LEAST-SQUARES AND CHI-SQUARE FOR THE BUDDING AFICIONADO: ART AND PRACTICE

0. USING SINGULAR VALUE DECOMPOSITION (SVD)

Occasionally, a normal-equation matrix $[\alpha] = \mathbf{X^T} \cdot \mathbf{X}$ is degenerate, or at least sufficiently ill-posed that inverting it using standard matrix inversion doesn't work. In its `invert` function, IDL even provides a keyword called `status` to check on this (although I find that it is not perfectly reliable; the best indicator of reliability is to check that the matrix product $[\alpha^{-\mathbf{1}}] \cdot [\alpha] = \mathbf{I}$, the unitary matrix). In these cases, Singular Value Decomposition (SVD) comes to the rescue.

First, we reiterate the least-squares problem. Least squares begins with equations of condition (equation **??**), which are expressed in matrix form as

$$\mathbf{X} \cdot \mathbf{a} = \mathbf{y} \tag{0.1}$$

In our treatments above we premultiply both sides by $\mathbf{X^T}$, on the left generating the curvature matrix $[\alpha]$ to obtain the normal equations (equation **??** or its equivalent derived from equations **??**)

$$[\alpha]{\cdot}\mathbf{a} = \mathbf{X^T} \cdot \mathbf{y} \ , \tag{0.2}$$

for which the solution is, of course,

$$\mathbf{a} = \left([\alpha]^{-1}{\cdot}\mathbf{X^T}\right) {\cdot}\mathbf{y} \tag{0.3}$$

We need to find the inverse matrix $[\alpha]^{-1}$. Above in this document we did this using simple matrix inversion, which doesn't work if $[\alpha]$ is degenerate.

SVD provides a bombproof and interestingly informative way do least squares. Perhaps surprising, with SVD you don't form normal equations. Rather, you solve for the coefficients directly. In essence, SVD provides the combination $\left([\alpha]^{-1}{\cdot}\mathbf{X^T}\right)$ without taking any inverses. By itself, this doesn't prevent blowup degenerate cases; however, SVD provides a straightforward way to eliminate the blowup and get reasonable solutions.

For a discussion of the details of SVD, see NR §2.6; for SVD applied to least squares, see NR §15.4; if you are rusty on matrix algebra, look at NR §11.0. Below, we provide a brief description of SVD. Implementing SVD in our least-squares solutions is trivially easy, and we provide the IDL prescription below in §0.3 and §0.4. *Be sure* to look at §0.2.1!!

0.1. Phenomenological description of SVD

The cornerstone of SVD is that our (or any) $M \times N$ (M rows, N columns) matrix $\mathbf{X}$, where $M \geq N$, can be expressed as a product of three matrices:

$$\mathbf{X} = \mathbf{U} \cdot [w] \cdot \mathbf{V^T} \ , \tag{0.4}$$

where

1. $\mathbf{U}$ is $M \times N$, $[w]$ is $N \times N$ and diagonal, and $\mathbf{V}$ is $N \times N$; and
2. the columns of $\mathbf{U}$ and $\mathbf{V}$ are unit vectors that are orthonormal. Because $\mathbf{V}$ is square, its rows are also orthonormal so that $\mathbf{V} \cdot \mathbf{V^T} = 1$. Recall that, for square orthonormal vectors, the transpose equals the inverse so $\mathbf{V^T} = \mathbf{V^{-1}}$.

So what?

Consider a single *row* m of the matrix $\mathbf{X}$. This row has a single value of x_m, with an associated value of y_m. For this row, we can consider the entry of each column n of $\mathbf{X}$ to be a basis function $f_n(x_m)$, evaluated at x_m. The N columns contain N basis functions. And, of course, we have N coefficients in the vector $\mathbf{a}$. These N basis functions and associated coefficients represent the entire set of M measurements y_m taken at positions x_m.

Now consider a single *column* n of the matrix $\mathbf{X}$. This column consists of the entire set of M values of $f_n(x_m)$ for a single value of n. We can regard the M values of x_m to be a vector of length M. Similarly, we can regard each column n of $\mathbf{X}$ to be an M-length vector consisting of the elements $f_n(x_m)$.

Finally, consider the *set of N columns* of the matrix $\mathbf{X}$. Each column n is an M-element vector with elements $f_n(x_m)$. Now, there is no reason for these N column vectors of $\mathbf{X}$ to have any special property, such as being orthogonal. In fact, consider a typical least-squares polynomial fit; this basis set is certainly *not* orthogonal, as we illustrated in our simple numerical example of §**??**. What SVD does is to replace the set of N original nonorthogonal vectors $\mathbf{X}$ by an orthogonal basis set that consists of the N rows of $\mathbf{V}$—in fact, they are not only orthogonal, but orthonormal.

0.2. Using SVD for Least Squares

Some of the original nonorthogonal column vectors of $\mathbf{X}$ might not only be nonorthogonal, they might be degenerate! This happens if the particular set of the M values x_m make some of the original nonorthogonal vectors linear combinations of others. This leads to the unfortunate situation of the curvature matrix α not having an inverse. In such cases, SVD comes to the rescue.

