3

Nonlinear Parameter Estimation

It is not always possible to establish a qualitative comparison of models. If the predictions overlap to some extent, or if the predictions vary depending on parameters, then we need to resort to a quantitative comparison of the models. Even when a qualitative test is possible, examining the quantitative predictions would still prove to be informative. It is not sufficient for a model to predict the correct qualitative ordering of performance across two conditions. It is also necessary for a model to make quantitative predictions that are more accurate than its competitors.

When the quantitative predictions of a model are to be evaluated, the test must be performed on the basis of an optimal selection of parameters. Otherwise, one could reject a perfectly good model simply because the researcher happened to select a poor set of parameters to evaluate the model. This implies that the quantitative comparison of the two models would depend completely on the researcher's arbitrary selection of parameters.

The purpose of this chapter is to review some of the technical issues involved in estimating parameters for cognitive models. First, we introduce a very simple experiment and its (fictitious) results to provide a concrete example for this chapter. Second, we present a correspondingly simple cognitive model for the example. In this section, we also introduce the distinction between linear and nonlinear models. This distinction is important because cognitive models usually fall into the latter class. Third, we introduce some alternative methods for measuring the quantitative accuracy of a model, including sum of squared errors and maximum-likelihood methods. Fourth, we review some properties of the parameters obtained by these methods, which provide a rationale for selecting an estimation method. Fifth, we discuss some of the technical details concerning the parameter search methods and try to indicate the most appropriate conditions for using each method. Finally, we present the results of using these methods on the cognitive model applied to the experimental data of our example.

The primary purpose of this chapter is to review some of the technical issues involved in estimating parameters for cognitive models. This chapter does not address the important problem of quantitative model comparisons, which is a complex topic that is discussed in detail in later chapters. First, we introduce a very simple experiment and its (fictitious) results to provide a concrete example for this chapter. The example data and model used in this chapter are based on a similar research that has been published by Ruben and Wenzel (1996).

Retention Experiment

Stimuli and Procedure

Suppose 5 amnesic participants and 5 control participants are initially trained on a category learning experiment in which they are trained to categorize mushrooms as edible or inedible. For the experiment, the stimuli are more complex, being characterized by four dimensions, namely, length of stem, width of rim, lightness, and texture. The participants are trained on the two categories for 400 trials, and then they are tested on new transfer stimuli. For the experiment analyzed in this chapter, the results are based on a new set of 200 transfer test stimuli that are presented at each of 11 different delay conditions: immediately after training, after 1 week, after 2 weeks, and so on, and finally after 10 weeks.

Figure 3.1 shows the percentage correct categorization as a function of delay, averaged over the 200 transfer stimuli, and averaged across the 5 participants within each group. The points plotted in the figure represent the (fictitious) data, and the lines connecting the points are predictions (discussed later). The amnesic group performed almost as well as the controls after a single-week delay. But after the first week, the control participants retained their memory for the categories longer than the amnesic group.

Retention Model

Now we will develop a simple model for this retention performance based on the connectionist version of the exemplar model discussed in the previous chapter. First, we need to introduce an assumption about the effect of delay for the exemplar model. A simple hypothesis is that the connection weights decay back toward zero with each delay period.

The exemplar model assumes that there exists a list of input nodes that are activated by the stimulus presented on trial t, and the activations of these

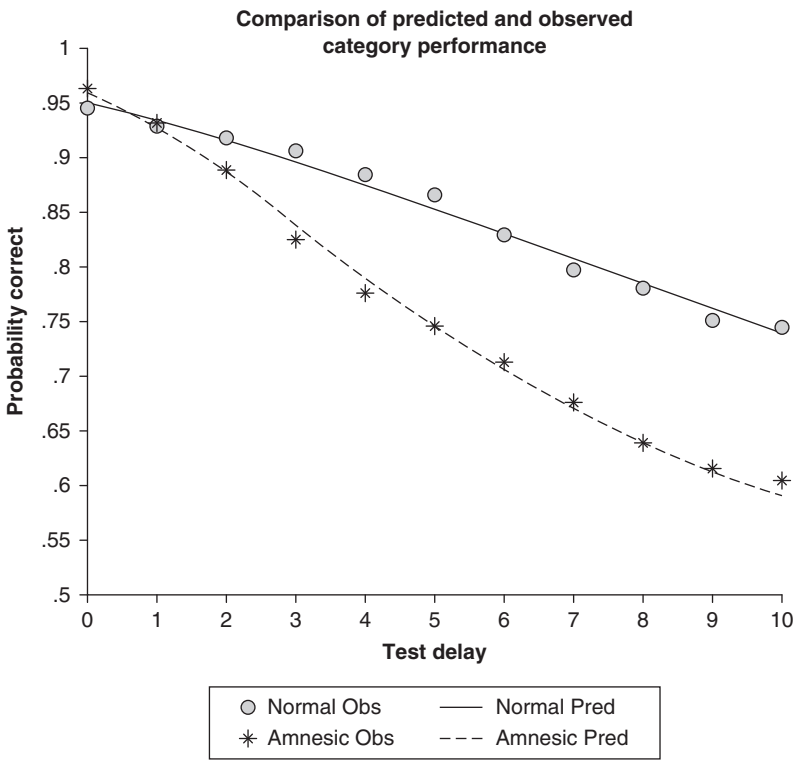

Figure 3.1 Fictitious Results

nodes are represented by the variables $\{x_1(t), x_2(t), \ldots, x_j(t), \ldots, x_J(t)\}$. The connection weight connecting an input node j to a response category k immediately after training is denoted as $w_{j,k}$. The connection weight that remains after waiting a time period equal to d weeks is denoted as $w_{j,k}(d)$, where $d = 0, \ldots, 10$ weeks. Note that $w_{j,k}(0) = w_{j,k}$, which is just the connection weight immediately after training. The decay hypothesis states that

$$w_{j,k}(d + 1) = \gamma \cdot w_{j,k}(d), \text{ for } d = 0, 1, \ldots, 10.$$

For example, $w_{j,k}(1) = \gamma \cdot w_{j,k}(0) = \gamma^1 \cdot w_{j,k}$, and $w_{j,k}(2) = \gamma \cdot w_{j,k}(1) = \gamma^2 \cdot w_{j,k}$, and $w_{j,k}(3) = \gamma \cdot w_{j,k}(2) = \gamma^3 \cdot w_{j,k}$; and inductively we have $w_{j,k}(d) = \gamma^d \cdot w_{j,k}$. The parameter γ is a decay rate that ranges between 0 and 1, and faster decay or forgetting is produced by smaller values of the decay parameter.

Next, consider how this delay affects the output activations after a delay. The output activation for kth category evoked by the stimulus presented on

trial t after a delay of d weeks is denoted as $r_k(t, d)$. Recall that the output activation for each category equals the sum of the products of the input activations and connection weights. After a delay of d weeks, this equals

$$\begin{aligned} r_k(t,d) &= \sum_j w_{j,k}(d) \cdot x_j(t) = \sum_j (\gamma^d \cdot w_{j,k}) \cdot x_j(t) \\ &= \gamma^d \cdot \sum_j w_{j,k} \cdot x_j(t) = \gamma^d \cdot r_k(t,0) = \gamma^d \cdot r_k(t), \end{aligned}$$

where $r_k(t)$ is the output activation immediately after training.

Finally, consider the probability of categorizing a stimulus into Category A after a delay of d weeks, denoted as $\Pr[\mathrm{A}|S(t), d]$. This is given by the ratio of the strengths of the output activations

$$\Pr[\mathrm{A}|S(t), d] = \frac{e^{b \cdot r_{\mathrm{A}}(t) \cdot \gamma^d}}{e^{b \cdot r_{\mathrm{A}}(t) \cdot \gamma^d} + e^{b \cdot r_{\mathrm{B}}(t) \cdot \gamma^d}}. \tag{3.1}$$

Note that in the original formulation of the exemplar model presented in Chapter 2, the output activations, $r_k(t)$, depend on the particular stimulus presented on trial t. Here, we will make a simplifying assumption to minimize the complexity of the model for this chapter. We will assume that all the stimuli belonging to Category A produce approximately the same output activations for Categories A and B, denoted as r_{C} and r_{I} for correct and incorrect, respectively. Similarly, we will assume that all the stimuli belonging to Category B produce approximately the same output activations for Categories B and A, which also equal r_{C} and r_{I} for correct and incorrect, respectively. Using this simplifying assumption, the probability of making a correct categorization is only a function of the delay:

$$P(d) = \frac{e^{b \cdot r_{\mathrm{C}} \cdot \gamma^d}}{e^{b \cdot r_{\mathrm{C}} \cdot \gamma^d} + e^{b \cdot r_{\mathrm{I}} \cdot \gamma^d}}. \tag{3.2}$$

Equation 3.2 is a simple model of retention, which expresses the performance measure on the left-hand side as a function of a single independent variable, denoted as d, for the delay, on the right-hand side. In this form, the model has four parameters $\{b, r_{\mathrm{C}}, r_{\mathrm{I}}, \text{and } \gamma\}$.

This simplifying assumption needs some justification. First, as we mentioned above, we want to focus on parameter estimation, and so we want to keep the model as simple as possible for pedagogical reasons. Beyond that,

simplifying assumptions help us see mathematical properties that are not apparent in the more complex model. Also, if the simplifying assumption is approximately correct, then we can make nearly the same predictions using a much simpler equation, and this greatly facilitates later analyses.

Statistical Models

It is difficult to evaluate a model in an absolute sense. No model is perfect, so how can one say a model is good without comparing it with other competitors? Therefore, we will compare this cognitive model with two other purely statistical models—the saturated model and the null model, which provide the upper and lower bounds for fit indices. These models are not derived from any cognitive principles, but they are helpful for evaluating the fit of the cognitive model.

The saturated model uses a new free parameter to predict the probability correct at each delay condition. (It is called the saturated model because it has the same number of free parameters as the number of data points.) This model perfectly reproduces the observed proportions, because it simply makes a prediction for each condition that is equal to the observed relative frequency for that condition. Thus, the prediction for condition d is simply: $p_d = (n_{Cd}/n)$, where n_{Cd} equals the frequency of correct choices, n_{Id} equals the frequency of incorrect choices, and $n = (n_{Cd} + n_{Id})$ the total number of trials for condition d ($n = 200$ observations per condition for each person in our example). The saturated model provides an upper bound for measuring model fit. Obviously, this model has no explanatory power, but it is useful in measuring how far the fit of the retention model is below the upper bound.

The null model assumes that probability correct remains constant across all conditions, that is, there is no true effect of the delay on performance. Any deviation from constant performance is assumed to be sampling error. This model has only one free parameter, which is estimated by the mean proportion correct, averaged across all the 11 conditions: $(\Sigma p_d/11)$. This mean is used as the prediction for all the 11 conditions. The null model is the simplest possible model, and it provides a lower bound for measuring model fit. Obviously, this model is wrong, but it is useful for measuring how far the fit of the retention model is above the lower bound.

The improvement in fit of the cognitive model over the null model indicates the amount of the treatment effect that is predicted by the cognitive model, and improvement of the saturated model over the cognitive model indicates the amount of the treatment effect left unexplained by the cognitive model.

Linear Versus Nonlinear Models

Estimating the parameters of the cognitive model is analogous to estimating the regression coefficients of the linear model. The basic idea is to search for the set of parameter values that produces the best fit to the data. However, there is an important technical difference between estimating parameters for the linear and cognitive models. The former is a special case of a statistical class of models called the general linear class, whereas the latter is in the nonlinear class.

