

Linear Least-Squares Data Fitting

6.1. Introduction

Recall that in chapter 3 we were discussing linear systems of equations, written in shorthand in the form

$$A\mathbf{x} = \mathbf{b}.$$

In chapter 3, we just considered square, nonsingular systems, which have exactly one solution. In this chapter, we look at the general case. The system might not have any solution, or it might have infinitely many. We will define something called the *least squares solution*, and look at 3 different methods for finding it.

6.1.1. Linear Least Squares Problems. Let us assume we have a set of m linear equations in n unknowns

$$A\mathbf{x} = \mathbf{b}.$$

Recall from chapter 3 that if $m < n$, the system is *underdetermined*, and in general has an infinite set of solutions. If $m > n$, the system is *overdetermined*, and in general has no solution. Even if $m = n$, there may be no solution or an infinite set of solutions, if the matrix is singular.

Let us tackle the case of no solution first. Recall the definition of *residual*

$$\mathbf{r} = A\mathbf{x} - \mathbf{b}.$$

If \mathbf{x} is a solution, the residual is zero. If there is no solution, the usual approach is to make \mathbf{r} as small as possible, and to call the corresponding \mathbf{x} a *generalized solution*.

Now \mathbf{r} is a vector. Making it as small as possible means that we have to minimize its norm. Which norm? Well, any choice of norm leads to a corresponding mathematical problem, and in general different norms lead to different generalized solutions. The norm that is easiest to handle in practice is the 2-norm, and that is all we will do in this chapter. In practice, one minimizes the *square* of the 2-norm, since then we don't have to carry all those square roots around. The result is the same.

If there is a unique \mathbf{x} for which the 2-norm of the residual is a minimum, that works fine. What if there are an infinite number of \mathbf{x} that minimize the residual? In the textbook, this is called the *degenerate case*, and is treated separately in section 6.8. I will treat it along with the non-degenerate case.

In the degenerate case, it is customary to choose the \mathbf{x} which is itself the smallest, out of all \mathbf{x} which minimize \mathbf{r} .

Altogether, we have the following definition: The *least squares solution* of a system

$$A\mathbf{x} = \mathbf{b}$$

of linear equations is the \mathbf{x} which minimizes

$$\|A\mathbf{x} - \mathbf{b}\|.$$

If there is more than one such \mathbf{x} , we choose the one for which $\|\mathbf{x}\|$ is a minimum.

Note that if there is a solution, it will also be the only generalized solution.

6.1.2. Source of Linear Least Squares Problems. A common source of these problems, and in fact the only source we will discuss in this class, is linear approximation at a discrete set of points, or curve fitting.

Recall from chapter 4 that the object of *interpolation* is to find a curve from a certain family that takes on given values. The object of *approximation* is to find a curve from a certain family that passes as close as possible to some given points or to a given curve. The case of approximating a curve is more advanced; we will concentrate on finding a curve that passes close to a finite set of points.

In the linear case, the interpolating or approximating function must be of the form

$$f(x) = \sum_j a_j \phi_j(x),$$

where the ϕ_j are the basis functions (polynomials, splines, or whatever), and the a_j are the coefficients to be determined. If we have n basis functions and want the curve to go through or near m points, we get an $m \times n$ system of linear equations.

Example: Put a straight line through the points $(1, 1)$, $(1.5, 2)$, $(2, 2)$, $(2.5, 3)$. If we assume the equation

$$f(x) = a_0 + a_1x$$

for the line (i.e. $\phi_0(x) = 1$, $\phi_1(x) = x$), we get the equations

$$a_0 + a_1 = 1$$

$$a_0 + 1.5a_1 = 2$$

$$a_0 + 2a_1 = 2$$

$$a_0 + 2.5a_1 = 3$$

The source of the numbers might be a lab experiment: you know that the solution is supposed to be a straight line, but your measurements contain errors, and there is only an approximate line passing through the points. \square

6.1.3. Weighted Least Squares Problems. Instead of minimizing the square of the 2-norm of the residual

$$\|\mathbf{r}\|^2 = \sum_j r_j^2$$

we could introduce weights $w_j \geq 0$, and instead minimize

$$\sum_j w_j r_j^2.$$

This is called a *weighted least squares problem*. This is useful for example if you know that some measurements are more accurate than others, and you want to put more emphasis on those than on some others.

