1 Introduction

Given a set of observations, one often wants to condense and summarize the data by fitting it to a “model” that depends on adjustable parameters. Sometimes the model is simply a class of functions, such as polynomials or trigonometric functions, and the fit determines the appropriate coefficients. Other times, the model’s parameters come from underlying theory that the data are supposed to satisfy. Modeling can also be used as a kind of constrained interpolation, where you want to extend a few data points into a continuous function, but with some underlying idea of what the function should look like.

Imagine that we have taken measurements of some unknown function \( f \) at points \( x_1, \ldots, x_n \). We would like to “reconstruct” this function to our best. Let us refer to the \( n \) measurements as \( f_1, \ldots, f_n \). If these measurements were perfect, then \( f_i = f(x_i) \) would hold for \( i = 1, \ldots, n \). Generally, they are subject to measurement errors (for instance, noise in the context of signal processing) so that

\[
f_i = f(x_i) + \epsilon_i,
\]

where \( \epsilon_i \) is some unknown error. Nevertheless, we would like to try to recover \( f(x) \). Were it not for the measurement errors, we might consider using interpolation. But it does not make sense to force an interpolating polynomial to pass through points \( (x_i, f_i) \) that are errant. The polynomial may just wiggle around a lot to pass through all the points \( (x_i, f_i) \) and thus have an order much higher than \( f(x) \) really has.

**Example 1.** Suppose we have taken measurements of the function \( f(x) = (x - 1)^2 \) at 7 points, evenly spaced over the interval \([-1, 2]\). The following table lists the points \( x_i \), the true values \( f(x_i) \), and the (errant) measurements \( f_i \).

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
<th>( f_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>4</td>
<td>4.1</td>
</tr>
<tr>
<td>(-\frac{1}{2})</td>
<td>2.25</td>
<td>2.3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1.05</td>
</tr>
<tr>
<td>( \frac{1}{2} )</td>
<td>.25</td>
<td>.20</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>.05</td>
</tr>
<tr>
<td>( \frac{3}{2} )</td>
<td>.25</td>
<td>.26</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>.90</td>
</tr>
</tbody>
</table>

The function \( f(x) \) is a quadratic. If we interpolated at the supporting points \( \{(x_i, f_i)\} \) we would get a 6th order polynomial. No doubt that this polynomial would contain some twists and turns that \( f(x) \) does not. In the general case, the interpolating polynomial matches \( f(x) \) well at the supporting points, but its derivatives may not.

Of course, there are many functions \( \{g(x)\} \) that could produce the measured values \( f_i \) above. How could we possibly hope to recover the correct function? Without further information, we cannot expect to recover the correct one.
Suppose, however, that we know that the underlying function is quadratic. Then we could pick the following as basis functions
\[1, \quad x, \quad x^2\]
and seek to find coefficients \(a, b, c\) such that
\[f(x) = a \cdot 1 + b \cdot x + c \cdot x^2.\]
If our measurements \(f_i\) were perfect, then the system of equations
\[f_i = a + bx_i + cx_i^2, \quad i = 1, \ldots, 7,\]
in the variables \(a, b, c\) would have a unique solution.

If the \(f_i\) are imperfect, then the system (1) is generally overconstrained. In this case, we can still obtain a least-squares solution, using for example, the technique of singular value decomposition.

In the general version of data fitting, we are given \(n\) measurements \((x_i, f_i)\). And we would like to reconstruct the function \(f(x)\). What we have is some parametrized family of functions:
\[F(x) = F(x; c_1, \ldots, c_k).\]
We then choose the parameters \(c_1, \ldots, c_k\) based on the observations \(\{(x_i, f_i)\}\) in such a way that \(F(x)\) is “close to” \(f(x)\). Often, for mathematical simplicity we write \(F(x)\) as a linear combination of some \(k\) basis functions:
\[F(x) = c_1 \phi_1(x) + \cdots + c_k \phi_k(x).\]
Here \(k\) should be large enough so that the information about \(f(x)\) can be represented by the choice of \(c_1, \ldots, c_k\) while in the meantime it should be small enough to avoid reproduction of noise. Our objective is to choose the coefficients \(c_1, \ldots, c_k\) well.

Why do we do things this way? The point is that \(k\) will in general be much smaller than \(n\). So rather than retain all \(n\) pieces of data (as with interpolation), most of which really contains little information, we try to extract the important or useful information. That information is encoded in the \(k\) basis functions.

## 2 Least Squares

How are the coefficients \(\{c_i\}\) chosen? Ideally, we would like to minimize the difference between \(f(x)\) and \(F(x)\). Unfortunately, we do not know \(f(x)\), so instead we simply minimize the difference between the two functions at the data points \(x_1, \ldots, x_n\). There are a variety of norms by which to measure the error:

- \(\infty\)-norm: \[\|f - F\|_\infty = \max_{1\leq i\leq n} |f_i - F(x_i)|,\]
- \(1\)-norm: \[\|f - F\|_1 = \sum_{i=1}^{n} |f_i - F(x_i)|,\]
- \(p\)-norm: \[\|f - F\|_p = \left(\sum_{i=1}^{n} |f_i - F(x_i)|^p\right)^{1/p}.\]

1Other basis functions are possible and indeed sometimes desirable, as long as they are independent and span the vector space of quadratic functions. Later we will also look at orthogonal bases.
Recall that the parameters \( \{c_j\} \) are hidden in this notation. In fact, \( F(x_i) = F(x_i; c_1, \ldots, c_k) \).