A linear model can be recognized by its simple form: Each unknown parameter is multiplied by a known number (the known score on a predictor variable), and these products are summed to produce the prediction: $y' = \sum \beta_j \cdot x_j$. The xs in the equation are treated as known numbers, and the parameters β_j are the unknown variables. The xs in this equation can be anything: For example, setting $x_1 = d^1$ and $x_2 = d^2$, where d is the value of the independent variable, still produces a linear model; so does setting $x_1 = \sin d$, $x_2 = \cos d$. Estimating the parameters of a linear model can usually be done with a single-step algorithm that is guaranteed to produce an optimal solution. Linear models satisfy a special condition: The average of the predictions from two different sets of parameters equals the prediction produced by the average of the two sets of parameters. Nonlinear models do not satisfy this property. (This distinction between linear and nonlinear models is presented in the appendix to this chapter.)

The retention model (Equation 3.2) is a nonlinear model, and this has important implications for the estimation of parameters—there are no known single-step solutions for estimating the parameters. Instead, an iterative process (a sequence of steps where the next step in the process depends on the earlier steps) must be used to search the parameter space for the optimal solution, and there is no guarantee that the optimal solution will be found. The later sections of this chapter treat these issues in more depth. But first we need to address several other important issues.

Parameter Identification

Before we charge ahead and try to estimate the parameters of a model, we need to check and make sure that it is indeed possible to obtain a unique optimal solution. One condition that is usually necessary for the parameters to be identifiable is that there are more data points than parameters. The difference between the number of data points and parameters is called the degrees of freedom (denoted as *df*), and this should generally be greater than

zero. However, this is only a necessary condition, and it is not a sufficient condition for all the parameters to be identified.

Referring back to our retention model, we have four parameters $\{b, r_C, r_I, \gamma\}$, but it is not possible to estimate all these parameters with the present experimental design. To understand this issue better, it will help to rewrite Equation 3.2 in its alternative form:

$$P(d) = \frac{e^{b \cdot r_C \cdot \gamma^d}}{e^{b \cdot r_C \cdot \gamma^d} + e^{b \cdot r_I \cdot \gamma^d}} \cdot \frac{e^{-b \cdot r_C \cdot \gamma^d}}{e^{-b \cdot r_C \cdot \gamma^d}} = \frac{1}{1 + e^{-b \cdot (r_C - r_I) \cdot \gamma^d}}. \qquad (3.3)$$

Suppose we generated artificial data with the parameters $b = 3$ and $(r_C - r_I) = 2$, so that the product in the exponent of Equation 3.3 equals $-6\gamma^d$. Then, suppose we try to fit the same artificial data, but we mistakenly set $b = 1$ during the fitting process. Can we still exactly reproduce the artificial data? The answer is yes, provided that we adjust $(r_C - r_I)$ by multiplying the original value by 3 so that it equals $(r_C - r_I) = 2 \cdot 3 = 6$. Then, exactly the same predictions are produced because the product in the exponent of Equation 3.3 remains the same. More generally, we can set the parameter b to any arbitrary nonzero value (e.g., $b = 1$) and then adjust the parameters r_C and r_I into $r_C \times b$ and $r_I \times b$ to accommodate this arbitrary selection. Only the product of $b \times (r_C - r_I)$ is needed. Therefore, we can eliminate the parameter b without any loss in predictive power of the model. Thus, the parameter b cannot be identified in this application, and we simply fix it to some easily interpretable value such as $b = 1$.

A similar problem of identification occurs with the two parameters r_C and r_I. Suppose, we generated the data setting $r_C = 10$ and $r_I = 5$. Then, suppose we tried to exactly fit the same data after arbitrarily setting $r_I = 0$. We could adjust r_C so that $r_C = (10 - 5) = 5$ and exactly reproduce the same results. Only the difference $(r_C - r_I)$ is important, and so we can replace the two parameters with a single parameter $r = (r_C - r_I)$, without any loss in generality. We would then interpret this estimate of r as the original difference.

What about the parameter γ? Can this be set to some arbitrary value without losing any generality? The answer is no because there is no way to adjust any other parameter in the model to compensate for changes in this parameter. This parameter is an exponential function of the independent variable, the delay d, and hence no other parameter can compensate for this feature. Thus, the decay rate is an identifiable parameter.

In sum, our retention model has only two identifiable parameters, the initial connection strength parameter r and the decay rate parameter γ that need to be estimated from the data, and so we have $11 - 2 = 9$ *df*, where 11 comes

from the number of data points from our 11 experimental conditions. Therefore, we can rewrite the retention model in the reduced form without any loss in generality:

$$P(d) = \frac{1}{1 + e^{-r \cdot \gamma^d}} \tag{3.4}$$

Hereafter, this form of the retention model will be used in all the remaining analyses.

There is one last comment that needs to be given about parameter identification. This issue depends on the design of the experiment. For some experimental designs, a set of parameters may not be identified, but if we changed the design, then they can be identified. For example, in the present experimental design, the parameter b is not identified because it is not manipulated. However, if we manipulated this parameter by some experimental factor at K different levels, then this parameter is not identified for the first level, but it is identified for the other $K - 1$ levels (i.e., this parameter is identified up to a ratio scale). For example, we could manipulate b across three levels by emphasizing the speed versus the accuracy of the decisions (low-, medium-, and high-speed stress). Then, this parameter would not be identified for the low-stress level, but it would be identified for the other two levels. (A general mathematical method for determining parameter identification is discussed in the appendix.)

Data Representation

Before we can estimate the parameters from the data, we need to be clear about the data to be used in the estimation procedure. At first, this may seem like a trivial step, but there are several important issues that must be decided. Consider the example from this chapter, where we have data from 5 participants within each group, and proportions from 11 delayed test conditions for each participant, with 200 observations per proportion.

Aggregate Modeling

The first approach is the aggregate data-fitting approach. Using this approach, we fit the choice proportions separately for each group, pooled across participants within each group. In this case, the data set would consist of the 11 choice proportions, per group, corresponding to the 11 delay conditions, per group, as seen in Figure 3.1. One set of parameters would be

estimated from the normal group, and a second set would be estimated from the amnesic group. However, this approach implicitly assumes that there are no important individual differences within each group. More technically, this approach assumes that there is no variance in the model parameters within each group. For example, this approach would require one to assume that all individuals within a group have exactly the same decay rates, or that all individuals within a group have exactly the same initial association strengths.

Many cognitive researchers consider individual differences to be very important and therefore reject the aggregate data approach (see Cohen, Sanborn, & Shiffrin, in press; Estes & Maddox, 2005). If individual differences are strong, and usually they are, then fitting the model to the aggregate data, averaged across individuals, can be very misleading. Consider the following well-known example from early learning theory. Early concept learning theorists were interested in comparing all-or-none learning models with incremental strength learning models. At the individual level, the all-or-none model produces a learning curve that starts at chance performance and jumps suddenly to a solution that produces perfect performance. In contrast, the incremental learning model produces a smooth and gradually increasing learning curve for each individual. Now suppose we generate a fictitious data set containing 100 simulated subjects from the all-or-none learning model, but we allow large individual differences in the amount of training required to find the correct solution to the problem. In this case, the true model is known to be the all-or-none model, and all the individual learning curves reveal a jump from chance to perfect performance, but the jump occurs at a different point in training for each person. Next, suppose we average the data across subjects. This average would generate a smooth and gradually increasing learning curve for proportion correct, consistent with the incremental learning model. Furthermore, if we compare the fits of the two models, the incremental model could produce a superior fit, even though we know for sure that it is the wrong model. The final section in this chapter provides a concrete example of this problem.

Individual Modeling

The second approach is the individual data-fitting approach. Using this approach, we fit the model to the 11 proportions from each individual separately, allowing separate parameters and even separate models to be best fit for each person. This approach requires a large amount of data from each individual. Figure 3.2 is an example of a (fictitious) data set from one of the individuals taken from each group. Comparing this with the average data shown in Figure 3.1, it is obvious that the individual data are noisier.

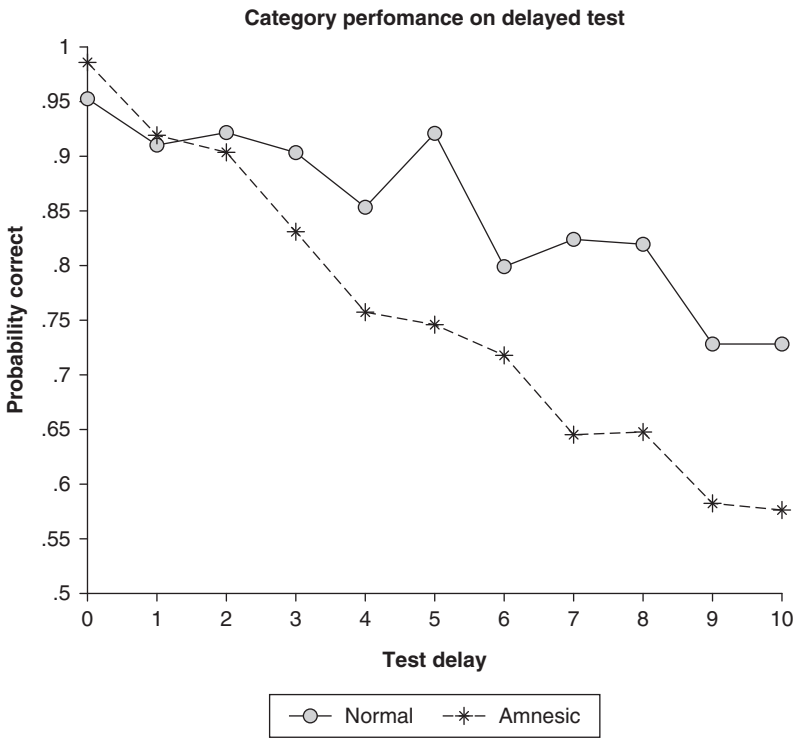

Figure 3.2 Performance for Two Individuals

One important advantage of the individual modeling approach is that it allows one to determine which model best fits each person from a set of competing cognitive models. In other words, this method allows for individual differences in the best-fitting type of model, and the percentage of individuals best fit by each cognitive model can be examined. Obviously, another advantage of the individual modeling approach is that it allows for any type of individual differences in parameters. Using this approach, one can estimate the distribution of parameters separately for the amnesic and control groups. Furthermore, one can compute the means of the parameters for each group as well as perform statistical tests on these means to determine whether the differences between groups are statistically significant. For example, after estimating the five sets of parameters for the normal and amnesic groups, we can compute the mean decay rate for each group and test whether the mean decay rate is significantly different for the two groups.

Hierarchical Modeling

There is a third approach, called the hierarchical data-fitting approach (see Lee, 2008; Rouder & Lu, 2005), which is a compromise between the first two. The idea is to fit a single probability mixture model to all the data from all the participants. The mixture model incorporates an extra, higher-level set of assumptions regarding the distribution of the parameters across individuals within each group. For example, to formulate a hierarchical version of the retention model, we would need to postulate a bivariate density function for each group that represents the distribution of the two model parameters within each group. Thus, we do not estimate the parameters for any individual, and instead we estimate the parameters of the bivariate density function of each group. This approach requires a large number of participants to obtain accurate estimates of the mixture density.

The hierarchical modeling approach has an advantage over the aggregate modeling approach because it allows for a distribution of parameters across individual differences. It also has an advantage over the individual modeling approach because it avoids fitting separate parameters to each person. If we assume that the hypothesized mixture density for the distribution of parameters is true, then the hierarchical model provides more precise estimates of the distribution of parameters as compared with the individual model fitting approach. However, if the wrong mixture density function is assumed for the distribution of parameters, then the hierarchical modeling approach could produce poorer estimates of the distribution of parameters than the individual modeling approach.

