Part Three
Data-Driven Models

The first two parts of this book have covered techniques to explore the behavior of a model but have assumed that the model is known. Frequently, finding the model in the first place is the most difficult, interesting, and important question. Models can come from introspection or observation (or both); here we turn to the problem of inferring a model from measured data. The model may be used to characterize and classify the data, to generalize from the measurements in order to make predictions about new observations, or most ambitiously to learn something about the rules underlying the observed behavior.

In any data-driven modeling effort the two central tasks are always choosing the functional form of the model, and using the data to determine the adjustable parameters of the model. These are closely connected but can lead to different kinds of errors. *Model mismatch* errors are those that arise from a model that is unable to represent the data, and *model estimation* errors come from using incorrect values for the model parameters. Decreasing one kind of error is likely to increases the other kind. This is called a *bias/variance tradeoff* – if you want less bias in the estimate of a model parameter, it usually costs you more variance. A more flexible model that can better represent the data may also be more easily led astray by noise in the data. Each of the coming chapters covers some kind of optimization to minimize these errors. Since one person's data may be another's noise, it will be important throughout to keep an eye on which principle is being used to optimize what quantity with respect to which error measure, and on how the "best" solution is defined. After all, everything is optimal with respect to something, or conversely no one thing is optimal with respect to everything (the *No Free Lunch* theorem [Macready & Wolpert, 1996; Wolpert & Macready, 1997; Wolpert & Macready, 2005]).

Chapter 12 introduces the canon of function fitting. Chapters 13–17 revisit these standard choices in order to uncover the power of less-familiar alternatives, and Chapters 19–21 look at the essential role of time in modeling.

12 Function Fitting

The goal of function fitting is to choose values for the parameters in a function to best describe a set of data. There are many possible reasons to do this. If a specific meaningful form for the function with a small number of free parameters is known in advance, this is called *parametric fitting*, and finding the parameter values themselves may be the goal. For example, an exponential decay constant might be sought to determine a reaction rate. If the form of the function is not known, and so a very flexible function with many free parameters is used, this becomes *nonparametric fitting* (although this distinction is often vague). One reason to do nonparametric fitting is to try to find and describe underlying trends in noisy data, in which case the fit must build in some prior beliefs and posterior observations about what defines the difference between the signal and the noise. In *function approximation* there is no noise; the goal is to take known (and perhaps laboriously calculated) values of a function and find a new function that is easier to evaluate and that can interpolate between, or extrapolate beyond, the known values.

It's useful to view function fitting in a context such as the *Minimum Description Length* principle (*MDL*) [Rissanen, 1986], or the related *Algorithmic Information Theory* [Chaitin, 1990]. One extreme is to report the observed data itself as your model. This is not hard to do, but the "model" is very large and has no ability to generalize. Another extreme is report the smallest amount of information possible needed to describe the data, but this may require a great deal of supporting documentation about how to use the model. A tidy computer program is of no use to a Martian unless it comes with a complete description of a computer that can run it. The best model typically lies between these extremes: there is some amount of information about the data, and some about the model architecture. According to MDL, the sum of these two kinds of information taken together should be as small as possible. While this principle cannot be applied directly (like so many other attractive ideas, it includes a solution to the *halting problem* of deciding if an arbitrary program will terminate, which is known to be impossible [Turing, 1936; Chaitin, 1994]), it is a useful guiding principle that can be made explicit given specific assumptions about a problem.

In this chapter we will look at the basic features of function fitting: the general principles by which data can be used to constrain a model, the (often overlooked) connection with the choice of an error measure, how to fit a model with linear and nonlinear parameters, and the limits on what we can expect to learn from fitting. We will not be particularly concerned with the functional form used; coming chapters will look in much more detail at the representation of data, functions, and optimization strategies.

12.1 MODEL ESTIMATION

The general fitting problem has three ingredients: a model architecture (which we'll call m) that has a set of adjustable parameters φ, and measured data d. The goal is to find values of the parameters that lead to the best agreement (in some sense) between the predictions of the model and the data. An example of m might be a polynomial of a given order, where the φ are the coefficients.

A reasonable way to go about finding the best coefficients is to ask for the φ that are most likely given the choice of the model and the measured data. This means that we want to find the φ that maximizes $p(\varphi|d,m)$. Using Bayes' rule (equation 6.11), our job is then to find

$$\begin{aligned}
\max_{\varphi}\ p(\varphi|d,m) &= \max_{\varphi}\ \frac{p(\varphi,d,m)}{p(d,m)} \\
&= \max_{\varphi}\ \frac{p(d|\varphi,m)\ p(\varphi|m)\ p(m)}{p(d|m)\ p(m)} \\
&= \max_{\varphi}\ \frac{p(d|\varphi,m)\ p(\varphi|m)}{p(d|m)} \\
&= \max_{\varphi}\ \frac{p(d|\varphi,m)\ p(\varphi|m)}{\int_{\varphi} p(d,\varphi|m)\ d\varphi} \\
&= \max_{\varphi}\ \frac{p(d|\varphi,m)\ p(\varphi|m)}{\int_{\varphi} p(d|\varphi,m)\ p(\varphi|m)\ d\varphi} \\
&= \max_{\varphi}\ \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \quad . \qquad (12.1)
\end{aligned}$$

The probability has been factored into three terms. The *likelihood* measures the match between the data and the predictions of the model with the coefficents, based on an *error model*. The *prior* introduces advance beliefs about which values of the coefficients are reasonable and which are not. And the *evidence* measures how well the model can describe the data.