With SVD you *don't form normal equations*, so you don't explicitly calculate $[\alpha]^{-1}$. We apply SVD to $\mathbf{X}$, as in equation 0.4:

$$\mathbf{X} = \mathbf{U}\cdot[w]\cdot\mathbf{V^T} \ , \tag{0.5}$$

NR §15.4 shows that the covariance matrix is given by

$$[\alpha^{-1}] = \mathbf{V}\cdot\left[\frac{1}{w^2}\right]\cdot\mathbf{V^T} \tag{0.6}$$

Given this, the fact that $\mathbf{V^T}\cdot\mathbf{V} = \mathbf{I}$, and the general matrix identity $(\mathbf{P}\cdot\mathbf{Q})^\mathbf{T} = \mathbf{Q^T}\cdot\mathbf{P^T}$ it's straightforward to show that

$$\mathbf{a} = \left(\mathbf{V}\cdot\left[\frac{1}{w}\right]\cdot\mathbf{U^T}\right)\cdot\mathbf{y} \tag{0.7a}$$

or

$$\mathbf{a} = \left(\mathbf{V}\cdot\left[\frac{\mathbf{1}}{\mathbf{w}}\right]\right)\cdot\left(\mathbf{U^T}\cdot\mathbf{y}\right) \tag{0.7b}$$

Here in equation 0.7a, we see that $\left(\mathbf{V}\cdot\left[\frac{1}{w}\right]\cdot\mathbf{U^T}\right)$ is identical to $\left([\alpha]^{-1}\cdot\mathbf{X^T}\right)$ in equation 0.3. In the rewritten equation 0.7b we see that we can regard the coefficient vector being defined by the orthogonal vectors of $\mathbf{V}$. Moreover, the covariance matrix of equation 0.6 is diagonal, so that the principal axes of the χ^2 ellipse are defined by the orthonormal column vectors of $\mathbf{V}$ with lengths proportional to $\left[\frac{1}{w^2}\right]$. See NR Figure 15.6.5 and the associated discussion.

The N orthonormal vectors in $\mathbf{V}$ define an N-dimensional space for $\mathbf{a}$ with N orthogonal directions. Suppose that a particular value w_n is small. In equation 0.5, this means that the associated orthonormal vector v_n—i.e., the associated direction n—in $\mathbf{V}$ is not well-represented by the set of original nonorthogonal vectors in $\mathbf{X}$. This, in turn, means that you can't represent that direction in $\mathbf{a}$ of equation 0.7a without amplifying that orthonormal vector by a large factor; these amplification factors $\propto w_n^{-1}$. This is an unsatisfactory situation because it means some combinations of the original m measurements are highly weighted. As $w_n \rightarrow 0$ this situation becomes not only unsatisfactory, but numerically impossible.

Consider the limiting case, $w_n = 0$. In this case, the original x_m values do not have any projection along associated orthonormal vector v_n in $\mathbf{V}$. (These v_n are called "null" orthonormal vectors.) In equation 0.7a, you can add any multiple of the null v_n to the solution for $\mathbf{a}$ and it won't change $\mathbf{a}$ at all (!) because it has absolutely no effect on the fit to the data (because of the particular set of values x_m).

What multiple is appropriate? Common sense says that, because these $\mathbf{V}$ vectors have no meaning for the solution, the multiple should be *zero*. So in equation 0.7a, instead of trying, in vain, to include this null vector v_n by using a huge multiple, you toss in the towel and eliminate it altogether by replacing its corresponding $w_n^{-1} = \infty$ by $w_n^{-1} = 0$. So we have the rule: *wherever* $w_n = 0$ *(or sufficiently small), replace* w_n^{-1} *by 0!* This replacement provides the minimum length for $\mathbf{x}$ and thereby constitutes the least-squares solution.

0.2.1. Important Conclusion for Least Squares!!!

Suppose you have degeneracy, or near-degeneracy. *This, in turn, means that the formulation of the least-squares model is faulty*: some of the original basis functions represent (or *nearly*) represent the same physical quantity, or at least one of the functions is (nearly) a linear combination of others. Sometimes you can discover the problem and fix it by imposing additional constraints, or by finding two unknown coefficients that are nearly identical. If so, you can reformulate the model and try again.

Even if you can't discover the root cause(s) and remove the degeneracy, the SVD solution allows you to bypass problems associated with degeneracy and provides reasonable best-fit parameter values.

0.2.2. How Small is "Small"?

When looking at w_n, just exactly how small is "small"?

0.2.3. Strictly Speaking...

Strictly speaking, this is governed by numerical machine accuracy. NR suggests measuring this by the ratio of the largest w_n to the smallest. This ratio is the *condition number*. Single precision 32-bit floats have accuracy $\sim 10^{-6}$, so if the condition number is larger than 10^6 you certainly have problems.

0.2.4. Practically Speaking...

Practically speaking it's not machine accuracy that counts. Rather, it's the accuracy of your data and the ability of those uncertain data to define the parameters. In any real problem, plot the vector w. If the values span a large range, then the data with small w_n might not be defining the associated vectors well enough. If so, then *practically* speaking, these vectors are degenerate.

How to tell what constitutes "small w_n"? As far as I know, and as far as NR says, it's an art.

0.3. Doing SVD in IDL

Doing SVD in IDL is easy. Ida's `la_svd` procedure[1] provides the SVD decomposition. Thus for equation 0.4, the IDL SVD decomposition is given by

$$\mathbf{la\_svd}, \mathbf{X}, \mathbf{w}, \mathbf{U}, \mathbf{V} \ ; \tag{0.8}$$

the notation is identical to that in equation 0.4.

0.4. My routine *lsfit_svd*

I've written an IDL routine `lsfit_svd` that makes the above process of dealing with the weights easy. When used without additional inputs, it returns the standard least-squares results such as the derived coefficients and the covariance matrix; it also returns $[w]$, $\mathbf{U}$, and $\mathbf{V}$. It allows you to input those matrices and, also, the $\left[\frac{1}{w}\right]$ matrix so that you can tailor the weights to your heart's content.

[1] Old versions of IDL (before 5.6) had a significantly worse algorithm called *svdc*. Don't use this unless you have to.