The hierarchical data-fitting approach is an attractive approach, but there remain some important advantages for the individual modeling approach. First, the latter does not require any extra assumptions about the distribution of parameters across individuals. Second, individual modeling allows one to compare the fits of the competing models separately for each person. A third advantage of the individual modeling approach is that it allows one to examine correlations between model parameters and other individual difference assessments. For example, after fitting the retention model to each participant from the amnesic group, one could examine the correlation between the retention parameter and brain images produced by functional MRI techniques.

Recommendations

In summary, the individual modeling approach is ideal for cognitive experiments involving a small number of participants and a large amount of

data per person. In this case, one can obtain precise estimates from each person, and no assumptions must be made about the distribution of parameters across individuals. The hierarchical approach is difficult to apply with a small number of participants because it is difficult to estimate the density function of the parameters from a small number of individuals. The hierarchical approach is ideal for studies with a large number of participants and a small amount of data per person. In this case, the parameter estimates obtained from the individual modeling approach are poorly estimated, and the density function of the parameters for the hierarchical approach can be estimated more precisely. Most cognitive modeling applications employ a small number of participants across a large number of trials, and so most of our examples will use the individual modeling approach. However, the last chapter in this book presents an example of using the hierarchical approach. The aggregate approach may be justified only when there is too little data per person to use the individual modeling approach and the model is too complex to use the hierarchical approach (Cohen et al., in press).

Objective Functions

There are various measures, or *objective functions*, that have been used to assess the fit of predictions to data. Objective functions map parameters into fit indices: For each combination of parameter values, the predictions are computed, and the fit to the data is measured. We will review the three most commonly used objective functions: the least-squares objective, the weighted least-squares objective, and the likelihood objective.

We will initially consider fitting the results from the normal participant shown in Figure 3.2. The data from this person are reproduced in Table 3.1. For example, at delay $d = 2$, the observed proportion for this person is $p_2 = .9204$.

The first step for all these objectives is to generate the predicted probabilities from a model for each delay condition using a specific set of parameter values. To illustrate the process, we will initially evaluate the predictions of

Table 3.1 Results for the Control Participant Shown in Figure 3.2

Delay	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Observed	.9538	.9107	.9204	.9029	.8515	.9197	.7970	.8228	.8191	.7277	.7276
Predicted	.9526	.9168	.8721	.8229	.7736	.7277	.6871	.6523	.6232	.5993	.5798

the retention model (using Equation 3.4) generated by setting $r = 3$ and $\gamma = .80$. These are not the optimal parameters, and they should be considered as only an initial guess. We will show how to find the optimal parameters later. Table 3.1 above also shows the predicted probabilities from Equation 3.4 with $r = 3$ and $\gamma = .80$. Considering the first delay condition, $d = 0$, the prediction is $P(0) = 1/\{1 + \exp[-3 \cdot (.80)^0]\} = .9526$, and the probability of an incorrect response is $1 - (.9526) = .0474$. Skipping up to the delay condition $d = 2$, the predicted probability is $P(2) = 1/\{1 + \exp[-3 \cdot (.80)^2]\} = .8721$, and the incorrect probability is $1 - .8721 = .1279$. Finally, moving down to the final delay condition, $d = 10$, the predicted probability is $P(10) = 1/\{1 + \exp[3 \cdot (.80)^{10}]\} = .5798$, and the probability incorrect is $1 - .5798 = .4202$.

Least-Squares Objective

Perhaps the most commonly used method for measuring fit is to sum the squared deviations between the observed and predicted values. For example, at delay $d = 2$, the observed proportion equals $p_2 = .9204$, the prediction for this condition was $P(2) = .8721$, and so the error is $[p_2 - P(2)] = (.9204 - .8721) = .0483$. Squaring this error produces $[p_2 - P(2)]^2 = (.0483)^2 = .0023$. This computation is performed on all 11 proportions to yield the sum of squared errors (*SSE*):

$$SSE = \Sigma[p_d - P(d)]^2,$$

which in this case is $SSE = 0.1695$.

One problem with the *SSE* measure of fit is that it penalizes all errors the same. However, some errors may be considered more serious than others, depending on the precision of the estimated proportion $p_d = (n_{Cd}/n)$, where n_{Cd} is the number correct and n is the total number of trials per condition. Suppose that the retention model is the true model. Then, the variance of the sample proportion is given by

$$\text{Var}[p_d] = \sigma^2(d) = \frac{P(d) \cdot Q(d)}{n},$$

where $Q(d) = [1 - P(d)]$. For example, $\sigma^2(0) = (.9526)(.0474)/200 = .00023$ and $\sigma^2(10) = (.5798)(.4202)/200 = .00122$. As can be seen from this formula, the variance is a quadratic function of probability: This quadratic has an inverted U-shaped form with a maximum at $P(d) = .50$ and a minimum at the extremes of 1.0 and 0.0. Based on this sampling distribution, we expect smaller estimation errors at short delay conditions, where the probability is closer

to 1.0, and we expect larger estimation errors at longer delay conditions, where the probability is closer to .50. Thus, errors at the longer delays should be given less weight because they are likely to be the result of estimation error. The least-squares criterion gives all errors equal weight. As we shall discuss later, this failure to take the variance of the estimation error into account causes the least-squares method to be an inefficient method for estimating parameters.

Weighted Least-Squares Objective

The next criterion is closely related to the least-squares method. The weighted least-squares method (denoted as *WSSE*) is computed from the squared deviations between the predicted and observed data points, but these deviations are weighted by the inverse of the variance of the prediction.

$$WSSE = \sum \left(\frac{[p_d - P(d)]}{\sigma_d} \right)^2 = \sum \frac{1}{\sigma_d^2} \cdot [p_d - P(d)]^2$$

which in this case is $WSSE = 158.4059$. The *WSSE* statistic is mathematically equivalent to the Pearson chi-square statistic:

$$\chi^2 = n \cdot \sum \left(\frac{[p_d - P(d)]^2}{P(d)} \right) + \left(\frac{[q_d - Q(d)]^2}{Q(d)} \right),$$

where $q_d = (1 - p_d)$ and $Q(d) = [1 - P(d)]$. The proof is simple if we examine each term being summed. First, we note that $[q_d - Q(d)]^2 = \{(1 - p_d) - [1 - P(d)]\}^2 = [p_d - P(d)]^2$. Then, using a common denominator, we combine the two terms within the Pearson chi-square statistic as follows:

$$\begin{aligned} & n \cdot \frac{Q(d)[p_d - P(d)]^2 + P(d)[q_d - Q(d)]^2}{P(d) \cdot Q(d)} \\ & \quad = n \cdot \frac{[Q(d) + P(d)] \cdot [p_d - P(d)]^2}{P(d) \cdot Q(d)} \\ & \quad = n \cdot \frac{1 \cdot [p_d - P(d)]^2}{P(d) \cdot Q(d)} = \frac{[p_d - P(d)]^2}{\sigma^2(d)}. \end{aligned}$$

Comparing the left- and the right-hand sides, we see that each term in the Pearson chi-square sum is exactly equal to each term in the *WSSE* sum. Thus, the two statistics are equivalent.

Likelihood Objective

The last method computes the likelihood that a model would have generated the observed data, given a fixed set of parameter values. Once again, we will illustrate this method using the retention model with the parameters set equal to $r = 3$ and $\gamma = .80$.

To compute this likelihood, it is useful to list all the 2,200 trials in the exact order of presentation. Part of these data is shown below in Table 3.2, which is from the same data that were summarized in Table 3.1. The first column indicates the trial number, the second column indicates the observed response on each trial (in this case, whether the choice was correct or incorrect), the next column indicates the experimental delay condition, and the last column shows the predicted probability of being correct for that person on that trial.

To compute the likelihood, we need to make an important assumption. We must assume that the response on any trial is *statistically independent* of

Table 3.2 List of Data That Were Summarized in Table 3.1

Trial	*Choice*	*Delay*	*Prediction*
1	C	0	.9526
2	I	0	.0474
3	C	0	.9526
4	C	0	.9526
5	C	0	.9526
. . .			
310	C	2	.8721
320	C	2	.8721
330	I	2	.1279
340	I	2	.1279
. . .			
2,190	C	10	.5798
2,200	I	10	.4202

the response on any other trial. A statistically independent sequence of binary responses is called a Bernoulli process. This assumption is probably false because of temporally extended factors such as fatigue, drifting attention, or response alternation tendencies; but it is often made for simplicity. Violations of statistical independence can be checked by statistically testing the autocorrelations at various lags, or testing the spectral density for deviations from white noise. However, for the purposes of this chapter, we will simply assume that the process is a Bernoulli process (in fact, it is true, because the data were artificially generated that way). This assumption is commonly made in cognitive modeling research, but caution about this assumption must always be kept in mind.[1] In a later chapter, we will relax this assumption and allow for statistical dependencies across trials.

Assuming statistical independence across trials, the joint probability of the responses across all trials is simply the product of the probabilities from each trial. (A general form for the likelihood function is given in the appendix to this chapter.) Referring to Table 3.2, the likelihood for the retention model, denoted as L_R, is computed by multiplying all the probabilities under the predicted column:

$$\begin{aligned} L_R = {} & (.9526)(.0474)(.9526)(.9526)(.9526)\ldots(.8721) \\ & (.8721)(.1279)(.1279)\ldots(.5798)(.4202). \end{aligned}$$

As you might expect, this product turns out to be a very small number, and so it is more convenient to work on the log scale by taking the natural logarithm of the product, which is called the log likelihood. (Recall that the log of a product equals the sum of the logs.) In this case, we obtain

$$\begin{aligned} \ln(L_R) = {} & \ln(.9526) + \ln(.0474) + \ln(.9526) + \ln(.9526) + \ln(.9526) + \ldots \\ & + \ln(.8721) + \ln(.8721) + \ln(.1274) + \ln(.1274) + \ldots \\ & + \ln(.5798) + \ln(.4202) \\ = {} & -969.9514. \end{aligned}$$

We can simplify this log-likelihood expression by counting the number of correct responses for each condition, n_{Cd}, and counting the number of incorrect responses for each condition, n_{Id}. By combining all the terms contributed by the correct responses for each condition and also combining all the terms contributed by the incorrect responses to each condition, we obtain a much shorter formula:

$$\ln(L_R) = \sum_{d=1,\ldots,11} n_{C,d} \cdot \ln[P(d)] + n_{I,d} \cdot \ln[Q(d)] = -969.9514. \quad (3.5a)$$

This is the standard formula for the log likelihood of a Bernoulli process, and it is closely related to the formula for the binomial distribution.[2]

It is difficult to judge whether this likelihood produced by this model is good or bad without having some idea about the maximum that can possibly be obtained for any model and any set of parameters. Assuming statistically independent observations, this maximum can be determined by evaluating the log likelihood of the saturated model. Recall that the saturated model simply uses the observed relative frequencies as the prediction for each delay condition:

$$\ln(L_S) = \sum n_{C,d} \cdot \ln[p_d] + n_{Id} \cdot \ln[q_d]. \tag{3.5b}$$

Using the observed relative frequencies shown in Table 3.1 to compute the log likelihood for the saturated model produces the result $\ln(L_S) = -879.9013$. Note that the saturated model has higher log likelihood as compared with that produced by the retention model. This must always be true.

