CS 542G: Least Squares, Normal Equations

Robert Bridson

October 1, 2008

1 Least Squares

Last time we set out to tackle the problem of approximating a function that has been sampled with errors. To avoid the potentially non-smooth error from introducing wiggles in a smooth interpolant, our new approach is as follows:

- restrict the space of possible functions to well-behaved, non-wiggly, plausible candidates,
- and from that space, choose the one that fits the data best.

To get to a result, we need to better define both of these steps: what function space should we choose, and how do we define the "best" fit to the data given that it won't exactly interpolate?

1.1 Best Fit

For now, let's call the set of allowable functions V without further specification. Instead, we'll go on to address the second issue, defining how well a function from V fits the data. We'll first define the **residual** r, which is just the vector of differences between the measured values $\{f_i\}$ and what our function $f(x)$ estimates them to be:

$$r_i = f_i - f(x_i), \quad i = 1, \ldots, n$$

If this vector is small, we have a good fit. The obvious way to make this precise is to choose a norm to use in measuring r: the best fit function is one that minimizes $\|r\|$.

(As an aside, there are more sophisticated approaches than simply using a norm. For example, in many applications there may be **outliers** present, measurements which were completely wrong. A robust

method for approximating the data should detect and then filter out (ignore) these useless data points. However, often the first step on the way to detecting the outliers is to minimize a norm of the entire residual vector and then look for suspiciously large entries in the resulting r.)

There are many norms to choose from, of course. The 1-norm in particular is rather interesting, since it naturally is fairly robust to bad errors. An interesting exercise is to prove that if $f(x)$ is restricted to be a constant, the constant which minimizes the 1-norm of the residual is the **median** of the data values—which is well known in statistics to be useful, since outliers often have little effect on it.

However, it turns out the 2-norm is the simplest norm to work with and is therefore often preferred.[1] The main reason is that the 2-norm is smoothly differentiable with respect to r, whereas the 1-norm is not thanks to the absolute values.

Writing this out, and noting that minimizing the square of a norm is the same as minimizing the norm, our problem is:

$$\begin{aligned} \min_{f \in V} \|r\|_2^2 &\Leftrightarrow \min_{f \in V} \sum_{j=1}^{n} r_j^2 \\ &\Leftrightarrow \min_{f \in V} \sum_{j=1}^{n} (f_j - f(x_j))^2 \end{aligned}$$

We are minimizing a sum of squares, hence the usual name **least squares**.

1.2 The Choice of Function Space

Returning to the question of what V is: for now, we'll assume V is a vector space, i.e. if functions f and g are in V and α is a real scalar then the function $\alpha f + g$ is also in V. This gives rise to **linear least squares** (which should not be confused with choosing V to contain linear functions!).

We can then represent V with a basis, a set of linear independent functions that span all of V. Again, V should be chosen with the application in mind. A very common choice when little is known is to use low degree polynomials: for V the space of all quadratics in 1D, a simple basis could be $\{1, x, x^2\}$. In general, we'll assume V has dimension k and we have selected k basis functions, $\phi_1(x), \ldots, \phi_k(x)$.

Note, right off the bat, that k shouldn't be larger than the number of sample points n: otherwise there's no hope of getting a well-posed problem, since infinitely many functions from V will be able to approximate the data equally well. In most least squares problem, k is significantly smaller than n.

[1] The 2-norm, or slight variations of it, also is the one that pops up most commonly in physical applications, and that we've already seen in the context of deriving RBFs, minimizing a roughness measure involving the integral of the square of a differential quantity.

Any function from V now can be represented as a linear combination of these basis functions:

$$f(x) = \sum_{i=1}^{k} \alpha_i \phi_i(x)$$

All we have to do is determine the coefficients $\alpha_1, \ldots, \alpha_k$.