If you solve equation (12.1) then you are an official card-carrying *Bayesian* [Bernardo & Smith, 1994; Kruschke, 2010]. The reason that there are not too many of them around is that solving equation (12.1) represents a lot of work. First of all, it's necessary to explicitly put priors on every parameter that is used. Then, the integration for the evidence is over all values of all the parameters, which can be an enormous computational task for a large model. Although efficient techniques have been developed for these kinds of integrals using *Monte-Carlo* sampling techniques that replace exact integration with a probabilistic approximation [Besag *et al*., 1995], they are still computationally intensive. Finally, the maximization over parameters is just an inner loop; the best description is given by a maximization over model architectures as well, or, even better, over the combined outputs from multiple models [Burnham & Anderson, 2002].

Much of the work in Bayesian model estimation goes into the integration for the evidence term. But this does not affect a single maximization over φ; it comes in making comparisons among competing model architectures. If we decide in advance that we are going to stick with one architecture then equation (12.1) can be simplified by dropping

the conditioning on the model:

$$\max_{\varphi} \; p(\varphi|d) = \max_{\varphi} \; \frac{p(d|\varphi)\; p(\varphi)}{p(d)} \quad . \tag{12.2}$$

Now the evidence term has become a simple prior on the likelihood of the data set. Even this can usually be dropped; it's relevant only in combining multiple data sets of varying pedigrees. Finding parameters with (12.2) is called *Maximum A Posteriori* estimation (*MAP*).

MAP still requires putting a prior on the parameters. This is a very powerful idea, to be explored in the next chapter, but if we make the simplest choice of a uniform prior $p(\varphi) = p(d) = 1$ then we're left with

$$\max_{\varphi} \; p(\varphi|d) = \max_{\varphi} \; p(d|\varphi) \quad . \tag{12.3}$$

This is the easiest kind of model estimation of all, called *Maximum Likelihood* (*ML*). That is what we will now apply.

12.2 LEAST SQUARES

Let's assume that we are given a set of N noisy measurements of a quantity y_n as a function of a variable x_n, and we seek to find values for coefficients φ in a function $y_n = y(x_n, \varphi)$ that describes their relationship (the generalization to vector variables will be straightforward). In Section 6.1.2 we learned that in the absence of any other information the *Central Limit Theorem* tells us that the most reasonable choice for the distribution of a random variable is Gaussian, and so we will make that choice for the distribution of errors in y_n. Problem 12.3 will use an entropy argument to reach the same conclusion. In practice, many systems choose to ignore this insight and have non-Gaussian distributions; the real reason why Gaussianity is so commonly (and frequently implictly) assumed is that it leads to a particularly simple and useful error model: *least squares*.

If the errors do have a Gaussian distribution around the true value $y(x_n, \varphi)$ then the probability to observe a value between y and $y + dy$ is given by a Gaussian centered on the correct value

$$p(y)\; dy = \frac{1}{\sqrt{2\pi\sigma_n^2}}\; e^{-[y-y(x_n,\varphi)]^2/(2\sigma_n^2)}\; dy \quad . \tag{12.4}$$

The variance σ_n^2 might depend on quantities such as the noise in a photodetector or the number of samples that are measured.

We will further assume that the errors between samples are independent as well as identically distributed (*iid*). This means that the probability to see the entire data set is given by the product of the probabilities to see each point,

$$p(\text{data}|\text{model}) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma_n^2}}\; e^{-[y_n-y(x_n,\varphi)]^2/(2\sigma_n^2)} \quad . \tag{12.5}$$

We seek the φ that maximizes this probability. If p is maximal then so is its logarithm

(the *log-likelihood*), and since the log of a product is equal to the sum of the logs, this becomes

$$-\log p(\text{data}|\text{model}) = \sum_{n=1}^{N} \frac{[y_n - y(x_n, \varphi)]^2}{2\sigma_n^2} + \frac{1}{2}\log(2\pi\sigma_n^2) \quad . \tag{12.6}$$

Because we've moved the minus sign to the left hand side we now want to find the φ that minimizes the right hand side. The first term measures the distance between the data and the model, and the second one catches us if we try to cheat and make a model with a huge variance that explains everything equally well (or poorly). We can drop the second term since it does not depend on the parameters φ that we are adjusting, and so we want to find the values that satisfy

$$\min_{\varphi} \sum_{n=1}^{N} \frac{[y_n - y(x_n, \varphi)]^2}{2\sigma_n^2} \quad . \tag{12.7}$$

If the variances σ_n^2 are constant (in particular, if we set $\sigma_n^2 = 1$ when we have no idea at all what it should be) this reduces to

$$\min_{\varphi} \sum_{n=1}^{N} [y_n - y(x_n, \varphi)]^2 \quad . \tag{12.8}$$

This is the familiar *least squares* error measure. It is the maximum likelihood estimator for data with normally distributed errors, but it is used much more broadly because it is simple, convenient, and frequently not too far off from an optimal choice for a particular problem. An example of where least squares might be a bad choice for an error measure is a bi-modal data set that has two peaks. The least squares error is minimized by a point between the peaks, but such a point has very little probability of actually occurring in the data set.