When maximum likelihood is used to estimate parameters, the lack of fit is usually measured by a statistic called G^2. This is computed as follows. First, the difference between the log likelihoods of the saturated model and the retention model is computed: $\ln(L_S) - \ln(L_R) = (-879.9013) - (-969.9514) = 90.0501$. This difference is called the log-likelihood ratio because $\ln(L_S/L_R) = \ln(L_S) - \ln(L_R)$. The G^2 statistic is defined as twice the difference in the log-likelihood ratio:

$$G^2 = 2[\ln(L_S) - \ln(L_R)], \tag{3.6}$$

and in this case $G^2 = (2)(90.0501) = 180.1002$. This measures the lack of fit between the cognitive model and the saturated model. The parameters that maximize the likelihood objective are equivalent to the parameters that minimize G^2. Hereafter, when we refer to the maximum-likelihood objective, we will actually be minimizing G^2.

Relations Among Objectives

Here, we briefly point out some of the mathematical relationships among the three objective functions that we have reviewed. First, consider least squares and weighted least squares. If the variance does not change across conditions so that the weight is constant, then the weighted least-squares objective is equivalent to the least-squares objective. For example, linear regression models usually assume homogeneous variance (constant variance across conditions), and in that case the two objectives are identical. But as

we noted earlier, choice probabilities do not satisfy the homogeneous variance assumption. Neither does choice response time—the variance of response time for each condition tends to increase with the mean response time for each condition. Thus, it is unsafe to assume homogenous variance across conditions. (However, in the appendix, we present a method for transforming the dependent variable so that a weighted least-squares problem can be solved using least-squares methods.)

Next, consider the relation between weighted least squares and maximum likelihood. If we assume that the model is true, then the G^2 statistic, which is used with the maximum-likelihood objective, has the same asymptotic distribution as the *WSSE* statistic—both are asymptotically chi-square distributed (Rao, 1965). Furthermore, these two methods produce parameter estimates that have the same asymptotic normal distribution (see the appendix for more details about this point). Therefore, with large sample sizes, these two objectives produce very similar results.

Finally, consider the relation between least squares and maximum likelihood. This relation depends on the performance measure. In the present case, we are considering proportion correct as the performance measure, which has a binomial distribution. However, if the performance measure was normally distributed, and we assumed homogenous variance across conditions, then G^2 is linearly related to the sum of squared errors. Under these special conditions, minimizing sum of squared error is equivalent to maximizing the likelihood function (see the appendix).

Efficiency of Estimators

We have presented three methods for estimating parameters, and this naturally leads one to a question: How do we choose an objective for estimating parameters? This is usually decided on the basis of the statistical properties of the estimates produced by each method. There are two important properties of estimators: consistency and efficiency. Suppose the model we are fitting is the true model that actually generated the data. Then, the parameter values that minimize lack of fit to the sample data are sample estimates of the true parameters. A method for estimating parameters is consistent if the parameter estimates that it produces converge (in distribution) to the true values as the sample size for each condition increases to infinity. The method is efficient if, among all the consistent estimators, it produces estimates that have minimum variance. All three methods generally satisfy the consistency property, so this does not lead to a basis for preference. However, the methods differ with respect to the efficiency property. If homogeneity of variance is not satisfied, then the least-squares method is not efficient. This is an important reason for using either the weighted least

squares or the maximum-likelihood objectives. Mathematically, it has been proven that parameter estimates obtained from minimizing weighted least squares or maximum likelihood are consistent and efficient (Rao, 1965).

Searching for Optimal Parameters

Recall that the retention model produced a $G^2 = 180.1002$ using the specific parameter values $\gamma = .8$ and $r = 3$. But this is not necessarily the best or worst choice. In fact, if we set $\gamma = .8$ and $r = 2$ we obtain a much worse fit, $G^2 = 359.913$; and we shall see later that the optimal parameters are obtained by setting $\gamma = .9042$ and $r = 2.9166$, which produces a $G^2 = 15.8077$. How we found this will be discussed below.

Grid Search

To glean some idea about how computer search programs work, we will analyze the search problem in a bit more detail. Consider forming a two-dimensional grid by crossing 151 values for γ (.80, .51, .52, . . . , .95) with 201 values for r (2, 1.01, 1.02, . . . , 4.0) producing $151 \cdot 201 = 30{,}351$ points on the grid. Suppose we compute a G^2 fit statistic for the data from Table 3.2 at each of these points on the grid, and then plot the G^2 values above each point on the grid. This produces the response surface plot shown in Figure 3.3. The left horizontal axis represents the 201 values of the association strength parameter r; the right horizontal axis represents the 151 values of the decay parameter γ; and the vertical axis represents the value of G^2 above each grid point. The surface produced by this looks like a curved sheet of paper. Our task is to find the grid point that lies at the bottom of this curved sheet. If we start from one of the corners and move step by step in the direction downhill, we will eventually reach the bottom where a move in any direction takes us back uphill. This is the minimum point for which we are looking, and in this case, it happens to be located at ($\gamma = .9042$, $r = 2.9166$). This is called hill climbing (downhill).

The curves shown in the horizontal plane at the bottom of the response surface are contour curves. Each contour curve indicates the set of grid points that produce exactly the same G^2 lack of fit. The outer contour curves show the points that produce a large G^2 or poor lack of fit, and the inner contour curves that surround the minimum point produce small G^2 or good fit. Notice that the inner contour curve that surrounds the minimum is elliptical with the major axis aligned with the negative correlation between the strength and decay parameters. This shows that changes in the parameters along this line produce only small changes in fit.

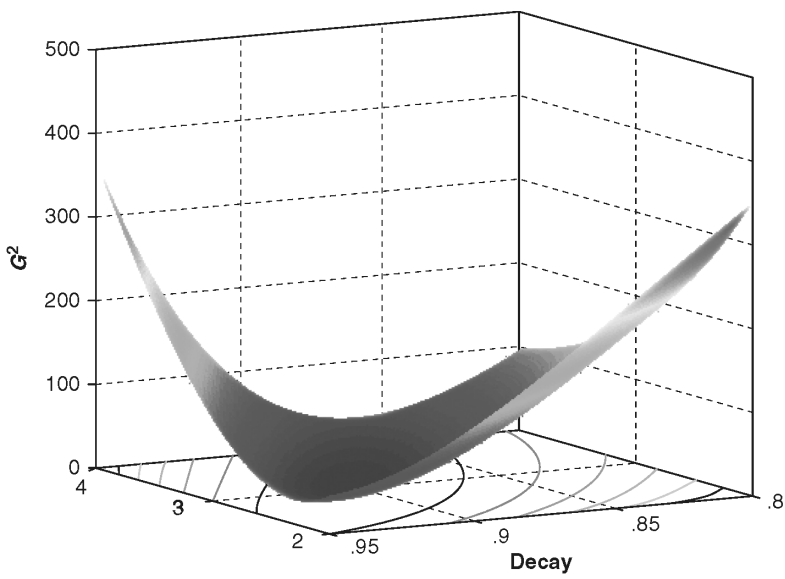

Figure 3.3 Response Surface Analysis

The graphical analysis described above is not really practical in most model-fitting applications. There are two reasons for this. One is that the graphical analysis can only be used with one or two parameters and the other is that most applications have more than two parameters. A grid search can be used with any number of parameters to find a rough idea of where the maximum is located, but this solution is usually too crude, and more precise estimates are usually required. Grid searches are useful for finding a starting point for a more precise analysis. Parameter optimization for models with many parameters requires the use of sophisticated search processes on a computer. Mathematical programming languages such as MATLAB, Mathematica, GAUSS, or SAS provide programs for performing search of parameters for nonlinear models.

Below we sketch the basic idea of a steepest-descent algorithm.

Steepest-Descent Search

Computer search programs are designed to find the parameters that produce the minimum of an objective function, G^2 in our case. Some of these programs use a form of steepest descent, which can be roughly described as follows. We can start at any point, say, for example, the point ($\gamma = .8$, $r = 4$) in the back corner of Figure 3.3. From here, we can consider moving a small

step to a nearby grid point, say the point ($\gamma = .801$, $r = 3.99$). The line connecting these two points forms a direction. If we compute the change in the G^2 statistic produced by moving in this direction, we would find that it produces a decrease in G^2. But this may not be the direction producing the maximum decrease. Alternatively, we could consider moving a small step to another adjacent grid point such as ($\gamma = .8$, $r = 3.99$) or ($\gamma = .801$, $r = 4$), and compute the change in G^2 produced by this direction. By checking all the possible small moves at adjacent grid points in every direction, we could find the move that produces the largest decrease in G^2, which is the direction of steepest descent. The program takes a very small step in the direction of steepest descent. After making this move to this new position, we have a new set of parameters that are better than the starting values. This completes the first iteration of the search process, and the whole procedure is repeated. For the second iteration, we start from the current point after the first iteration, find the direction of steepest-descent from this new point, and take a second small step in the direction of the gradient at this point to produce the second position after the second iteration. This process continues until we reach a point where a move in any direction fails to decrease the objective, our G^2 statistic. At this minimum point, the surface becomes perfectly flat, and all directions lead uphill (which is at $\gamma = .9042$ and $r = 2.9166$ in this case).

Mathematically, the direction of steepest descent is found by computing the *gradient* of the objective function at current point of search in the parameter space. The gradient is equal to the partial derivative of the objective function with respect to each parameter. Steepest-descent search programs move downhill in the direction of the gradient until the gradient reaches zero. Steepest-descent programs do not require the user to supply the gradient; instead the program uses finite difference methods to estimate the gradient automatically for the user. In the appendix to this chapter, we provide more details about the steepest-descent search methods, as well as an example program.

Constraints on Parameters

Often the parameter estimates for a cognitive model need to be constrained so that they fall within theoretical boundaries. For example, the decay rate should range from 0 (complete forgetting) to 1 (no forgetting), and the initial strength should be nonnegative. Sometimes, however, an unconstrained search algorithm will explore parameter values that fall outside the boundaries. When this happens, it may be a signal that the model is not fitting very well or the data are too noisy. In this case, it may be necessary to force the parameter search to stay within the theoretical boundaries. There are a couple of ways to do this.

Constrained steepest-descent programs are designed to satisfy what are known as the Kuhn Tucker conditions, which are necessary conditions for optimality under nonlinear constraints. These programs incorporate the information about the Kuhn Tucker conditions directly into a modification of the objective function, called the Lagrangian function, and the search is based on this modified objective. This method provides the most effective way to deal with the parameter constraint problem.

Other search programs (described below) cannot incorporate information about the Kuhn Tucker conditions into the search process. These alternative search algorithms require another method for incorporating constraints. Another way to do this is to reparameterize the model using new parameters that have no constraints. For example, if we wish to impose the constraint $r > 0$, we could define the initial strength as $r = e^{u}$, and search for the parameter u instead of r; and if we wish to constrain the decay rate γ to range from 0 to 1, then we could define the decay rate as $\gamma = 1/(1 + e^{-v})$, and estimate v instead of γ. Using this method, the search program searches for the pair of parameters (u, v), which are then used to compute the two cognitive parameters r and γ, and finally, the latter are inserted into the retention model to make predictions. Although this method of imposing constraints works, it is less than ideal because it adds extra complications to the search process.