The mathematics gets more complicated in weighted least squares problems, but the basic ideas remain the same. We will not consider this in detail.

6.2. Exploring Data

We will skip this section.

6.3. The Normal Equations

This method is only capable of minimizing the residual, i.e. it can only handle the non-degenerate case. This implies that we must have $m \geq n$. (The case $m < n$ is always degenerate).

There are two approaches to deriving this method. One approach is to take a semester of linear algebra, after which you can derive the equations in two or three lines. The other approach uses calculus and is a little messy, so I will just describe it in words:

You write out the square of the 2-norm of the residual as a quadratic function in the unknowns x_1 through x_n . To minimize it, you set the n partial derivatives to zero, and get a system of n linear equations in n unknowns.

Either way, you end up with the so-called *normal equations*. In matrix notation, they are written as

$$A^T A \mathbf{x} = A^T \mathbf{b}$$

This is now a square system of size $n \times n$ and can be solved by Gaussian elimination. (Actually, the matrix $A^T A$ is always symmetric, so there is a faster way known as *Cholesky decomposition*, but Gaussian elimination also works).

The matrix $A^T A$ may turn out to be singular. If that happens, your matrix is degenerate, and you should switch to a different method for solving the least squares problem.

Example: For the equations from above, we get

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2 \\ 1 & 2.5 \end{pmatrix}, \quad A^T A = \begin{pmatrix} 4 & 7 \\ 7 & 13.5 \end{pmatrix},$$

$$\mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \end{pmatrix}, \quad A^T \mathbf{b} = \begin{pmatrix} 8 \\ 15.5 \end{pmatrix},$$

The solution of the normal equations is $a_0 = -0.1$, $a_1 = 1.2$, so the line that best fits the data is

$$y = 1.2x - 0.1$$

□

The normal equations approach may work quite well for small values of n . However, even for moderately large values of n (like 20 or 30), the normal equations tend to become unstable. The reason is that the condition number of $A^T A$ is approximately the square of the condition number of A . (The condition number of a non-square matrix is defined on page 62 of NMS).

Thus, if A has a reasonable condition number of about 1,000, then $A^T A$ already has a condition number of 10^6 . For this reason, the method of normal equations is not recommended for anything but very small systems.

6.4. Orthogonal Factorizations

The technique described in this section (called the *QR factorization*) takes about twice as much computer time as the normal equations, but it is much more stable. It can also handle the degenerate case, and is recommended for most cases.

Remark: In the degenerate case, this method does not actually find the least squares solution described earlier. It does find an \mathbf{x} which minimizes the residual, but instead of resolving the problem of multiple solutions by choosing the \mathbf{x} with minimum 2-norm, it is content to select an \mathbf{x} for which the 2-norm is small, but not necessarily the minimum. The exact effect is hard to describe geometrically.

Recall the basic approach to solving a system of linear equations: we used a sequence of allowed operations to transform the system to a simpler one (triangular matrix), and then solved the simpler one. Allowable operations were those that did not change the solution, like interchanging rows and adding a multiple of one row to another.

The same approach is used here: we apply a sequence of allowed operations to a least squares problem to transform it to an easier one. The easier problem involves again a triangular matrix. Allowable operations in this case are those that do not change the 2-norms of vectors. This means rotations and reflections.

Let me state that again, for emphasis: The least squares solution of

$$A\mathbf{x} = \mathbf{b}$$

is in general not the same as the least squares solution of

$$BA\mathbf{x} = B\mathbf{b},$$

but they *are* the same if B is a matrix corresponding to a rotation or reflection. These matrices are called *orthogonal*.

The official definition is this: a square matrix Q is called *orthogonal* if $Q^T = Q^{-1}$. Orthogonal matrices have many nice properties; among the most important ones are

- $\det(Q) = \pm 1$.
- Orthogonal matrices correspond to either a rotation of n -dimensional space (if $\det = 1$), or a rotation combined with a reflection (if $\det = -1$).
- The column vectors of Q all have 2-norm 1 and are mutually orthogonal. (Vectors are called orthogonal if their dot product is zero). Geometrically, this means they are perpendicular. From this property the matrices got their name.