Instead of the square of the deviation between the model and the data, other powers can be used as an error measure. The first power (the magnitude of the difference) is the maximum likelihood estimate if the errors are distributed exponentially, and higher powers place more emphasis on outliers.

12.3 LINEAR LEAST SQUARES

Once we've chosen our error measure we need to find the parameters for the distribution that minimizes it. Perhaps the most important example of such a technique is *linear least squares*, because it is straightforward to implement and broadly applicable.

To do a least squares fit we will start by expanding our unknown function as a linear sum of M known basis functions f_m

$$y(x) = \sum_{m=1}^{M} a_m f_m(x) \quad . \tag{12.9}$$

We want to find the coefficients a_m that minimize the sum of the squared errors between this model and a set of N given observations $y_n(x_n)$. The basis functions f_m need not be orthogonal, but they must not be linear (otherwise the sum would be trivial); it is the

coefficients a_m that enter linearly. For example, the f_m could be polynomial terms, with the a_m as the coefficients of the polynomial.

A least squares fit can be written as a matrix problem

$$\begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_M(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_M(x_2) \\ f_1(x_3) & f_2(x_3) & \cdots & f_M(x_3) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_{N-1}) & f_2(x_{N-1}) & \cdots & f_M(x_{N-1}) \\ f_1(x_N) & f_2(x_N) & \cdots & f_M(x_N) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} . \quad (12.10)$$

If we have the same number of free parameters as data points then the matrix will be square, and so the coefficients can be found by inverting the matrix and multiplying it by the observations. As long as the matrix is not singular (which would happen if our basis functions were linearly dependent), this inversion could be done exactly and our fit would pass through all of the data points. If our data are noisy this is a bad idea; we'd like to have many more observations than we have model parameters. We can do this if we use the *pseudo-inverse* of the matrix to minimize the least squared error. The *Singular Value Decomposition* (*SVD*) is a powerful technique that solves this (as well as many other problems).

12.3.1 Singular Value Decomposition

For a general linear equation $\mathbf{A} \cdot \vec{v} = \vec{b}$, the space of all possible vectors $\vec{b}$ for which the equation is solvable is the *range* of **A**. The dimension of the range (i.e., the number of vectors needed to form a basis of the range) is called the *rank* of **A**. There is an associated homogeneous problem $\mathbf{A} \cdot \vec{v} = 0$; the vectors $\vec{v}$ that satisfy the homogeneous equation lie in the *nullspace* of **A**. If there is no nullspace then the matrix is of *full rank* (which is equal to the number of columns of **A**).

If **A** is an arbitrary $N \times M$ matrix, an important result from linear algebra is that it can always be written in the form [Golub & Loan, 1996]

$$\begin{pmatrix} \\ \mathbf{A} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{U} \\ \\ \end{pmatrix} \underbrace{\begin{pmatrix} w_1 & & & \mathbf{0} \\ & w_2 & & \\ & & \ddots & \\ \mathbf{0} & & & w_M \end{pmatrix}}_{\mathbf{W}} \begin{pmatrix} \\ \mathbf{V}^T \\ \\ \end{pmatrix} \quad (12.11)$$

where **U** and **V** are *orthogonal* matrices whose inverse is equal to their transpose $\mathbf{U}^T \cdot \mathbf{U} = \mathbf{V} \cdot \mathbf{V}^T = \mathbf{I}$ (where **I** is the $M \times M$ identity matrix). This is called the *Singular Value Decomposition* (*SVD*) of the matrix, and the elements of the $M \times M$ diagonal matrix **W** are the *singular values* w_i.

The reason that the SVD is so important is that the columns of **U** associated with nonzero singular values ($w_i \neq 0$) form an orthonormal basis for the range of **A**, and the columns of **V** associated with $w_i = 0$ form an orthonormal basis for the nullspace of

A. The singular values w_i give the lengths of the principal axes of the hyper-ellipsoid defined by $\mathbf{A} \cdot \vec{x}$, where $\vec{x}$ lies on a hyper-sphere $|\vec{x}|^2 = 1$.

In terms of the SVD, the solution to $\mathbf{A} \cdot \vec{v} = \vec{b}$ is

$$\vec{v} = \mathbf{V} \cdot \mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot \vec{b} \quad . \tag{12.12}$$

(the errors in this are discussed in Section 12.5). Since **W** is diagonal, its inverse $\mathbf{W}^{-1}$ is also diagonal and is found by replacing each diagonal element by its inverse. This sounds like a recipe for disaster since we just saw that $w_i = 0$ for elements of the nullspace. The odd solution to this problem is simply to declare that $1/0 = 0$, and set to zero the diagonal elements of $\mathbf{W}^{-1}$ corresponding to $w_i = 0$. To see how this apparent nonsense works, let's look for another solution to $\mathbf{A} \cdot \vec{v} = \vec{b}$. Assume that $\vec{v}$ is found according to the prescription for zeroing singular values in equation (12.12). We can add an arbitrary vector $\vec{v}'$ from the nullspace ($\mathbf{A} \cdot \vec{v}' = 0$) and still have a solution. The magnitude of the sum of these vectors is

$$\begin{aligned} |\vec{v} + \vec{v}'| &= |\mathbf{V} \cdot \mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot \vec{b} + \vec{v}'| \\ &= |\mathbf{V} \cdot (\mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot \vec{b} + \mathbf{V}^T \cdot \vec{v}')| \\ &= |\mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot \vec{b} + \mathbf{V}^T \cdot \vec{v}'| \quad . \end{aligned} \tag{12.13}$$