1.3 Solving Least Squares the Obvious Way

We can now finally find the best fit function, with the usual calculus approach of differentiating $\|r\|^2$ with respect to the coefficients, setting these derivatives to zero, and solving these equations for the coefficients. Let's work it out, taking the derivative with respect to α_p (for $p = 1, \ldots, k$):

$$\begin{aligned}
0 &= \frac{\partial \|r\|^2}{\partial \alpha_p} \\
&= \frac{\partial}{\partial \alpha_p} \sum_{j=1}^{n} (f_j - f(x_j))^2 \\
&= \sum_{j=1}^{n} 2(f_j - f(x_j)) \left(-\frac{\partial f(x_j)}{\partial \alpha_p} \right) \\
&= \sum_{j=1}^{n} -2(f_j - f(x_j))\phi_p(x_j) \\
&= \sum_{j=1}^{n} -2 \left(f_j - \sum_{i=1}^{k} \alpha_i \phi_i(x_j) \right) \phi_p(x_j)
\end{aligned}$$

Dividing by two, switching the order of the nested sums, and rearranging what's left gives:

$$\sum_{i=1}^{k} \left(\sum_{j=1}^{n} \phi_i(x_j)\phi_p(x_j) \right) \alpha_i = \sum_{j=1}^{n} \phi_p(x_j) f_j$$

This is a linear equation. In fact, if we define matrix $B \in \mathbf{R}^{k \times k}$ and vector $c \in \mathbf{R}^k$ as follows,

$$\begin{aligned}
B_{pi} &= \sum_{j=1}^{n} \phi_i(x_j)\phi_p(x_j) \\
c_p &= \sum_{j=1}^{n} \phi_p(x_j) f_j
\end{aligned}$$

then this is just the p'th row of the linear system $B\alpha = c$.

The matrix B is clearly symmetric: $B_{pi} = B_{ip}$. Therefore we can expect to be able to use something more efficient than LU factorization to solve the system. However, it's not immediately obvious if B is SPD, which would be a big advantage.

1.4 The Normal Equations

In some sense we have now solved the problem—we can construct B and c, and hand them to a LAPACK routine which can handle arbitrary symmetric matrices. However, as with other problems we've seen in this course (interpolating data, solving linear systems) we can do better than the obvious "operational mindset" approach (jumping straight into algorithmic details) by viewing the problem from a higher level. In particular, here we will introduce matrices earlier on in the problem when we first define the residual.

The residual vector, in linear least squares, is defined from:

$$\begin{aligned} r_i &= f_i - f(x_i) \\ &= f_i - \sum_{j=1}^{k} \phi_j(x_i)\alpha_j \end{aligned}$$

Define the vector $f \in \mathbf{R}^n$ from the measured data values $(f_1, f_2, \ldots, f_n)$ and the matrix $A \in \mathbb{R}^{n \times k}$ as:

$$A_{ij} = \phi_j(x_i)$$

Then the residual vector is simply $r = f - A\alpha$.

The problem is now:

$$\min_{\alpha} \|f - A\alpha\|^2$$

We can expand the 2-norm out as a dot-product (expressed with transposes and matrix multiplication):

$$\begin{aligned} \|f - A\alpha\|^2 &= (f - A\alpha)^T (f - A\alpha) \\ &= f^T f - f^T A\alpha - \alpha^T A^T f + \alpha^T A^T A\alpha \end{aligned}$$

Finally, setting the gradient[2] with respect to α of this expression to zero gives us:

$$\begin{aligned} -2A^T f + 2A^T A\alpha &= 0 \\ \Leftrightarrow \quad A^T A\alpha &= A^T f \end{aligned}$$

This is the same linear system as before, but now we see the coefficient matrix B is in fact $A^T A$. As long as A has full column rank—i.e. when we evaluate the linearly independent basis functions of V at the n sample points we still have linearly independent vectors, making up the columns of A—then this matrix must be SPD. That allows Cholesky factorization, which should make us happy!

This linear system has a special name, the **normal equations**. It is the most direct way of solving a linear least squares problem, and as long as $A^T A$ is reasonably well conditioned is a great method.

[2] You may be uncomfortable with differentiating expressions such as this with respect to vectors; you can always write out the products and do it entry by entry if you're worried.

1.5 Conditioning Problems

The normal equations are sometimes ill-conditioned however, worse than an error analysis of the original least squares problem might warrant. We'll take a look at a simple example, with A just 2×2:

$$A = \begin{pmatrix} 1 + 10^{-8} & -1 \\ -1 & 1 \end{pmatrix}$$

Here A is square and invertible, so the least squares problem actually has an exact fit $\alpha = A^{-1}f$ as its solution. The condition number of A is on the order of $4 \cdot 10^8$, large but still perfectly workable with double precision floating point.