- The same thing is true for the rows.
- Transpose, inverse and product of orthogonal matrices are orthogonal.
- If \mathbf{v} is any vector, then \mathbf{v} and $Q\mathbf{v}$ have the same 2-norm. This is because the 2-norm is the geometric length, and a rotation or reflection does not change the length.
- The condition number of an orthogonal matrix is 1.

Orthogonal matrices are popular in numerical analysis because operations with them are very stable. The condition number of a matrix not only affects the solution of systems of equations, but also simple multiplication of a vector by the matrix. If there are roundoff errors in \mathbf{v} , then $A\mathbf{v}$ usually has larger errors.

Not so for orthogonal matrices, where the condition number is 1. You could multiply \mathbf{v} by a thousand orthogonal matrices in a row (sometimes you have to do that), and it would not magnify the roundoff error at all. If numerical stability is important, methods based on orthogonal matrices are usually recommended.

The property $Q^T = Q^{-1}$ is nice, too: If you need the inverse, it is right there.

In addition to that, orthogonal matrices are a natural choice for use in least squares problems, since they leave the solution invariant.

Now, back to the main topic. Recall that Gaussian elimination was based on the factorization

$$A = LU,$$

where the matrix A was factored into a lower triangular matrix L and an upper triangular matrix U . There is another factorization (called the *QR factorization*)

$$A = QR,$$

where A is factored into an orthogonal matrix Q and an upper triangular matrix R .

Let us not worry about how exactly you do that, even though the book does that in some detail. I could talk about this topic for several days. There are two main techniques, based on *Householder matrices* and on *Givens rotations*. They sort of work like Gaussian elimination: we find an orthogonal matrix which wipes out a particular element below the diagonal (in the case of Givens rotations) or an entire column (in the case of Householder matrices). Then we find another orthogonal matrix which wipes out the next element or column, etc. In the end we multiply them all together to form Q .

Anyway, after we have factored A , we can transform the system to one with an upper triangular matrix, and the same solution:

$$\begin{aligned} A\mathbf{x} &= QR\mathbf{x} = \mathbf{b}. \\ R\mathbf{x} &= Q^T\mathbf{b} \end{aligned}$$

We are mostly interested in the non-degenerate case. The matrix R in this case looks like this:

$$R = \begin{pmatrix} r_{11} & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & r_{nn} \\ \vdots & & & 0 \\ 0 & \cdots & \cdots & 0 \end{pmatrix}$$

where $r_{ii} \neq 0$. The rows of R that are all zero do not contribute anything to the solution. We just throw them away and solve the rest.

Example: For the same example as before, the program DQRLS (the double precision version of program SQRLS in the book) produces the following factorization:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2 \\ 1 & 2.5 \end{pmatrix} = QR$$

$$= \begin{pmatrix} -0.5 & 0.670820393250 & 0.023606797750 & 0.547213595500 \\ -0.5 & 0.223606797750 & -0.439344662917 & -0.712022659167 \\ -0.5 & -0.223606797750 & 0.807868932583 & -0.217595468167 \\ -0.5 & -0.670820393250 & -0.392131067417 & 0.382404531833 \end{pmatrix} \begin{pmatrix} -2 & -3.5 \\ 0 & -1.11803398875 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

You can verify that this Q has all the right properties (all columns and rows are orthogonal and have length 1, etc.) Actually, the last two columns of Q are not needed in the following, but I calculated them, anyway. The last two columns of Q are only needed to calculate the last two entries in $Q^T \mathbf{b}$, and we know that those will be thrown away.