Flat Minimum Problem

Several problems can occur with the parameter search process. If the response surface is very flat, then the steepest-descent search process may terminate prematurely because the changes in the objective function are too small to detect improvements. Notice in Figure 3.3 that the surface is steep along the line formed by connecting the left corner point (.95, 4) with the front corner point (.95, 2). Changes in parameters along this line produce a very large change in the lack-of-fit measure. But the surface is very flat along the line connecting (.95, 2) to (.90, 3). In other words, near the minimum point, decreasing the decay parameter γ can be compensated by increases in the strength parameter r to produce almost the same fit. If we start at the point where the function is the minimum, and move along this negatively correlated line where the function is flat, then substantial changes in parameter values produce small changes in the lack of fit. This insensitivity of the objective function to changes in parameter values causes instability in the parameter estimates. More precisely, the curvature of the objective function near minimum determines the variance of the parameter estimates. Flatness near the minimum produces parameter estimates with large standard errors.

Local Minima Problem

There is an important limitation with steepest-descent search algorithms called the local minimum problem. Note that the objective function plotted in Figure 3.3 has only one minimum. In fact, the surface can be well approximated by a bowl-shaped quadratic function near the minimum. This is a desirable situation but there is no guarantee that this will occur. Consider, for example, the search for a maximum problem illustrated in Figure 3.4. In this example, a grid of points for a single parameter is plotted on the horizontal axis, and the objective function is plotted on the vertical axis. If the steepest-ascent search program began its search on the far left side, then it would succeed in finding the global maximum, which is the desired point. However, if the program started on the far right side, then it would get stuck at the local maximum and never find the global maximum. The only way to avoid getting stuck at the local maximum in this case is to try out several starting positions.

As the number of parameters increases, the possibility of more than one local minimum increases and this becomes a serious problem. When several local minima are possible, it is important to try out a grid of starting positions

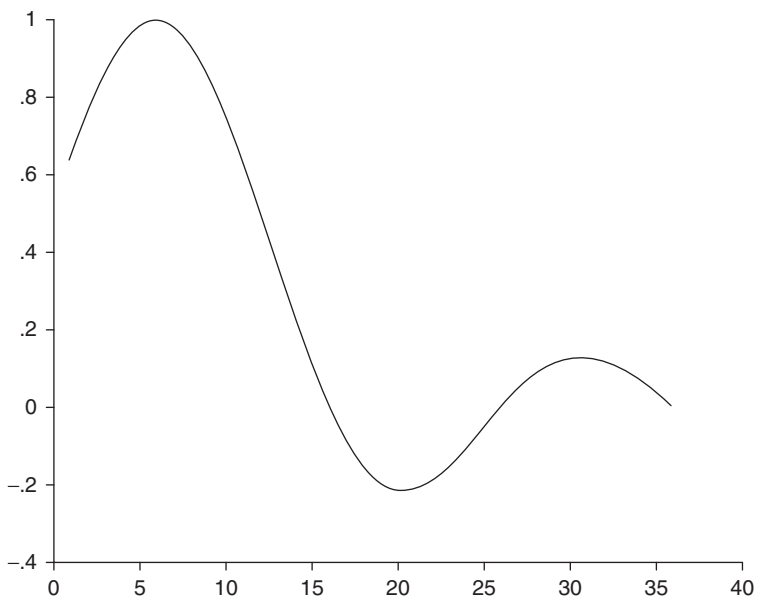

Figure 3.4 Example of a Local Maximum

or a random sample of starting positions. If two different starting positions produce different search results, then of course, you choose the search result that yields the smallest value for our objective function. But this would also serve as a warning that local minima are a problem.

For very difficult search problems, such as highly nonlinear problems involving a large number of parameters, many local minima may be present, and then it may be advantageous to use a stochastic search called simulated annealing to avoid getting stuck in a local minimum (Ingber, 1993; Kirkpatrick, Gelatt, & Vecchi, 1983). Briefly, the simulating annealing algorithm works as follows. From a given starting position, the program randomly selects a new position and evaluates this new position compared with the old position. With probability p, the algorithm selects the superior of these two points, and with probability $(1 - p)$ it selects the inferior. This procedure is repeated for many iterations; however, with each iteration the value of p gradually increases. The rationale behind this method is the following. Early in the search, the procedure bounces up and down a lot so that it can escape out of a local minimum; but later in the search, the procedure converges toward the global minimum.

Discontinuities

Steepest-descent algorithms require computing the gradient, and this is only feasible for objective functions that are smooth and continuous. When discontinuities, such as breaks, jumps, or kinks, exist in the objective function, then steepest-descent is no longer applicable, and some other non-derivative-based search process must be employed.

The most commonly used direct search method is the Nelder-Mead simplex algorithm (Nelder & Mead, 1965). To briefly describe this algorithm, consider the problem of searching the parameter space for the best-fitting two parameters of the retention model. The parameter space is represented by the rectangular horizontal plane shown in Figure 3.3. Each point on this plane represents a pair of parameters. This program starts by taking three points in this parameter space (three pairs of parameters), forming a small triangle (called the simplex), and evaluating these three points with respect to the objective function. Then, a new fourth point is generated, either within or nearby the edge of the triangle, from the original three points. For example, the new point may be formed by averaging the three points, producing a fourth point. Finally, the best three out of these four points is retained, and the worst point is dropped. This procedure is repeated until no new point can be generated that improves the objective function.

Many cognitive models encounter this problem of discontinuity. In particular, this problem occurs whenever the predictions of the model involve random elements. For example, a neural network model may use randomly selected initial connection weights prior to category learning. If this were the case, then the predictions from this neural network model would vary randomly even when the model parameters are held constant. If the predictions vary randomly, then so does the measure of fit. The result is that the objective function is no longer a simple function of the parameters. Even when the model parameters are held constant, the objective function will change abruptly due to the random elements. At one stage of the search process, one point in the parameter space may produce a better fit than another, but at the next stage, the order could reverse simply due to the random elements. For these types of models, involving random elements, derivative-based search methods cannot work, and a direct search algorithm is required.

Discrete-Valued Parameters

Some cognitive models involve integer- as well as continuous-valued parameters (e.g., the number of items that can be stored in a short-term store). One way to deal with this type of problem is to use a steepest-descent algorithm to minimize the continuously valued parameters for a specific value of the integer parameter. Then, repeat this process for each value of the discrete parameter producing a finite set of search results. Finally, select the discrete parameter value that results in the minimum of all these searches. However, if there are very many values of the discrete parameters to check, then this method becomes too difficult to employ.

Under these conditions, it may be advantageous to use a genetic algorithm to search for the optimal combination of discrete and continuous parameters. Genetic algorithms were invented by John Holland (1975) and are based on concepts from biological evolution. This method maintains a population of candidates (called genes), where each candidate is a binary coded representation of a combination of parameter values. Then genetic principles of random crossovers and mutations are used to generate new candidates, and evolutionary principles of fitness and selection are used for reproduction of candidates.

Results From Each of the Estimation Methods

What happens when we use different objective functions to estimate the parameters for the same data shown in Table 3.1? Will we obtain the same

parameter estimates? In general, the answer is no. To illustrate, we used a modified steepest-descent algorithm to minimize *SSE*, *WSSE*, and G^2 for the data shown in Figure 3.2, separately for the normal participant (see Table 3.3a) and the amnesic participant (see Table 3.3b). This table shows the solutions for the best-fitting parameters produced by each method. As can be seen in this table, although the results are not exactly the same, they turn out to be very similar in this case. All three methods produce an estimate of the decay rate that is faster for the amnesic participant as compared with the normal participant. However, there is no guarantee that the results will always turn out this similar.

The parameters that minimized the G^2 criterion were found for all 5 participants from each of the two groups, and the results, averaged across participants, are shown in Table 3.4. Each cell shows the mean for each group, and the standard deviation is shown in parentheses. Once again, we observe a faster decay rate for the amnesic group. These data were artificially generated from the retention model using the following population parameters: ($\gamma = .90$, $r = 3.0$) for the normal population and ($\gamma = .80$, $r = 3.2$) for the amnesic population. Thus, the estimation procedure did a good job of recovering the true parameter values in this example.

Table 3.3a Parameter Estimates Obtained From the Normal Person Shown in Figure 3.2

	Decay, γ	*Strength, r*	*Fit Index*
SSE	.8997	3.0046	*SSE* = 0.0095
WSSE	.9040	2.8863	*WSSE* = 14.81
Likelihood	.9042	2.9166	G^2 = 15.8077

Table 3.3b Parameter Estimates Obtained From the Amnesic Person Shown in Figure 3.2

	Decay, γ	*Strength, r*	*Fit Index*
SSE	0.7902	3.3222	*SSE* = 0.0031
WSSE	0.7850	3.4042	*WSSE* = 5.4033
Likelihood	0.7820	3.4744	G^2 = 5.6596

Table 3.4 Means and Standard Deviations Averaged Across Participants Within Each Group

Group	*Mean Decay*	*Mean Strength*	*Mean* G^2
Normal ($N = 5$)	.9022 (.0135)	2.9373 (.1898)	10.8018 (5.0029)
Amnesic ($N = 5$)	.8062 (.0171)	3.1550 (.2145)	7.5540 (2.1606)

Variance of the Parameter Estimates

The standard deviations shown in Table 3.4 represent the variation in the estimated parameters across individuals within each group. There appears to be more variation in the strength parameter as compared with the decay parameter. It is important to note, however, that the variance of each parameter is influenced by two sources: One is variation caused by individual differences in the parameter across people within the same group; but another source is estimation error variance caused by noisy sample data.

In this example, the data were actually simulated with no variance in the true parameters across individuals within a group—all 5 participants within each group were generated using exactly the same parameter values. Therefore, the variation in parameter estimates across individuals for this artificial example is entirely attributed to estimation error. The standard error for the strength parameter is larger than that for the decay parameter.

It is possible to compute the standard errors of the parameters for each individual when steepest-descent algorithms are used with the weighted least squares or likelihood objectives. At the completion of the parameter search for the minimum, the steepest-descent algorithms can compute a matrix called the inverse Hessian matrix (see the appendix to this chapter for more details about this matrix). For large sample sizes, the element located in the ith diagonal position of this matrix provides an estimate of the variance of the ith parameter, and the square root of this element gives the standard error of this parameter. For example, the retention model has two parameters, and so the inverse Hessian is a 2×2 matrix with two diagonal elements. The first diagonal element provides an estimate of the variance of the decay rate, and the second diagonal element provides an estimate of the variance of the initial strength. In particular, for the normal participant shown in Table 3.1, the inverse Hessian matrix is

$$\begin{pmatrix} .0001 & -.0008 \\ -.0008 & .0160 \end{pmatrix}.$$

Therefore, an estimate of the standard error for this person on the decay rate equals $(.0001)^{.5} = .01$, and an estimate of the standard error for the initial strength is $(.0160)^{.5} = .1265$. These results reflect the sample estimates of the standard deviations (based on the estimate from 5 participants) shown in Table 3.4.

The off-diagonal elements of the inverse Hessian matrix are also informative. These indicate the covariance between a pair of parameters. For example, using the data from the normal participant shown in Table 3.1, the estimate of the covariance between the parameters equals –.0008, and the corresponding correlation is (–.0008)/(.01)(.1265) = –.8187, which is highly negative. This negative covariance reflects the parameter trade-off that we observed near the minimum in Figure 3.3.

For large sample sizes, when the weighted least squares or the likelihood objectives are used, the parameters are asymptotically normally distributed. Therefore, one can compute 95% confidence interval estimates of the parameters using a *z* table. Consider, for example, the data from the amnesic participant shown in Table 3.3b. The estimate of the decay rate for this person shown in Table 3.3b is .782. Using the methods discussed above, we find that the standard error of this estimate equals .1818. Finally, the 95% confidence interval is .782 ± (1.96)(.1818), which produces the interval [.4257, 1.1383]. The upper bound is too high, because it does not make sense to have decay rates greater than 1. A Bayesian method (see Chapter 6) that builds in this prior probability would produce a better estimate in this case.

