Curve Fitting

November 19, 2015

We have seen that if we have a large number of data points of the form (x_i, y_i) where we expect $y_i = \sin(x_i) + \epsilon_i)$, where ϵ_i are some small random errors, then polynomial interpolation does not do a very good job of approximating the sine function. This is because, we are trying to do an *exact* fit instead of a "best" fit. As a slightly different example, consider what would have happened if Kepler were trying to find an *exact* fit to all the positions of a given planet; he would certainly not have got a curve of degree 2!

Thus, there are situations, where we have a "model" that we expect the data to conform to. There are *parameters* in the model and we would like to find those values of these parameters that give the "closest" fit to the data. We have used the words "best", "model" and "closest" without any explanation, so we not try to explain these.

Let $\mathcal{P}$ denote the space of parameters. We have a map $\phi : \mathcal{P} \to \mathcal{M}$ where the latter is the space of models. Now, $\mathcal{M}$ can be identified with a subspace of the space of functions from x-values to the space of y-values, where each of these can be vector values as well. On this space of functions we put a metric which we use to measure which solution is the closest and call it the best fit.

Less abstractly, suppose that we expect the function to look like $f(x) = \sum_{i=j}^{m} a_j \phi_j(x)$; here a_j are the parameters. This is an example of a linear model; even when the functions $\phi_j(x)$ are non-linear! Moreover, a natural metric on the space of functions is the distance of the actual values $(f(x_j))_{i=1}^n$ from the "experimental" values (y_i), given by the formula

$$\left(\sum_{i=1}^{n} (y_i - f(x_i))^2 \right)^{\frac{1}{2}}$$

This is the RMS or root-mean-square or l^2 norm.

This can be turned into a problem in linear algebra as follows. Consider the vectors $w_j = (\phi_j(x_i))_{i=1}^n$ for $j = 1, \ldots, m$; these are m vectors in $\mathbb{R}^n$.

On the other hand we have the vector $y = (y_i)_{i=1}^n$ in the same space. The problem is thus to find the vector u in the *linear span* of the w_j so that y is closest to u in the usual sense of distance in $\mathbb{R}^n$. It follows that this happens when $(y-u)\cdot w_j = 0$ for all j.

Let us assume that e_j is a collection of orthogonal vectors that have the same linear span as w_j. It follows that $w = \sum_{j=1}^m b_j w_j$. We can then determine b_j using the equation $y \cdot e_j = w \cdot e_j = b_j e_j \cdot e_j$.

A Gram-Schmidt type process called the conjugate-gradient method can be used to determine such a collection of orthogonal vectors iteratively as follows.

Let e_1 be a vector among the w_j that has non-zero length (in order to reduce numerical errors, we usually take the *longest* vector). We then calculate

$$u_1 = y - \frac{y \cdot e_1}{e_1 \cdot e_1} e_1$$

We note that $y_1 \cdot e_1 = 0$.

Induuctively, let us assume that we have found $e_1, \ldots e_k$ so and u_k so that u_k is a linear combination of y with $e_1,\ldots, e_k$ which is orthogonal to these latter vectors. Moreover, e_i is a linear combination of $w_1,\ldots w_i$. For each $j > k$, we now take vectors of the form

$$t_j = w_j - \sum_{l=1}^{k} \frac{w_j \cdot e_l}{e_l \cdot e_l} e_l$$

and let e_{k+1} be a non-zero vector from among them (or the longest one for numerical stability). We can then define

$$u_{k+1} = u_k - \frac{u_k \cdot e_{k+1}}{e_{k+1} \cdot e_{k+1}} e_{k+1}$$

This completes the inductive step.

We note that the calculation only involves the values y_i and the matrix $A = (\phi_j(x_i))$. Thus, it is possible to represent the entire calculation rather easily on a computer. Moreover, the parameters a_j can be inductively computed as well by always representating of e_j in terms of the matrix that converts the w_j to this basis. The task of doing this is left as an exercise.

1 Non-linear case

In case of a nonlinear model of the form $\phi(a_1, \ldots, a_m; x)$, we use a method that is geometrically somewhat similar to the method above!

We first note that we have a map $\mathbb{R}^m \to \mathbb{R}^n$ given by

$$\Phi : (a_1, \ldots, a_m) \mapsto (\phi(a_1, \ldots, a_m, x_i))_{i=1}^n$$

Loosely speaking, we can think of the image $\mathcal{M}$ as the "space of models". It is actually the evaluation of these models at the chosen x-values. However, we can imagine that we have chosen enough x-values to determine the model uniquely. Our problem is then to determine a point on $\mathcal{M}$ that is closest (in the usual metric on $\mathbb{R}^n$) to the point y.

Let us say that $v_p = (b_1, \ldots, b_m)$ is a guess for the values of the parameters for the model that fits the data. Consider the matrix of partial derivatives

$$A_p = \left(\frac{\partial \phi}{\partial a_j}(x_i)_{|a_l = b_l; \forall l} \right)$$

The tangent space to the above space $\mathcal{M}$ at $\Phi(v_p)$ is the collection of points in $\mathbb{R}^n$ of the form $Aw + \Phi(v_p)$ as w varies over $\mathbb{R}^m$. Suppose we find a point u on this tangent space which is closest to y; moreover, let us assume that u is the image of w under the above transformation. Our next guess is then $v_{p+1} = v_p + w$; note that this would be the exact solution to the problem if Φ depends linearly on the parameters. (This method is more or less that same as Newton's method for finding the root of a non-linear equation except for an increase in the number of variables!) The problem of finding w (and u) is what was solved above in the linear parametric case.

Just as in the case of Newton's method, the convergence of the v_p's depends on a "good" initial guess for the parameters and nice behaviour of the function Φ.