The new system is now

$$R\mathbf{x} = Q^T \mathbf{b}$$

$$\begin{pmatrix} -2 & -3.5 \\ 0 & -1.11803398875 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} -4 \\ -1.341640787 \\ -0.415737865 \\ -0.164809064 \end{pmatrix}$$

The triangular system becomes

$$\begin{pmatrix} -2 & -3.5 \\ 0 & -1.11803398875 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} -4 \\ -1.341640787 \end{pmatrix}$$

which gives the same solution $a_0 = -0.1$, $a_1 = -1.2$ as before. \square

If the problem is degenerate, the matrix looks like this:

$$R = \begin{pmatrix} r_{11} & \cdots & \cdots & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & r_{kk} & \cdots & r_{kn} \\ \vdots & & & 0 & \cdots & 0 \\ \vdots & & & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & \cdots & 0 \end{pmatrix}$$

Here again we throw out the zero equations, but there is no unique way of solving the rest. Usually, one sets x_{k+1} through x_n to zero and solves the rest. In other words: we throw away the part of R which is not triangular.

As mentioned earlier, this does not correspond exactly to the solution \mathbf{x} with minimum 2-norm, but it does give a useful and unique answer.

Likewise, if $m < n$, we get a matrix

$$R = \begin{pmatrix} r_{11} & \cdots & \cdots & \cdots & \cdots & r_{1n} \\ 0 & r_{22} & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & r_{mm} & \cdots & r_{mn} \end{pmatrix}$$

and again set x_{m+1} through x_n to zero.

6.5. Subroutine SQRLS

Read this section on your own.

6.6. Historical Perspective: Gauss

Read this section on your own.

6.7. Degenerate Least-Squares Problems

Read this section on your own. I have already incorporated degenerate problems above.

6.8. The Singular Value Decomposition

The ultimate in equation solving is the *singular value decomposition (SVD)*. It is extremely stable and really tells you what is going on, if you know how to use it right. However, it takes 5 to 10 times as long as the normal equations and should be reserved for special cases.

The *singular value decomposition* of a matrix A is given by

$$A = U\Sigma V^T,$$

where U and V are orthogonal, and Σ is diagonal. If A is of size $m \times n$, then U is $m \times m$, V is $n \times n$ and Σ is $m \times n$. Depending on whether m is smaller or larger than n , Σ may look like either

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \sigma_n \\ \vdots & & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \quad \text{or} \quad \Sigma = \begin{pmatrix} \sigma_1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & \sigma_m & 0 & \cdots & 0 \end{pmatrix}$$

All the σ_j are non-negative real numbers, even if the matrix is complex. They are called the *singular values*. Generally, one arranges them by size, so that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

We will assume this ordering from now on.

With this assumption on the ordering, the matrix Σ is uniquely determined. The other matrices U and V are not unique, but that makes no difference, as long as we can find some that work.

The singular values are the square roots of the eigenvalues of $A^T A$ (or $A^H A$ in the complex case). The quotient of the largest and smallest singular value is a good estimate of the condition number of A :

$$\text{cond}(A) \approx \frac{\sigma_1}{\sigma_n}.$$

The matrix A is degenerate if and only if one or more of the singular values are zero. Very small singular values signal a numerical problem. However, once we have the singular value decomposition of a matrix, we can take steps to fix the situation. I will explain this more below.

Let us go back to the least squares problem. We want to solve

$$A\mathbf{x} = \mathbf{b},$$

or equivalently

$$U\Sigma V^T \mathbf{x} = \mathbf{b}.$$

We can do this in two steps:

Step 1. Multiply through by U^T . As before, this will not affect the least squares solution, because U is orthogonal. We get

$$\Sigma V^T \mathbf{x} = U^T \mathbf{b}.$$

Let $\mathbf{y} = V^T \mathbf{x}$, $U^T \mathbf{b} = \mathbf{c}$ and solve the least squares problem

$$\Sigma \mathbf{y} = \mathbf{c}$$

Step 2. Once you have \mathbf{y} , solve the problem

$$V^T \mathbf{x} = \mathbf{y}.$$

This is actually trivial, namely

$$\mathbf{x} = V\mathbf{y}.$$

What we have achieved is to reduce a general least squares problem to one with a diagonal matrix. This is even better than the reduction to a triangular problem we had before, in the QR-algorithm.