Model Evaluation

Chi-Square Lack-of-Fit Test

Both the weighted least squares and the maximum-likelihood methods provide a statistical test of deviations between the cognitive model and the saturated model. If the sample size is sufficiently large, then both statistics, χ^2 and G^2, are approximately chi-square distributed. If we wish to test the null hypothesis of no difference between the cognitive model and the saturated model, then we can refer the computed χ^2 or G^2 statistic to a standard chi-square table with *df* equal to the number of data points minus the number of parameters in the cognitive model. If the statistic exceeds the table value at a specified significance level, then the null hypothesis can be rejected, which means that there are some significant deviations in fit of the model to the data.

The sample size was $n = 200$ per condition for each person, which is fairly large, and so these chi-square tests are reasonable in this case. The degrees

of freedom for this problem is $df = 11 - 2 = 9$ (11 data points minus 2 parameters), and the table chi-square at the .05 significance level is 16.92. Comparing this with the *WSSE* and G^2 statistics shown in Table 3.3, we fail to reject the null hypothesis. In other words, we conclude that there are no statistically significant deviations from the model predictions. (This is the correct decision in this case, because the data were in fact artificially generated from the model.)

These statistical tests of lack of fit are of limited scientific use for the following reasons. First, we know a priori that our cognitive model is imperfect, and so we are bound to have deviations between it and the real cognitive system generating the empirical data. Second, statistical tests of lack of fit are only valid for large sample sizes, and as the sample size gets large, the power to reject the incorrect null hypothesis increases toward 1.0, even on the basis of small deviations. Therefore, with a sufficiently large sample size, we are almost guaranteed to reject the null hypothesis as a result of deviations that are certain to exist between the model and the true generating process. In short, failure to reject the null hypothesis only tells us that the sample size is too small to detect the imperfections that must exist, and rejection of the null hypothesis only tells us that the model is imperfect, which we knew from the beginning.

R^2 Index of Model Fit

It is difficult to judge whether the *SSE* statistic is good or bad, so we need to compare it with that produced by the saturated and null models. The sum of squared errors produced by the saturated model is obviously zero, because it exactly reproduces the observed choice proportions. The sum of squared errors produced by the null model is equal to the sum of squared deviations around the mean (denoted as *TSS*). An index of fit, called R^2, is defined as

$$R^2 = 1 - (SSE/TSS) = TSS/TSS - SSE/TSS = (TSS - SSE)/TSS = SSP/TSS,$$

where *SSE* is the sum of squared *errors* produced by the cognitive model, and *SSP* is the sum of squares *predicted* by the model. Thus, R^2 is the ratio of the predicted sum of squares to the total sum of squares, that is, the proportion predicted by the model. If $R^2 = 0$, the cognitive model is no better than the null model, and if $R^2 = 1$, the cognitive model is equal to the saturated model.[3]

The mean for the normal participant (the data shown in Table 3.1), equals 0.8503, which produces a $TSS = 0.0615$; the *SSE* for this individual (shown

in Table 3.3a) is $SSE = 0.0095$. Therefore, $R^2 = 1 - (.0095/.0615) = .8455$, which is a mediocre fit. For the amnesic participant, the $R^2 = 1 - (.0031/.0615) = .9841$, which is much better. Note that *SSE* is a measure of badness of fit, whereas R^2 is a measure of goodness of fit. These two indices are linearly related to each other, so whatever parameters happen to minimize *SSE* will also maximize R^2.

Aggregate Versus Individual Modeling

To illustrate some of the problems that can be encountered by fitting average rather than individual data, two sets of predictions were generated from the retention model. One set was generated by setting $\gamma = .80$ and $r = 3.00$ (with no sampling error), which we will call the Person A data set; a second set was generated by setting $\gamma = .40$ and $r = 3.20$ (with no sampling error), which we will call the Person B data set. A third data set was constructed from the first two by averaging the corresponding values of the first two data sets, which we will call the Average Person data set. All three data sets are graphed in Figure 3.5 below, with Person A as the top curve, Person B as the bottom curve, and the Average Person is the middle curve.

The retention model was fit to all three data sets, solving for γ and r that minimized the *SSE* criterion.[4] The retention model perfectly fits the Person A data set with $R^2 = 1.0$; it also perfectly fits the Person B data set with $R^2 = 1.0$. This is exactly what we expect because this model generated these data. However, it did not generate the Average Person data set, and it cannot perfectly fit these data. In this third case, we find $R^2 = .9756$. This failure to fit the Average Person data is a direct consequence of the nonlinearity of the model.

Suppose we try fitting another model, called the power function model, to these same three data sets:

$$P(d) = r \cdot (d + 1)^{\gamma}.$$

This model also has two parameters, γ and r, that must be estimated from each data set. The power function model was fit to all three data sets, solving for γ and r that minimized the *SSE* criterion. The power function model cannot perfectly fit the Person A data set, and we find $R^2 = .8938$ for this case. It cannot perfectly fit the Person B data set either, and we find $R^2 = .9335$ for this case. Both these results are expected because the two individual data sets were not generated by the power function—they were generated by the retention model. However, when we fit the power function

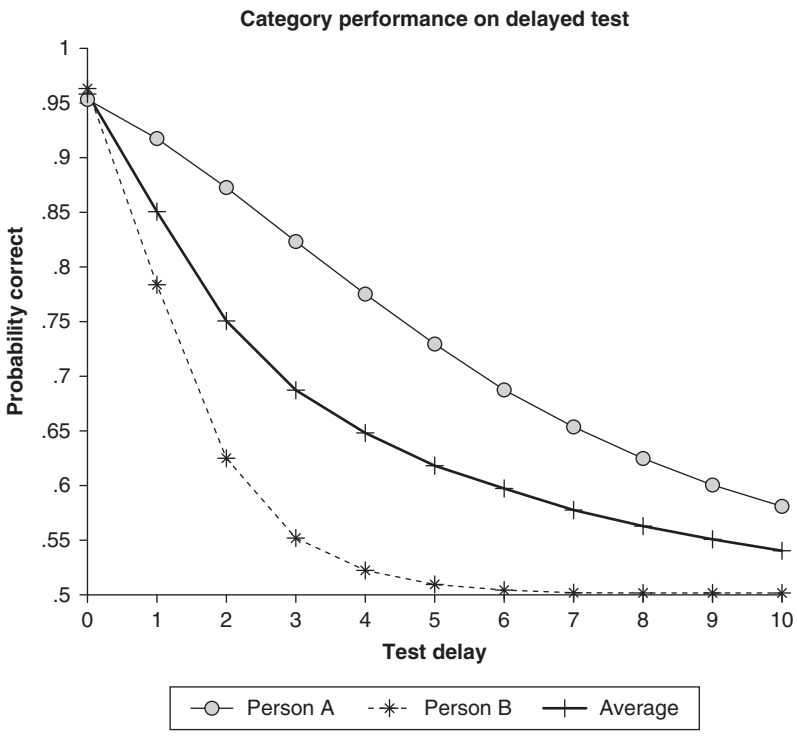

Figure 3.5 Two Fictitious Data Sets and Their Average

to the Average Person data set, we find $R^2 = .9924$, almost a perfect fit, and clearly superior to the retention model.

In this example, the retention model is the true model, and the power model is the false model, for the two individual data sets. When we compare the true model with the false model using the average data, the false model fits better. But if we compare the models using the individual data sets, the true model fits better in both cases. This example was based on earlier analyses of this problem by Myung, Kim, and Pitt (2000). This illustrates how aggregating data can mislead researchers, and it points out the importance of comparing models at the individual level of analysis.

Conclusion

Often a researcher wishes to compare the quantitative predictions between two cognitive models. In this case, it is necessary to obtain optimal estimates

of the parameters from both models to give each model the best chance of predicting the data. We don't want to reject a model simply because we arbitrarily chose poor parameters. Furthermore, the parameter estimates themselves can be of great interest to the researcher, because they provide measures of important underlying cognitive processes. For example, in a memory retention model, we may be interested in estimating the difference in the decay rate parameter for amnesic and control subjects. We cannot extract this decay rate by simply looking at the raw data. Instead, we need to estimate the decay rate using the cognitive model. Cognitive models are usually nonlinear and this makes the parameter estimation process more complex than it is for linear models such as a regression model. However, with the availability of nonlinear search programs commonly available in mathematical programming languages, this process is becoming easier and more practical.

Appendix

This appendix provides more technical details about several of the topics concerning nonlinear parameter estimation. First, we present more general statements of both the least squares and the likelihood objectives. Then, we describe two of the most commonly used steepest-descent types of algorithms: the quasi–Newton-Raphson algorithm and the modified Gauss-Newton algorithm.

Generalized Least Squares

Weighted and unweighted least-squares methods require one to write the cognitive model as a nonlinear function, here denoted as M, that maps the independent variables and the parameters into a set of predictions. Define $\mathbf{Y}$ as a column vector of observations, $\mathbf{X}$ is a matrix of known independent variables, $\boldsymbol{\theta}$ is a vector of parameters, and $\mathbf{E}$ is a column vector of errors. Mathematically, the cognitive model is represented by the following nonlinear model:

$$\mathbf{Y} = M(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{E},$$

where $\text{Var}[\mathbf{E}] = \boldsymbol{\Sigma}$ is the variance-covariance matrix of the errors.

Parameters that minimize the least-squares criterion, $SSE = \mathbf{E}^{\mathrm{T}}\mathbf{E}$, are efficient if we assume that $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. When this assumption is not valid, we need

to find parameters that minimize $WSSE = \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{E}$ to obtain efficient estimates. However, any weighted least-squares minimization problem can be linearly transformed into a least-squares minimization problem as follows.

First, we express the variance-covariance matrix of $\mathbf{E}$ in terms of its eigenvectors and eigenvalues: $\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T = \mathbf{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\Lambda}^{1/2}\mathbf{V}^T$, where $\mathbf{V}$ is the orthonormal matrix of eigenvectors of $\boldsymbol{\Sigma}$, $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues of $\boldsymbol{\Sigma}$, and $\boldsymbol{\Lambda}^{1/2}$ contains the square roots of the eigenvalues. The inverse also can be expressed in terms of eigenvectors and eigenvalues as

$$\boldsymbol{\Sigma}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^T = (\mathbf{V}\boldsymbol{\Lambda}^{-1/2})(\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T),$$

where $\boldsymbol{\Lambda}^{-1/2}$ is the inverse of $\boldsymbol{\Lambda}^{1/2}$. Then, the weighted sums of squares can be rewritten as

$$WSSE = \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{E} = \mathbf{E}^T\mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^T\mathbf{E} = (\mathbf{E}^T\mathbf{V}\boldsymbol{\Lambda}^{-1/2})(\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T\mathbf{E}) = (\mathbf{E}^*)^T\mathbf{E}^*,$$

where $\mathbf{E}^* = (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)\mathbf{E}$. Note that the variance-covariance matrix of $\mathbf{E}^*$ is

$$\mathrm{Var}[\mathbf{E}^*] = (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)\mathrm{Var}[\mathbf{E}](\mathbf{V}\boldsymbol{\Lambda}^{-1/2}) = \boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T \cdot \mathbf{V}\boldsymbol{\Lambda}^{1/2}\boldsymbol{\Lambda}^{1/2}\mathbf{V}^T \cdot \mathbf{V}\boldsymbol{\Lambda}^{-1/2} = \mathbf{I},$$

and so $\mathbf{E}^*$ satisfies the assumption required for least squares to produce efficient estimates. Now, we replace the original model with a new model:

$$\begin{aligned}\mathbf{Y}^* &= (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)\mathbf{Y} = (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)M(\mathbf{X}, \boldsymbol{\theta}) + (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)\mathbf{E} \\ &= M^*(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{E}^*,\end{aligned} \tag{A3.1}$$

where $\mathbf{Y}^* = (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)\mathbf{Y}$ and $M^*(\mathbf{X}, \boldsymbol{\theta}) = (\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)M(\mathbf{X}, \boldsymbol{\theta})$. In essence, we estimate the parameters that minimize the sum of squared error for the new model $\mathbf{Y}^* = M^*(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{E}^*$. Finding the parameter vector $\boldsymbol{\theta}$ that minimizes $SSE = (\mathbf{E}^*)^T\mathbf{E}^*$ also minimizes $WSSE = \mathbf{E}^T\boldsymbol{\Sigma}^{-1}\mathbf{E}$. Later, if we wish to recover the predictions on the original scale, we apply the inverse transformation

$$(\mathbf{V}\boldsymbol{\Lambda}^{1/2})M^*(\mathbf{X}, \boldsymbol{\theta}) = (\mathbf{V}\boldsymbol{\Lambda}^{1/2})(\boldsymbol{\Lambda}^{-1/2}\mathbf{V}^T)M(\mathbf{X}, \boldsymbol{\theta}) = M(\mathbf{X}, \boldsymbol{\theta}).$$