Multiplication by **V** was eliminated in the last line because **V** is an orthogonal matrix and hence does not change the magnitude of the answer (equation 13.7). The magnitude of $|\vec{v} + \vec{v}'|$ is made up of the sum of two vectors. The first one will have its ith element equal to 0 for every vanishing singular value $w_i = 0$ because of the rule for zeroing these elements of $\mathbf{W}^{-1}$. On the other hand, since $\vec{v}'$ is in the nullspace,

$$\begin{aligned} \mathbf{A} \cdot \vec{v}' &= \vec{0} \\ \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T \cdot \vec{v}' &= \vec{0} \\ \mathbf{W} \cdot \mathbf{V}^T \cdot \vec{v}' &= \mathbf{U}^T \cdot \vec{0} \\ \mathbf{W} \cdot \mathbf{V}^T \cdot \vec{v}' &= \vec{0} \quad . \end{aligned} \tag{12.14}$$

This means that the ith component of the vector $\mathbf{V}^T \cdot \vec{v}'$ must equal 0 for every $w_i \neq 0$. Returning to the last line of equation (12.13), the left hand term can be nonzero only if $w_i \neq 0$, and the right hand term can be nonzero only if $w_i = 0$: these two vectors are orthogonal. Therefore, the magnitude of their sum is a minimum if $\vec{v}' = \vec{0}$. Adding any component from the nullspace to $\vec{x}$ increases its magnitude. This means that for an underdetermined problem (i.e., one in which there is a nullspace), the SVD along with the rule for zeroing elements in $\mathbf{W}^{-1}$ chooses the answer with the smallest magnitude (i.e., no component in the nullspace).

Now let's look at what the SVD does for an overdetermined problem. This is the case that we care about for fitting data, where we have more measurements than free parameters. We can no longer hope for an exact solution, but we can look for one that minimizes the *residual*

$$|\mathbf{A} \cdot \vec{v} - \vec{b}| = \sqrt{(\mathbf{A} \cdot \vec{v} - \vec{b})^2} \quad . \tag{12.15}$$

Let's once again choose $\vec{v}$ by zeroing singular values in equation (12.12), and see what happens to the residual if we add an arbitrary vector $\vec{v}'$ to it. This adds an error of

$\vec{b}' = \mathbf{A} \cdot \vec{v}'$:

$$\begin{aligned} |\mathbf{A} \cdot (\vec{v} + \vec{v}') - \vec{b}| &= |\mathbf{A} \cdot \vec{v} + \vec{b}' - \vec{b}| \\ &= |(\mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T) \cdot (\mathbf{V} \cdot \mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot \vec{b}) + \vec{b}' - \vec{b}| \\ &= |(\mathbf{U} \cdot \mathbf{W} \cdot \mathbf{W}^{-1} \cdot \mathbf{U}^T - \mathbf{I}) \cdot \vec{b} + \vec{b}'| \\ &= |\mathbf{U} \cdot [(\mathbf{W} \cdot \mathbf{W}^{-1} - \mathbf{I}) \cdot \mathbf{U}^T \cdot \vec{b} + \mathbf{U}^T \cdot \vec{b}']| \\ &= |(\mathbf{W} \cdot \mathbf{W}^{-1} - \mathbf{I}) \cdot \mathbf{U}^T \cdot \vec{b} + \mathbf{U}^T \cdot \vec{b}'| \quad . \end{aligned} \tag{12.16}$$

Here again the magnitude is the sum of two vectors. $(\mathbf{W} \cdot \mathbf{W}^{-1} - \mathbf{I})$ is a diagonal matrix, with nonzero entries for $w_i = 0$, and so the elements of the left term can be nonzero only where $w_i = 0$. The right hand term can be rewritten as follows:

$$\begin{aligned} \mathbf{A} \cdot \vec{v}' &= \vec{b}' \\ \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T \cdot \vec{v}' &= \vec{b}' \\ \mathbf{W} \cdot \mathbf{V}^T \cdot \vec{v}' &= \mathbf{U}^T \cdot \vec{b}' \quad . \end{aligned} \tag{12.17}$$

The ith component can be nonzero only where $w_i \neq 0$. Once again, we have the sum of two vectors, one of which is nonzero only where $w_i = 0$, and the other where $w_i \neq 0$, and so these vectors are orthogonal. Therefore, the magnitude of the residual is a minimum if $\vec{v}' = 0$, which is the choice that SVD makes. Thus, the SVD finds the vector that minimizes the least squares residual for an overdetermined problem.