Let us consider the general (possibly degenerate) case right away. The diagonal system will look like this:

$$\begin{pmatrix} \sigma_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & \sigma_k & 0 & \cdots & 0 \\ \vdots & & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & \cdots & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix}$$

As before, we have a number of zero equations which we throw away. The remaining equations have the solution

$$\begin{aligned} y_1 &= c_1/\sigma_1 \\ y_2 &= c_2/\sigma_2 \\ &\dots\dots \\ y_k &= c_k/\sigma_k \\ y_{k+1} &= \text{arbitrary} \\ &\dots\dots \\ y_n &= \text{arbitrary} \end{aligned}$$

If we want to pick a unique solution by choosing the \mathbf{y} with the smallest norm, there is only one way to do that: set y_{k+1} through y_n to zero. This gives you the true least squares solution, in the sense defined before (minimize the residual first; if there are infinitely many solutions, pick the smallest one).

Example: Consider the same example as before, with

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.5 \\ 1 & 2 \\ 1 & 2.5 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \end{pmatrix}.$$

Subroutine DSVDC (**D**ouble precision **S**ingular **V**alue **D**e**C**omposition) from LINPACK gives the following:

$$U = \begin{pmatrix} -.325868337856 & .770590569877 & .023606797750 & .547213595500 \\ -.432366606927 & .336242646335 & -.439344662917 & -.712022659167 \\ -.538864875998 & -.098105277206 & .807868932583 & -.217595468167 \\ -.645363145069 & -.532453200748 & -.392131067417 & .382404531833 \end{pmatrix},$$

$$\Sigma = \begin{pmatrix} 4.148428878243 & 0 \\ 0 & .539015623295 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad V^T = \begin{pmatrix} -.468240633469 & -.883600989796 \\ .883600989796 & -.468240633469 \end{pmatrix}$$

Again, the last two columns of U are not really needed. We find

$$\mathbf{c} = U^T \mathbf{b} = \begin{pmatrix} -4.204420738913 \\ -0.350494294107 \\ -0.415737865167 \\ -0.164809063667 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} -1.013497124408 \\ -0.650248859142 \end{pmatrix},$$

and finally

$$\mathbf{x} = V \mathbf{y} = \begin{pmatrix} -0.1 \\ 1.2 \end{pmatrix}$$

as before. \square

To understand why small singular values are bad, consider the example

$$\begin{pmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 10^{-8} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

The solution is obviously

$$y_1 = c_1/10,$$

$$y_2 = c_2$$

$$y_3 = 10^8 c_3$$

Whatever error was contained in c_3 has been magnified by a huge factor and is now probably larger than the solution itself.

The same thing would have happened with the original matrix A , before you did a SVD, but you would not have been able to see it. Now we have rotated the coordinate system so that the “bad” part points in the direction of \mathbf{e}_3 , and we see exactly what the problem is. In this particular case, it would be better to set y_3 to zero.

There is an analogy between this approach and filtering of signals in Electrical Engineering:

The Fourier transform decomposes a signal into its frequency components. The noise is contained mostly in the high frequency components, so you can try to cut those off. You lose some of the fine detail of the signal, but you get rid of a lot of the noise. Where exactly you cut depends on the nature of the signal.

The singular value decomposition decomposes the matrix into its effect on various directions. The error comes mostly from the very small singular values, so you can try to cut those off. You lose some fine detail in the solution, but you get rid of a lot of the error. Where exactly you cut depends on the nature of the problem.

Note that we can sometimes significantly lower the condition number of the rest of the matrix by throwing out a few of the very small singular values. In the above example, getting rid of σ_3 lowers the condition number from 10^9 to 10. In this way, you can still get a halfway satisfactory solution out of a very ill-conditioned system. However, the singular value decomposition is much more computer intensive than other methods and should be used sparingly.

6.8.1. Summary. We have discussed three methods for finding the least squares solution of a system of linear equations. They were

- The method of normal equations. This is the fastest method, but it has two drawbacks: it only works for non-degenerate, overdetermined systems, and it often is numerically unstable.
- QR factorization. This method is recommended for most cases. It can handle all cases and is usually stable. It takes about twice as much computer time as the method of normal equations.
- SVD. This is for cases where QR factorization fails. It takes five to ten times more computer time than the normal equations, but is extremely stable. It can also be used for other purposes, for example when Gaussian elimination fails.