In summary, we can always transform a weighted least-squares problem into a least-squares problem using the transformation shown above. The parameters obtained by minimizing the *SSE* of the transformed dependent variable will be exactly the same as the parameters obtained by minimizing the *WSSE* of the original dependent variable. Furthermore, these parameters will have the same asymptotic distribution as the maximum-likelihood estimates.

The Generalized Likelihood Function

Suppose we observe a sequence of N ordered observations from an individual denoted as $(y_1, y_2, \ldots, y_t, \ldots, y_N)$, where each y_t has a discrete distribution.[5] In general, this sequence may be generated by any type of statistically dependent stochastic process. In this chapter, each observation indicated the choice made on each trial; but more generally, this could be any discrete measure, such as confidence ratings. The joint probability of these N observations is given by

$$\begin{aligned}&\Pr[y_1 \cap y_2 \cap \ldots \cap y_t \cap \ldots \cap y_N] \\ &= \Pr[y_1] \cdot \Pr[y_2|y_1] \cdot \Pr[y_3|y_1 \cap y_2] \cdot \ldots \\ &\cdot \Pr[y_N|y_1 \cap y_2 \cap \ldots \cap y_{N-1}].\end{aligned} \tag{A3.2}$$

For convenience, define $\Pr[y_t|t-1] = \Pr[y_t|y_1 \cap y_2 \cap \ldots \cap y_{t-1}]$ as the probability of observing the observation y_t given all the preceding observations. Then, the log-likelihood statistic is defined as the natural log of Equation A3.1:

$$\ln(L) = \sum \ln(\Pr[y_t|t-1]). \tag{A3.3}$$

If we assume statistical independence, then $\Pr[y_t|t-1] = \Pr[y_t]$, and the stochastic process reduces to what is called an independent process. The product rule for computing the likelihood is only valid for an independent process, which is a very specialized case of the more general likelihood function (see Anderson, 1971).

The G^2 statistic for comparing the cognitive model with the saturated model is defined as twice the difference in log likelihoods of the two models:

$$G^2 = 2[\ln(L_S) - \ln(L_R)]. \tag{A3.4}$$

The G^2 statistic is the primary measure used to compare two models when using the maximum-likelihood method. When we search for parameters that maximize the likelihood function, or minimize the G^2 function, then these parameters are called maximum-likelihood estimates.

Newton-Raphson Search

This search method is applicable to any of the three objective functions that are presented in this chapter (Fletcher, 1987). For generality, define F as the objective function, where we could set $F = SSE$, or $F = WSSE$, or $F = G^2$.

Define $\boldsymbol{\theta}$ as a column vector containing all the parameters of the cognitive model. For example, in the previous chapter, $\boldsymbol{\theta} = [\gamma\ r]^{\mathrm{T}}$, which is a 2×1 vector containing the decay rate and initial strength parameters. We insert $\boldsymbol{\theta}$ into the cognitive model to generate predictions, and then the objective function F evaluates these predictions. Mathematically, F is a function that maps $\boldsymbol{\theta}$ into a real-valued measure of fit. Our goal is to try to find an iterative scheme that changes the parameter vector at each step from $\boldsymbol{\theta}$ to $(\boldsymbol{\theta} + \boldsymbol{\delta})$ to produce a reduction in the objective function from $F(\boldsymbol{\theta})$ to $F(\boldsymbol{\theta} + \boldsymbol{\delta})$.

Originally, Isaac Newton invented a procedure based on the following idea. First, we approximate the objective function by a second-order Taylor series expansion:

$$F(\boldsymbol{\theta}+\boldsymbol{\delta}) = F(\boldsymbol{\theta}) + \boldsymbol{\delta}^{\mathrm{T}} \cdot \frac{\partial F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} + \frac{1}{2} \cdot \boldsymbol{\delta}^{\mathrm{T}} \cdot \frac{\partial^2 F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \cdot \partial \boldsymbol{\theta}^{\mathrm{T}}} \cdot \boldsymbol{\delta} + \text{residual}.$$

This quadratic approximation to the objective function is accurate as long as the change, $\boldsymbol{\delta}$, is small in magnitude. This expression has two terms involving $\boldsymbol{\delta}$. The first term contains the first-order partial derivative of $F(\boldsymbol{\theta})$ with respect to the parameter vector, denoted $\nabla = \dfrac{\partial F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ with $\nabla_i = \dfrac{\partial F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i}$, which is the gradient, and this provides the direction of steepest ascent (the negative of the gradient provides the direction of steepest descent). The second term contains the second-order partial derivative of $F(\boldsymbol{\theta})$ with respect to the parameter vector, denoted as $\mathbf{H} = \dfrac{\partial^2 F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \cdot \partial \boldsymbol{\theta}^{\mathrm{T}}}$ with $H_{ij} = \dfrac{\partial^2 F(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i \partial \boldsymbol{\theta}_j}$, which is called the Hessian matrix, and its importance will be discussed later. Using this new notation, we can rewrite the Taylor series expansion as

$$F(\boldsymbol{\theta} + \boldsymbol{\delta}) = F(\boldsymbol{\theta}) + \boldsymbol{\delta}^{\mathrm{T}} \cdot \nabla + (1/2)\boldsymbol{\delta}^{\mathrm{T}} \cdot \mathbf{H} \cdot \boldsymbol{\delta} + \text{residual}.$$

Now we try to find the direction vector $\boldsymbol{\delta}$ that minimizes this approximation (ignoring the residual). To do this, we take the first derivative of the approximation with respect to the direction vector $\boldsymbol{\delta}$, and set it equal to zero, which yields

$$(\partial/\partial\boldsymbol{\delta})[F(\boldsymbol{\theta}) + \boldsymbol{\delta}^{\mathrm{T}} \cdot \nabla + (1/2)\boldsymbol{\delta}^{\mathrm{T}} \cdot \mathbf{H} \cdot \boldsymbol{\delta}] = \nabla + \mathbf{H} \cdot \boldsymbol{\delta} = 0.$$

Solving the last equation for $\boldsymbol{\delta}$, we obtain

$$\boldsymbol{\delta} = \mathbf{H}^{-1} \cdot (-\nabla),$$

which provides the direction for minimizing our approximation to the objective function. But stepping from $\boldsymbol{\delta}$ to $(\boldsymbol{\theta} + \boldsymbol{\delta})$ minimizes the approximation, and not the original objective function. Therefore, we need to repeat this procedure. With each repetition, our approximation gets better, and we get closer to the minimum of the true objective function. Thus, we obtain a new estimate on iteration $\boldsymbol{\theta}(n)$, from the old estimate, $\boldsymbol{\theta}(n-1)$, by the following updating rule:

$$\boldsymbol{\theta}(n) = \boldsymbol{\theta}(n-1) + s \cdot \boldsymbol{\delta} = \boldsymbol{\theta}(n-1) + s \cdot \mathbf{H}^{-1} \cdot (-\nabla), \qquad \text{(A3.5)}$$

where s is called the step size for the change. Another line search routine is used to find the step size s, which makes the improvement in the direction $\boldsymbol{\delta}$ as large as possible. The whole process starts with the selection of an initial guess $\boldsymbol{\theta}(0)$, which may come from a grid search. The iterations continue until the magnitude of the direction for change $|\delta|$ falls below some criterion. Equation A3.4 is called the Newton-Raphson search procedure.

The Hessian matrix $\mathbf{H}$, evaluated at the minimum, is very important for evaluating the precision of the parameter estimates of the cognitive model. The programs used to implement the quasi-Newton methods have options for computing and displaying this matrix and its inverse.

First of all, the rank of the Hessian matrix can be used to determine whether or not the parameters of the model are identified. If Hessian matrix is full rank, then the parameters are identified, and if it is less than full rank, then they are not identified. Note that if the Hessian matrix is not full rank, then the inverse does not exist, and one cannot use the quasi-Newton search methods. Consequently, if the parameters are not identified, the some parameters need to be eliminated or redefined to achieve identification.

Second, the diagonal elements of the inverse of the Hessian matrix provide information about the precision of the parameter estimates. If it is assumed that the cognitive model is the true model that generated the data, then the following theorem holds for the weighted least-squares and likelihood objectives: The parameter estimates are asymptotically normally distributed, with a mean equal to the true parameter values, and with a variance-covariance matrix equal to the inverse of the Hessian matrix. Thus, the diagonal elements of $\mathbf{H}^{-1}$ are used to estimate the variances of the parameters, and the off-diagonal elements are used to estimate the covariances between parameter estimates. To be more concrete, define h_{ii}^{*} as the diagonal element of $\mathbf{H}^{-1}$ in the ith row, then we estimate the standard error of θ_i by $\sqrt{h_{ii}^{*}}$.

Application to Linear Models

It is informative to apply these ideas to the standard linear model. Define $\mathbf{Y}$ as a vector of observations, $\mathbf{X}$ is a matrix of known independent variables, $\boldsymbol{\theta}$ is a vector of parameters, and $\mathbf{E}$ is a vector of errors. The linear model is

$$\mathbf{Y} = M(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{E} = \mathbf{X} \cdot \boldsymbol{\theta} + \mathbf{E}.$$

Linear models satisfy the following important property:

$$\begin{aligned} M(\mathbf{X}, a\boldsymbol{\theta}_1 + b\boldsymbol{\theta}_2) &= \mathbf{X}(a\boldsymbol{\theta}_1 + b\boldsymbol{\theta}_2) = a\mathbf{X}\boldsymbol{\theta}_1 + b\mathbf{X}\boldsymbol{\theta}_2 \\ &= aM(\mathbf{X}, \boldsymbol{\theta}_1) + bM(\mathbf{X}, \boldsymbol{\theta}_2). \end{aligned} \tag{A3.6}$$

Nonlinear models fail to satisfy this property.