The computational cost of finding the SVD of an $N \times M$ matrix is $\mathcal{O}(NM^2 + M^3)$. This is comparable to the ordinary inversion of an $M \times M$ matrix, which is $\mathcal{O}(M^3)$, but the prefactor is larger. Because of its great practical significance, good SVD implementations are available in most mathematical packages. We can now see that it is ideal for solving equation (12.10). If the basis functions are chosen to be polynomials, the matrix to be inverted is called a *Vandermonde* matrix. For example, let's say that we want to fit a 2D bilinear model $z = a_0 + a_1 x + a_2 y + a_3 xy$. Then we must invert

$$\begin{pmatrix} 1 & x_1 & y_1 & x_1 y_1 \\ 1 & x_2 & y_2 & x_2 y_2 \\ 1 & x_3 & y_3 & x_3 y_3 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{N-1} & y_{N-1} & x_{N-1} y_{N-1} \\ 1 & x_N & y_N & x_N y_N \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{N-1} \\ z_N \end{pmatrix} \quad . \tag{12.18}$$

If the matrix is square ($M = N$), the solution can go through all of the data points. These are interpolating polynomials, like those we used for finite elements. A rectangular matrix ($M < N$) is the relevant case for fitting data. If there are any singular values near zero (given noise and the finite numerical precision they won't be exactly zero) it means that some of our basis functions are nearly linearly dependent and should be removed from the fit. If they are left in, SVD will find the set of coefficients with the smallest overall magnitude, but it is best for numerical accuracy (and convenience in using the fit) to remove the terms with small singular values. The SVD inverse will then provide the coefficients that give the best least squares fit. Choosing where to cut off the spectrum of singular values depends on the context; a reasonable choice is the largest singular value

weighted by the computer's numerical precision, or by the fraction of noise in the data [Golub & Loan, 1996].

In addition to removing small singular values to eliminate terms which are weakly determined by the data, another good idea is to scale the expansion terms so that the magnitudes of the coefficients are comparable, or even better to rescale the data to have unit variance and zero mean (almost always a good idea in fitting). If 100 is a typical value for x and y, then 10^4 will be a typical value for their product. This means that the coefficient of the xy term must be ~ 100 times smaller than the coefficient of the x or y terms. For higher powers this problem will be even worse, ranging from an inconvenience in examining the output from the fit, to a serious loss of numerical precision as a result of multiplying very large numbers by very small numbers.

12.4 NONLINEAR LEAST SQUARES

Using linear least squares we were able to find the best set of coefficients $\vec{a}$ in a single step (the SVD inversion). The price for this convenience is that the coefficients cannot appear inside the basis functions. For example, we could use Gaussians as our bases, but we would be able to vary only their amplitude and not their location or variance. It would be much more general if we could write

$$y(x) = \sum_{m=1}^{M} f_m(x, \vec{a}_m) \quad , \tag{12.19}$$

where the coefficients are now inside the nonlinear basis functions. We can still seek to minimize the error

$$\chi^2(\vec{a}) = \sum_{n=1}^{N} \left(\frac{y_n - y(x_n, \vec{a})}{\sigma_n} \right)^2 \quad , \tag{12.20}$$

but we will now need to do an iterative search to find the best solution, and we are no longer guaranteed to find it (see Section 14.5).

The basic techniques for nonlinear fitting that we'll cover in this chapter are based on the insight that we may not be able to invert a matrix to find the best solution, but we can evaluate the error locally and then move in a direction that improves it. The gradient of the error is

$$(\nabla \chi^2)_k = \frac{\partial \chi^2}{\partial a_k} = -2 \sum_{n=1}^{N} \frac{y_n - y(x_n, \vec{a})}{\sigma_i^2} \frac{\partial y(x_n, \vec{a})}{\partial a_k} \quad , \tag{12.21}$$

and its second derivative (the *Hessian*) is

$$\begin{aligned} \mathbf{H}_{kl} &= \frac{\partial^2 \chi^2}{\partial a_k \partial a_l} \\ &= 2 \sum_{n=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_n, \vec{a})}{\partial a_k} \frac{\partial y(x_n, \vec{a})}{\partial a_l} - [y_n - y(x_n, \vec{a})] \frac{\partial^2 y(x_n, \vec{a})}{\partial a_l \partial a_k} \right] \quad . \end{aligned} \tag{12.22}$$

Since the second term in the Hessian depends on the sum of terms proportional to the

residual between the model and the data, which should be small and can change sign, it is customary to drop this term in nonlinear fitting.

From a starting guess for $\vec{a}$, we can update the estimate by the method of *steepest descent* or *gradient descent*, taking a step in the direction in which the error is decreasing most rapidly

$$\vec{a}_{new} = \vec{a}_{old} - \alpha \nabla \chi^2(\vec{a}_{old}) \quad , \tag{12.23}$$

where α determines how big a step we make. On the other hand, χ^2 can be expanded around a point $\vec{a}_0$ to second order as

$$\chi^2(\vec{a}) = \chi^2(\vec{a}_0) + [\nabla \chi^2(\vec{a}_0)] \cdot (\vec{a} - \vec{a}_0) + \frac{1}{2}(\vec{a} - \vec{a}_0) \cdot \mathbf{H} \cdot (\vec{a} - \vec{a}_0) \quad , \tag{12.24}$$

which has a gradient

$$\nabla \chi^2(\vec{a}) = \nabla \chi^2(\vec{a}_0) + \mathbf{H} \cdot (\vec{a} - \vec{a}_0) \quad . \tag{12.25}$$

The minimum ($\nabla \chi^2(\vec{a}) = 0$) can therefore be found by iterating

$$\vec{a}_{new} = \vec{a}_{old} - \mathbf{H}^{-1} \cdot \nabla \chi^2(\vec{a}_{old}) \tag{12.26}$$

(this is *Newton's method*). Either of these techniques lets us start with an initial guess for $\vec{a}$ and then successively refine it.

