})$ as the desired output for input vector $\mathbf{x}$. For pattern classification, $O(\mathbf{x})$ will be an s-dimensional vector with the desired output pattern set to one and the remaining patterns set to zero. As we mentioned before, there is no analytic way to minimize the mean square error $E = \sum_{\mathbf{x}} \| MLP(\mathbf{x}) - O(\mathbf{x}) \|^2$. Instead, an iterative gradient descent algorithm called back propagation [89, 90] needs to be used to reduce error. Without loss of generality, we assume there is only one input vector $\mathbf{x} = (1, x_1, x_2, \ldots, x_N)$ with desired output $\mathbf{o} = (o_1, o_2, \ldots, o_s)$. All the layers in the MLP are numbered 0, 1, 2,... upward from the input layer. The back propagation algorithm can then be described as in Algorithm 4.1.

In computing the partial derivative $\frac{\partial E}{\partial w_{ij}^k(t)}$, you need to use the chain rule. w_{ij}^K is the weight connecting the output layer and the last hidden layer; the partial derivative is:

$$\begin{aligned}
\frac{\partial E}{\partial w_{ij}^K} &= \frac{\partial (\sum_{i=1}^{s} (y_i - o_i)^2)}{\partial w_{ij}^K} \\
&= \frac{\partial (\sum_{i=1}^{s} (y_i - o_i)^2)}{\partial y_j} \times \frac{\partial y_j}{\partial (w_{0j}^K + \sum_{i=1}^{N} w_{ij}^K v_i^{K-1})} \times \frac{\partial (w_{0j}^K + \sum_{i=1}^{N} w_{ij}^K v_i^{K-1})}{\partial w_{ij}^K} \\
&= 2(y_i - o_i) y_j (y_i - 1) v_i^{K-1}
\end{aligned} \tag{4.71}$$

For layers $k = K-1, K-2, \cdots$, one can apply chain rules similarly for gradient $\frac{\partial E}{\partial w_{ij}^k(t)}$.

The back propagation algorithm is a generalization of the minimum mean squared error (MMSE) algorithm. It uses a gradient search to minimize the difference between the desired outputs and the actual net outputs, where the optimized criterion is directly related to pattern classification. With initial parameters for the weights, the training procedure is then repeated to update the weights until the cost function is reduced to an acceptable value or remains unchanged. In the algorithm described above, we assume a single training example. In real-world application, these weights are estimated from a large number of training observations in a manner similar to hidden Markov modeling. The weight updates in Step 3 are accumulated over all the training data. The actual gradient is then estimated for the complete set of training data before the beginning of the next iteration. Note that the estimation criterion for neural networks is directly related to classification rather than maximum likelihood.

ALGORITHM 4.1: *THE BACK PROPAGATION ALGORITHM*

Step 1: Initialization: Set $t = 0$ and choose initial weight matrices $\mathbf{W}$ for each layer. Let's denote $w_{ij}^k(t)$ as the weighting coefficients connecting i^{th} input node in layer $k-1$ and j^{th} output node in layer k at time t.

Step 2: Forward Propagation: Compute the values in each node from input layer to output layer in a propagating fashion, for $k = 1$ to K

$$v_j^k = sigmoid(w_{0j}(t) + \sum_{i=1}^{N} w_{ij}^k(t) v_i^{k-1}) \quad \forall j \tag{4.72}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$ and v_j^k is denoted as the j^{th} node in the k^{th} layer

Step 3: Back Propagation: Update the weights matrix for each layer from output layer to input layer according to:

$$\overline{w}_{ij}^k(t+1) = w_{ij}^k(t) - \alpha \frac{\partial E}{\partial w_{ij}^k(t)} \tag{4.73}$$

where $E = \sum_{i=1}^{s} \| y_i - o_i \|^2$ and $(y_1, y_2, \ldots y_s)$ is the computed output vector in Step 2.

α is referred to as the learning rate and has to be small enough to guarantee convergence. One popular choice is $1/(t+1)$.

Step 4: Iteration: Let $t = t+1$. Repeat Steps 2 and 3 until some convergence condition is met.

4.4. UNSUPERVISED ESTIMATION METHODS

As described in Section 4.2, in unsupervised learning, information about class ω of the data sample $\mathbf{x}$ is unavailable. Data observed are *incomplete* since the class data ω is missing. One might wonder why we are interested in such an unpromising problem, and whether or not it is possible to learn anything from incomplete data. Interestingly enough, the formal solution to this problem is almost identical to the solution for the supervised learning case – MLE. We discuss vector quantization (VQ), which uses principles similar to the EM algorithm. It is important in its own right in spoken language systems.

4.4.1. Vector Quantization

As described in Chapter 3, source coding refers to techniques that convert the signal source into a sequence of bits that are transmitted over a communication channel and then used to