If we make the standard homogeneous variance assumption, then the variance-covariance matrix for these errors is simply $\text{Var}[\mathbf{E}] = \sigma^2\mathbf{I}$. In this case, the log likelihood can be expressed as (see Rao, 1965; or Wasserman, 2000)

$$\ln(L) = -\left\{\frac{SSE}{2\sigma^2} + \left(\frac{N}{2}\right) \cdot [\ln(2\pi) + \ln(\sigma^2)]\right\}.$$

Maximizing this likelihood for $\boldsymbol{\theta}$ is equivalent to minimizing the sum of squared error:

$$F = \frac{1}{\sigma^2}(\mathbf{Y} - \mathbf{X}\boldsymbol{\theta})^{\mathrm{T}}(\mathbf{Y} - \mathbf{X}\boldsymbol{\theta}) = \frac{1}{2\sigma^2}(\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - 2\boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{Y} + \boldsymbol{\theta}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}).$$

Thus, for linear models, the sum of squared error function is exactly a quadratic function.

The gradient is equal to

$$\nabla = (1/2\sigma^2)(-\mathbf{X}^{\mathrm{T}}\mathbf{Y} + \mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta}) = (-1/\sigma^2)\mathbf{X}^{\mathrm{T}}(\mathbf{Y} - \mathbf{X}\boldsymbol{\theta}).$$

The minimum of the objective function lies at the point where the gradient is zero:

$$-\mathbf{X}^{\mathrm{T}}\mathbf{Y} + \mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta} = 0 \rightarrow \mathbf{X}^{\mathrm{T}}\mathbf{Y} = \mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\theta},$$

producing the set of normal equations for the linear model. In this case, the optimal parameters can be found by solving the normal equations in a single step:

$$\boldsymbol{\theta} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{Y}). \tag{A3.7}$$

The Hessian of F and its inverse are equal to

$$\mathbf{H} = \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} \text{ and } \mathbf{H}^{-1} = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}.$$

The latter expression is the familiar formula for the variance-covariance matrix of the parameters for a linear model. The maximum-likelihood estimate of σ^2 is SSE/N. Substituting these estimates into the log likelihood yields the maximum log likelihood:

$$\ln(L) = -\left(\frac{N}{2}\right) \cdot \left[\ln\left(\frac{SSE}{N}\right) + \ln(2\pi) + 1\right]$$

and

$$G^2 = N \cdot \left[\ln\left(\frac{SSE}{N}\right) + \ln(2\pi) + 1\right].$$

Quasi-Newton Methods

In many cases, it is computationally intensive to determine the inverse of the Hessian matrix, $\mathbf{H}^{-1}$, for each iteration. To speed up the computations, procedures have been developed that build up an approximation to this inverse in an iterative fashion. These procedures start by setting $\mathbf{H} = \mathbf{I}$, the identity matrix, and in this case $\boldsymbol{\delta}$ is simply proportional to the gradient. This initial value for the inverse reduces the computations, but searching for the minimum using only the gradient is very slow. Thus, after each step, the $\mathbf{H}^{-1}$ matrix is updated to more closely approximate its true value, and using this estimate of the inverse improves the speed of the search and convergence to minimum. This revision of the algorithm is called the Broyden-Fletcher-Powell modification of the Newton-Raphson method. This is the most commonly used gradient search approach, and it is available in mathematical programming languages such as MATLAB, Mathematica, GAUSS, or SAS.

Gauss-Newton Search Methods

This search method is only applicable to least-squares or weighted least-squares objectives (Gallant, 1986). Given that we can always transform a *WSSE* problem into a *SSE* problem, we will only present the least-squares version of the Gauss-Newton search algorithm.

The Gauss-Newton approach differs from the Newton-Raphson approach in the following way. The Newton-Raphson approach uses a second-order Taylor series to approximate the objective function F, whereas the Gauss-Newton approach relies on a first-order approximation to the nonlinear function M that generates the model predictions. Once again, the

general idea is to find an iterative procedure to change from $\boldsymbol{\theta}$ to $(\boldsymbol{\theta} + \boldsymbol{\delta})$, so then we reduce the objective function from $SSE(\boldsymbol{\theta})$ to $SSE(\boldsymbol{\theta} + \boldsymbol{\delta})$.

This method begins by approximating the nonlinear model by a first-order Taylor series expansion:

$$M(\mathbf{X}, \boldsymbol{\theta} + \boldsymbol{\delta}) = M(\mathbf{X}, \boldsymbol{\theta}) + \frac{\partial M(\mathbf{X}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}^T} \boldsymbol{\delta} + \text{residual}.$$

The term involving $\boldsymbol{\delta}$, denoted as $\mathbf{J} = (\partial/\partial\boldsymbol{\theta})M(\mathbf{X}, \boldsymbol{\theta})$ with $J_{ij} = (\partial/\partial\boldsymbol{\theta}_j)\, M_i(\mathbf{X}, \boldsymbol{\theta})$, is called the Jacobian matrix. This approximation to M will be accurate for small $|\delta|$. Given a previous evaluation of the model at the parameter vector $\boldsymbol{\theta}$, we can approximate the predictions at a new parameter vector $(\boldsymbol{\theta} + \boldsymbol{\delta})$ by using the above approximation:

$$\mathbf{Y} = M(\mathbf{X}, \boldsymbol{\theta}) + \mathbf{J} \cdot \boldsymbol{\delta} + \mathbf{E}$$

$$\rightarrow \mathbf{Y} - M(\mathbf{X}, \boldsymbol{\theta}) = \mathbf{J} \cdot \boldsymbol{\delta} + \mathbf{E},$$

where $\mathbf{E}$ is a vector of residuals. Now we wish to find the direction of change $\boldsymbol{\delta}$ that minimizes $SSE = \mathbf{E}^T\mathbf{E}$ in the above equation. Note that the last equation is a linear equation with respect to $\boldsymbol{\delta}$. The solution for minimizing SSE with linear equations was presented earlier as Equation A3.7. To adapt the earlier solution to this problem, we treat $\mathbf{Z} = \mathbf{Y} - M(\mathbf{X}, \boldsymbol{\theta})$ as the new transformed dependent variable, and we treat the Jacobian matrix $\mathbf{J}$ as the matrix of independent variables. Then, we obtain the least-squares solution:

$$\boldsymbol{\delta} = (\mathbf{J}^T\mathbf{J})^{-1}(\mathbf{J}^T\mathbf{Z}).$$

This is the solution for the direction to change the parameter, and the new parameter vector is chosen to be a step in this direction:

$$\boldsymbol{\theta}(n) = \boldsymbol{\theta}(n-1) + s \cdot \boldsymbol{\delta} = \boldsymbol{\theta}(n-1) + s \cdot (\mathbf{J}^T\mathbf{J})^{-1}(\mathbf{J}^T\mathbf{Z}). \qquad \text{(A3.8)}$$

The step size, s, is chosen to produce the largest possible reduction in SSE in the direction given by $\boldsymbol{\delta}$.

The Gauss-Newton method is related to the Newton-Raphson method. The first term $\mathbf{J}^T\mathbf{J}$ is asymptotically equal to the Hessian matrix, and the second term $\mathbf{J}^T\mathbf{Z}$ is the negative of the gradient for the SSE objective function:

$$\begin{aligned} -\nabla &= -(\partial/\partial\boldsymbol{\theta}^T)SSE = -(\partial/\partial\boldsymbol{\theta}^T)\mathbf{E}^T\mathbf{E} \\ &= -(\partial/\partial\boldsymbol{\theta}^T)[\mathbf{Y} - M(\mathbf{X}, \boldsymbol{\theta})]^T[\mathbf{Y} - M(\mathbf{X}, \boldsymbol{\theta})] = 2\mathbf{J}^T\mathbf{Z}. \end{aligned}$$

Modified Gauss-Newton Method

If the initial starting position is far from the minimum, then the standard Gauss-Newton search encounters difficulty. In this case, it is better to start with a simple gradient search and gradually shift to the Gauss-Newton direction after the approximation starts to improve. This is called the Levenberg-Marquardt modification of the Gauss-Newton method. The modification is accomplished by adjusting the direction of search in the following manner:

$$\boldsymbol{\delta} = (\mathbf{J}^{\mathrm{T}}\mathbf{J} + \boldsymbol{\lambda}_n\mathbf{I})^{-1}(\mathbf{J}^{\mathrm{T}}\mathbf{Z}).$$

The coefficient $\boldsymbol{\lambda}_n$ determines the influence of $\mathbf{J}^{\mathrm{T}}\mathbf{J}$. Initially, this coefficient is set to a large value so that $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ has little influence, and most of the weight is placed on the gradient. Later in the iterations, this coefficient is set to a small value so that $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ has more influence on the direction. The modified Gauss-Newton method is closely related to the method of ridge regression sometimes used to obtain more robust estimates from linear models.

Example Program

Although the technical aspects of nonlinear parameter estimation can become quite complex, easy-to-use programs are available to perform this task from mathematical programming languages such as MATLAB, Mathematica, GAUSS, or SAS. The example program described below is a MATLAB program for estimating the parameters of the retention model presented in Chapter 3. The program has two parts: a "main" program and a function called "Fitr" (see Figure A3.1).

```
% main fitting program
global P;
load P; % load data
oldopts = optimset('fminunc');
options =optimset(oldopts,'Display,"iter');
% initial values
a = .8; r = 3; X0 = [a r]
''Initial Results'
G = Fitr(X0)
'Final Results'
[X G] = fminunc('Fitr,' X0,0ptions)
```

```
function G = Fitr(X)
global P
a = X(1); r = X(2); % param's
n = 200; % sample size
na = n*P; nb = n*(1-P);
t = (0:10)'; % time delays
p = 1./(1+exp(-r.*(a.^t))); % predictions
GR = na'*log(p) + nb'*log(1-p);
GC = na'*log(P) + nb'*log(1-P);
G = 2*(GC – GR); % G - Square
```

Figure A3.1 Example Parameter Estimation Program

The "main" program first loads the data stored in a vector called **P**. The "options" line defines some parameters that control the display of the fitting program. The "initial values" line defines the starting values for the iterative search process, and "initial results" line is used to compute and display the fits of these initial parameters. The "final results" line calls the MATLAB program that is used to search for the best-fitting parameters. In this case, we are using an unconstrained, modified Newton-Raphson method. The input arguments to this program include the user-defined name of the program used to compute the fit of the model, the initial parameters, and the options for the display.

The function called "Fitr" is a user-supplied function that has only one input argument and one output argument. The input argument is a vector containing the parameters to be used to fit the data. The output argument is the value of the objective function. The "global" statement is used to access the data vector **P**, which was read into the main program. The next few lines redefine the names of the parameters using more convenient names, define the sample sizes, and define the time delays. The MATLAB notation (0:10) creates a vector of integers ranging from 0 to 10. The "prediction" line computes the 11 predictions from the model for the entire vector of 11 time delays (the dot that appears before each mathematical operation is used to instruct the program to perform operations elementwise, that is, separately on each element in the vector). The next pair of lines computes the log likelihoods for the cognitive retention model and for the saturated model, and the last line computes the G^2 lack-of-fit index.

Notes

1. There is an entire literature on time series analysis of the autocorrelations, and these autocorrelations may reveal important hidden cognitive processes (see Gilden, 2001).

2. The log likelihood for the binomial distribution also includes the log of the constant $[n_A!/(n_A!\cdot n_B!)]$, which is contributed by the number of sequences for obtaining n_A corrects out of n trials. However, this constant has no bearing on any subsequent analyses, and it can be ignored. In particular, it cancels out the expression for G^2 in Equation 3.7, which is introduced later.

3. For nonlinear models, it is possible for R^2 to be less than zero, that is, worse than the null model.

4. The sum of squared error criterion was used because there is no sampling error in these data sets, and the other two criteria are not defined in this ideal case.

5. For continuously distributed observations, such as response times, we would only need to change our definitions from probability masses to probability densities, and the rest would remain the same.