However, the normal equations use the matrix A^TA, which in this case is:

$$A^TA = \begin{pmatrix} 2 + 2 \cdot 10^{-8} + 10^{-16} & -2 - 10^{-8} \\ -2 - 10^{-8} & 2 \end{pmatrix}$$

This matrix has condition number on the order of $16 \cdot 10^{16}$, which means the normal equations could fail dismally even in double precision arithmetic.

We can get a feeling for what's going on (at least for square matrices) by estimating the condition number as follows:

$$\begin{aligned} \kappa(A^TA) &= \|A^TA\|\|(A^TA)^{-1}\| \\ &= \|A^TA\|\|A^{-1}A^{-T}\| \\ &\sim \|A^T\|\|A\|\|A^{-1}\|\|A^{-T}\| \\ &\sim \|A\|^2\|A^{-1}\|^2 \\ &= \kappa(A)^2 \end{aligned}$$

Depending on the choice of matrix norm, the approximations $\|A^TA\| \sim \|A\|\|A^T\|$ and so forth may or may not be exact, but will always be fairly good. This means we can expect the condition number of the normal equations to be the square of the condition number of A (whatever that might mean in general, when A is rectangular), and that could easily get us into dangerous territory. Again, I want to underscore that if the problem is adequately well-conditioned, the normal equations approach works well; it's just not as stable as we would require for tougher problems.

1.6 Orthogonalizing

In search of a better algorithm, let's take a look at the ideal case for the normal equations. The best linear system in the world is one where the matrix is the identity, so we ideally want an A where $A^TA = I$. This

is equivalent to saying the colums of A should be orthonormal.

Of course, the A we are given in a real problem probably won't have orthonormal columns. However, note that each column of A was simply a basis function from V evaluated at the sample points; we could easily pick a different basis for V to arrive at a different A without changing the original approximation problem. In fact, we can do this at the matrix level: find a different set of k column vectors that span the same space as the columns of A, but happen to be orthonormal—these will have to correspond to just a different choice of basis for V.

The classic linear algebra procedure for finding a set of orthonormal vectors that span the same space as another set of vectors is **Gram-Schmidt**. Let a_i be the i'th column of A: we produce an orthonormal set of vectors $\{q_i\}$ with the same span as follows:

- $q_1 = \frac{a_1}{\|a_1\|}$, scaling the first column of A to unit 2-norm
- $\tilde{q}_2 = a_2 - q_1(q_1 \cdot a_2)$, projecting out components of a_2 in the span of the previous vector, and then $q_2 = \frac{\tilde{q}_2}{\|\tilde{q}_2}$ to get back to unit length
- ...
- $\tilde{q}_i = a_i - \sum_{j=1}^{i-1} q_j(q_j \cdot a_i)$, then $q_i = \frac{\tilde{q}_i}{\|\tilde{q}_i}$
- ...

At the i'th step, we project out components of a_i in the directions of all the earlier vectors, arriving at an intermediate $\tilde{q}_i$, and then normalize this to get a unit length q_i orthogonal to all the previous q_j vectors.

Putting these q vectors together as the columns of an $n \times k$ matrix Q, the linear least squares problem reduces to

$$\min_{\beta} \|f - Q\beta\|^2$$

Note that since we are implicitly using a different basis for V, I've switched from the coefficients α defining the solution in terms of the original basis functions to a new vector of coefficients β.

The discrete least square problem can be solved with the normal equations trivially:

$$\begin{aligned} Q^T Q\beta &= Q^T f \\ \Leftrightarrow \quad \beta &= Q^T f \end{aligned}$$

The remaining question is how to get back α in terms of β. The new solution β tells us the optimal linear combination of the q vectors, which are themselves linear combinations of the a vectors: we need

to collapse this relationship to get the optimal linear combination of the a vectors. Luckily, the Gram-Schmidt process spells out for us what the linear combinations relating q and a vectors are—or rather, looking closer, it tells us the linear combinations of q vectors that give us the a vectors. We'll work out the details next lecture.