12.4.1 Levenberg–Marquardt Method

Far from a minimum Newton's method is completely unreliable: the local slope may shoot the new point further from the minimum than the old one was. On the other hand, near a minimum Newton's method converges very quickly and gradient descent slows to a crawl since the gradient being descended is disappearing. A natural strategy is to use gradient descent far away, and then switch to Newton's method close to a minimum. But how do we decide when to switch between them, and how large should the gradient descent steps be? The *Levenberg–Marquardt method* [Marquardt, 1963] is a clever solution to these questions, and is the most common method used for nonlinear least squares fitting.

We can use the Hessian to measure the curvature of the error surface, taking small gradient descent steps if the surface is curving quickly. Using the diagonal elements alone in the Hessian to measure the curvature is suggested by the observation that this gives the correct units for the scale factor α in equation (12.23):

$$\delta a_i = -\frac{1}{\lambda \mathbf{H}_{ii}} \frac{\partial \chi^2}{\partial a_i} \quad , \tag{12.27}$$

where λ is a new dimensionless scale factor. If we use this weighting for gradient descent, we can then combine it with Newton's method by defining a new matrix

$$\begin{aligned} M_{ii} &= \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_i^2}(1 + \lambda) \\ M_{ij} &= \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_i \partial a_j} \quad (i \neq j) \quad . \end{aligned} \tag{12.28}$$

If we use this to take steps given by

$$\mathbf{M} \cdot \delta \vec{a} = -\nabla \chi^2 \tag{12.29}$$

or

$$\delta\vec{a} = -\mathbf{M}^{-1} \cdot \nabla\chi^2 \quad , \tag{12.30}$$

when $\lambda = 0$ this just reduces to Newton's method. On the other hand, if λ is very large then the diagonal terms will dominate, which is just gradient descent (equation 12.23). λ controls an interpolation between steepest descent and Newton's method. To use the Levenberg–Marquardt method, λ starts off moderately large. If the step improves the error, λ is decreased (Newton's method is best near a minimum), and if the step increases the error then λ is increased (gradient descent is better).

χ^2 can easily have many minima, but the Levenberg–Marquardt method will find only the local minimum closest to the starting condition. For this reason, a crucial sanity check is to plot the fitting function with the starting parameters and compare it with the data. If it isn't even close, it is unlikely that Levenberg–Marquardt will converge to a useful answer. It is possible to improve its performance in these cases by adding some kind of randomness that lets it climb out of small minima. If a function is hard to fit because it has very many local minima, or the parameter space is so large that it is hard to find sane starting values, then a technique that is better at global searching is called for. These extensions will be covered in Chapter 15.

12.5 ERRORS

The preceeding fitting procedures find the best values for adjustable parameters, but there's no guarantee that the best is particularly good. As important as finding the values is estimating their errors. These come in two flavors: *statistical* errors from the experimental sampling procedure, and *systematic* errors due to biases in the measuring and modeling process. There are a number of good techniques for estimating the former; bounding the latter can be more elusive and requires a detailed analysis of how the data are acquired and handled.

It's frequently the case that there is an error model for the measurements known in advance, such as counting statistics or the presence of Johnson noise in an amplifier [Gershenfeld, 2000]. For a linear fitting procedure it is then straightforward to propagate this through the calculation. If there is an error $\vec{\zeta}$ in a measurement, it causes an error $\vec{\eta}$ in an SVD fit:

$$\vec{v} + \vec{\eta} = \mathbf{V} \cdot \mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot (\vec{b} + \vec{\zeta}) \quad . \tag{12.31}$$

Since this is a linear equation, these random variables are related by

$$\vec{\eta} = \mathbf{V} \cdot \mathbf{W}^{-1} \cdot \mathbf{U}^T \cdot \vec{\zeta} \quad . \tag{12.32}$$

If the components of $\vec{\zeta}$ are zero-mean with a variance σ_ζ^2 then the variance in the ith component of $\vec{\eta}$ is

$$\begin{aligned}\sigma_{\eta,i}^2 &= \langle \eta_i \eta_i \rangle \\ &= \left\langle \sum_j V_{ij} \frac{1}{w_j} \sum_k U_{kj} \zeta_k \sum_l V_{il} \frac{1}{w_l} \sum_m U_{ml} \zeta_m \right\rangle\end{aligned}$$

$$
\begin{aligned}
&= \sum_j \sum_l V_{ij} V_{il} \frac{1}{w_j} \frac{1}{w_l} \sum_k \sum_m U_{kj} U_{ml} \underbrace{\langle \zeta_k \zeta_m \rangle}_{\sigma_\zeta^2 \delta_{km}} \\
&= \sum_j \sum_l V_{ij} V_{il} \frac{1}{w_j} \frac{1}{w_l} \underbrace{\sum_k U_{kj} U_{kl}}_{\delta_{jl}} \sigma_\zeta^2 \\
&= \sigma_\zeta^2 \sum_j \frac{V_{ij}^2}{w_j^2} \quad . \qquad (12.33)
\end{aligned}
$$

In the second line we've assumed that the measurement errors are an uncorrelated random variable with a fixed variance σ_b^2; if this is not the case then their covariance must be carried through the calculation. In the third line we've used the orthonormality of $\mathbf{U}$. Therefore, the error in the fit relative to that in the data is

$$
\frac{\sigma_{\eta,i}^2}{\sigma_\zeta^2} = \sum_j \frac{V_{ij}^2}{w_j^2} \quad . \qquad (12.34)
$$

Absent such insight into an error model for the measurements it's necessary to use the observations themselves to estimate the fitting errors. If you are in the fortunate position of being able to generate unlimited amounts of data this can be done following its definition by analyzing an ensemble of independent data sets and then reporting the distribution of fitting results. But in the much more likely circumstance of limited data it's not possible to produce an ensemble of measurements.

Or is it? *Bootstrap* resampling is based on the apparently-circular reasoning that the data set is drawn from the distribution that describes it and hence drawing from the data approximates that distribution. This is done by sampling with *replacement*. A random number generator is used to choose elements of the data set, with an element remaining in the original data set after it is chosen so that it can reappear in the derived one multiple times. Random selection continues until a new data set has been produced of the same size as the original one. This one uses the same elements, but it is an independent sampling of them. These data can then be fit, and the resampling done again as many times as desired. Problem 12.1 looks at an example of this.

Bootstrap error estimation was originally statistically suspect because it appears to violate the deeply-held belief that data should not be reused in analysis. But, over time, both theoretical and experimental work has shown that, while it is not as reliable as using truly independent data sets, bootstrap can provide useful information about errors not available in a single fit [Efron & Tibshirani, 1994]. Its typical performance can in fact be quite good, although the worst-case errors in the errors can be quite bad.

Beyond bootstrap, in fitting functions that will be used for making forecasts the most important error is how well the model generalizes on data not seen in the fitting procedure. This can be evaluated through *cross-validation*, discussed in Section 14.4.

12.6 ESTIMATION, FISHER INFORMATION, AND THE CRAMÉR–RAO INEQUALITY

We've seen increasingly powerful techniques to extract functions and errors from data. Is there no limit to this cleverness? Unfortunately, and not surprisingly, there is indeed a limit on how much information about unknown parameters can be extracted from a set of measurements. This chapter closes with a view of the information in a probability distribution that sets a limit on the accuracy of measurements.

Let $p_\alpha(x)$ be a probability distribution that depends on a parameter α (such as the variance of a Gaussian); the goal is to estimate the value of α from a series of measurements of x. Let $f(x_1, x_2, ..., x_N)$ be the estimator of α. It is *biased* if $\langle f(x_1, x_2, \ldots, x_N)\rangle \neq \alpha$, and it is *consistent* if $\lim_{N\to\infty} f(x_1, x_2, \ldots, x_N) = \alpha$. An estimator f_1 *dominates* f_2 if $\langle (f_1(x_1, x_2, \ldots, x_N) - \alpha)^2\rangle \leq \langle (f_2(x_1, x_2, \ldots, x_N) - \alpha)^2\rangle$. This raises the question of what is the minimum variance possible for an unbiased estimator of α? The answer is given by the *Cramér–Rao bound*.

Start by defining the *score*:

$$V = \frac{\partial}{\partial\alpha}\log p_\alpha(x) = \frac{\partial_\alpha p_\alpha(x)}{p_\alpha(x)} \quad . \tag{12.35}$$

The expected value of the score is

$$\begin{aligned}\langle V\rangle &= \int_{-\infty}^{\infty} p_\alpha(x)\frac{\partial_\alpha p_\alpha(x)}{p_\alpha(x)}\,dx \\ &= \int_{-\infty}^{\infty} \partial_\alpha p_\alpha(x)\,dx \\ &= \partial_\alpha \int_{-\infty}^{\infty} p_\alpha(x)\,dx \\ &= \partial_\alpha 1 \\ &= 0 \quad .\end{aligned} \tag{12.36}$$

This means that $\sigma^2(V) = \langle V^2\rangle$. The variance of the score is called the *Fisher information*:

$$J(\alpha) = \langle[\partial_\alpha \log p_\alpha(x)]^2\rangle \quad . \tag{12.37}$$

The score for a set of independent, identically distributed variables is the sum of the individual scores

$$\begin{aligned}V(x_1, x_2, \ldots, x_N) &= \partial_\alpha \log p_\alpha(x_1, x_2, \ldots, x_N) \\ &= \sum_{n=1}^{N} \partial_\alpha \log p_\alpha(x_n) \\ &= \sum_{n=1}^{N} V(x_n) \quad ,\end{aligned} \tag{12.38}$$

and so the Fisher information for the set is

$$\begin{aligned}J_N(\alpha) &= \langle[\partial_\alpha \log p_\alpha(x_1, x_2, \ldots, x_N)]^2\rangle \\ &= \langle V^2(x_1, x_2, \ldots, x_N)\rangle\end{aligned}$$

$$
\begin{aligned}
&= \left\langle \left(\sum_{n=1}^{N} V(x_n) \right)^2 \right\rangle \\
&= \sum_{n=1}^{N} \langle V^2(x_n) \rangle \\
&= N\ J(\alpha)
\end{aligned} \tag{12.39}
$$

(remember that the individual scores are uncorrelated).

The Cramér–Rao inequality states that the mean square error of an unbiased estimator f of α is lower bounded by the reciprocal of the Fisher information:

$$
\sigma^2(f) \geq \frac{1}{J(\alpha)} \quad . \tag{12.40}
$$

To prove this, start with the *Cauchy–Schwarz inequality*

$$
\begin{aligned}
\langle (V - \langle V \rangle)(f - \langle f \rangle) \rangle^2 &\leq \langle (V - \langle V \rangle)^2 \rangle \langle (f - \langle f \rangle)^2 \rangle \\
\langle Vf - \langle V \rangle f - V \langle f \rangle + \langle V \rangle \langle f \rangle \rangle^2 &\leq \langle V^2 - 2V \langle V \rangle + \langle V \rangle^2 \rangle \langle (f - \langle f \rangle)^2 \rangle \\
\langle Vf \rangle^2 &\leq \langle V^2 \rangle \langle (f - \langle f \rangle)^2 \rangle \\
\langle Vf \rangle^2 &\leq J(\alpha)\ \sigma^2(f)
\end{aligned} \tag{12.41}
$$

(remember $\langle V \rangle = 0$). The lefthand side equals one:

$$
\begin{aligned}
\langle Vf \rangle &= \int_{-\infty}^{\infty} \frac{\partial_\alpha p_\alpha(x)}{p_\alpha(x)} f(x) p_\alpha(x)\ dx \\
&= \int_{-\infty}^{\infty} \partial_\alpha p_\alpha(x) f(x)\ dx \\
&= \partial_\alpha \int_{-\infty}^{\infty} p_\alpha(x) f(x)\ dx \\
&= \partial_\alpha \langle f(x) \rangle \\
&= \partial_\alpha \alpha \\
&= 1 \quad ,
\end{aligned} \tag{12.42}
$$

thus proving the Cramér–Rao inequality.

Just like the information theoretic channel capacity, this sets a lower limit on what is possible but does not provide any guidance in finding the minimum variance unbiased estimator. The inequality measures how much information the distribution provides about a parameter. Not surprisingly, the Fisher information can be related to the entropy of the distribution; this is done by *de Bruijn's identity* [Cover & Thomas, 2006]. Roughly, the entropy measures the volume and the Fisher information measures the surface of the distribution.

One final caution about the Cramér-Rao bound. Not only may it not be reachable in practice, but it may be misleading because it is a bound on unbiased estimators. Unbiased does not necessarily mean better: it is possible for a biased estimator to dominate an unbiased one (as well as have other desirable characteristics). However, just like channel capacity, although it should not be taken too literally it does provide a good rough estimate of what is plausible and what is not.

12.7 SELECTED REFERENCES

[Press *et al*., 2007] Press, William H., Teukolsky, Saul A., Vetterling, William T., & Flannery, Brian P. (2007). *Numerical Recipes in C: The Art of Scientific Computing*. 3nd edn. Cambridge: Cambridge University Press.

Numerical Recipes is particularly strong for function fitting.

[Cover & Thomas, 2006] Cover, Thomas M., & Thomas, Joy A. (2006). *Elements of Information Theory*. 2nd edn. New York: Wiley-Interscience.

Good coverage of the many connections between information theory and statistics.

12.8 PROBLEMS

(12.1) Generate 100 points x uniformly distributed between 0 and 1, and let $y = 2+3x+\zeta$, where ζ is a Gaussian random variable with a standard deviation of 0.5. Use an SVD to fit $y = a + bx$ to this data set, finding a and b. Evaluate the errors in a and b

- (*a*) With equation (12.34)
- (*b*) By bootstrap sampling to generate 100 data sets
- (*c*) From fitting an ensemble of 100 independent data sets

(12.2) Generate 100 points x uniformly distributed between 0 and 1, and let $y = \sin(2 + 3x) + \zeta$, where ζ is a Gaussian random variable with a standard deviation of 0.1. Write a Levenberg-Marquardt routine to fit $y = \sin(a+bx)$ to this data set starting from $a = b = 1$, and investigate the convergence for both fixed and adaptively adjusted λ values.

(12.3) An alternative way to choose among models is to select the one that makes the weakest assumptions about the data; this is the purpose of *maximum entropy* methods. Assume that what is measured is a set of expectation values for functions f_i of a random variable x,

$$\langle f_i(x) \rangle = \int_{-\infty}^{\infty} p(x) f_i(x)\, dx \quad . \tag{12.43}$$

- (*a*) Given these measurements, find the compatible normalized probability distribution $p(x)$ that maximizes the *differential entropy*

$$S = -\int_{-\infty}^{\infty} p(x) \log p(x)\, dx \quad . \tag{12.44}$$

- (*b*) What is the maximum entropy distribution if we know only the second moment

$$\sigma^2 = \int_{-\infty}^{\infty} p(x)\, x^2\, dx \quad ? \tag{12.45}$$

(12.4) Now consider the reverse situation. Let's say that we know that a data set $\{x_n\}_{n=1}^N$ was drawn from a Gaussian distribution with variance σ^2 and unknown mean μ. Try to find an optimal estimator of the mean (one that is unbiased and has the smallest possible error in the estimate).