For each norm, the goal would be to choose these parameters to minimize the error. Such a minimization process would lead to solution of nonlinear equations in \( c_1, \ldots, c_k \), even when \( F(x) \) has the simple linear form

\[
F(x) = c_1 \phi_1(x) + \cdots + c_k \phi_k(x).
\]

(2)

However, as we shall see, in the case of a 2-norm with \( F(x) \) having the linear form, error minimization leads to linear equations that determine \( c_1, \ldots, c_k \). For this practical reason, 2-norms, that is, least squares, are so popular.

More formally, we want to choose \( \{c_j\} \) to minimize the quantity

\[
\|f - F\|_2 = \left( \sum_{i=1}^{n} \left| f_i - F(x_i; c_1, \ldots, c_k) \right| \right)^2.
\]

Equivalently, we are choosing

\[
c = \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix}
\]

to minimize the function

\[
E(c) = \sum_{i=1}^{n} \left( f_i - F(x_i; c) \right)^2 = (f - F)^T (f - F),
\]

where

\[
f = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} F(x_1; c) \\ \vdots \\ F(x_n; c) \end{pmatrix}.
\]

The vector \( e = f - F \) measures the error at the \( n \) data points.

At a minimum of \( E(c) \), the partial derivative of \( E \) with respect to \( c \) vanishes, that is,

\[
\frac{\partial}{\partial c} E(c) = 0.
\]

Let us write out these partial derivatives\(^2\):

\[
\frac{\partial}{\partial c} E(c) = -2(f - F)^T \frac{\partial F}{\partial c} = -2(f - F)^T \left( \frac{\partial F}{\partial c_1}, \ldots, \frac{\partial F}{\partial c_k} \right) = -2(f - F)^T (\phi_1, \ldots, \phi_k), \quad \text{by (2)},
\]

where, for \( 1 \leq j \leq k \),

\[
\phi_j = \begin{pmatrix} \phi_j(x_1) \\ \vdots \\ \phi_j(x_n) \end{pmatrix}.
\]

\(^2\)For vector calculus, you are referred to Appendix A in the lecture notes titled “Nonlinear Optimization.”
The vectors $\phi_i$s respectively encode the values of the basis functions $\phi_j$s at the $n$ data points. To summarize, $E(c)$ is a minimum only if
\[(f - F)^T \Phi = 0,\] (3)
where
\[\Phi = (\phi_1, \ldots, \phi_k).\]
Equations (3) are referred to as normal equations.

There are two ways to solve the normal equations (3). In the first approach, we rewrite $F$ as
\[F = c_1 \phi_1 + \cdots + c_k \phi_k = \Phi c,\] (4)
and substitute the expression into (3) to arrive at a system in terms of the variables $c$:
\[\Phi^T \Phi c = \Phi^T f.\] (5)
The $k \times k$ coefficient matrix $\Phi^T \Phi$ is non-singular as long as the matrix $\Phi$ has the full rank $k$, namely, the vectors $\phi_1, \ldots, \phi_k$ are linearly independent. This is almost always true since the basis functions $\phi_1, \ldots, \phi_k$ are independent of each other, since there is no reason for their values at the points $x_1, \ldots, x_n$ to be related. So we solve (5) to obtain
\[c = (\Phi^T \Phi)^{-1} \Phi^T f.\]
However, the matrix $\Phi^T \Phi$ is often very ill-conditioned, leading to unreliable results.

The second approach observes that the normal equations essentially state that at the minimizing $c$, the error vector $e = f - F$ is perpendicular to the column space of $\Phi$, which is the hyperplane spanned by $\phi_1, \ldots, \phi_k$. (All these vectors reside in $\mathbb{R}^n$, the space of possible data values at the points $x_1, \ldots, x_n$.) The obtained solution $F$ is thus the orthogonal projection of $f$ (the data vector) onto the column space of $\Phi$ (the space of permissible function vectors). This is illustrated in the following figure (drawn in $\mathbb{R}^n$). Practically, we hope that the effect of this orthogonal projection is to remove noise and reconstruct the function $f(x)$. This is classic least squares.
The idea is the following. We often cannot get the exact solution \( c \) to the overconstrained system of equations

\[ F(x_i; c) = f_i, \quad 1 \leq i \leq n, \]

written compactly as

\[ \Phi c = f, \tag{6} \]

which intends to match the predicted values with the measured values. So we get as close as possible instead. The SVD solution to (6) projects \( f \) orthogonally onto the space spanned by \( \phi_1, \ldots, \phi_k \), so it satisfies the normal equations (3). The solution vector \( c \) is then our best approximation.

**Example 2.** Let us return to Example 1. The basis functions are

\[ \phi_1(x) = 1, \quad \phi_2(x) = x, \quad \text{and} \quad \phi_3(x) = x^2. \]

We seek coefficients \( c_1, c_2, c_3 \) such that \( F(x) = c_1 + c_2 x + c_3 x^2 \) is the best least squares approximation to the data \((x_i, f_i)\). We have

\[
\begin{pmatrix}
4.1 \\
2.3 \\
1.05 \\
0.2 \\
0.05 \\
0.26 \\
0.9
\end{pmatrix}, \quad \begin{pmatrix}
1 \\
1 \\
1 \\
1 \\
1 \\
1 \\
1
\end{pmatrix}, \quad \begin{pmatrix}
-1.0 \\
-0.5 \\
0 \\
0.5 \\
1.0 \\
1.5 \\
2.0
\end{pmatrix}, \quad \begin{pmatrix}
1.0 \\
0.25 \\
0 \\
0.25 \\
1.0 \\
2.25 \\
4.0
\end{pmatrix}.
\]

If we use (5), we would solve

\[
\begin{pmatrix}
7 & 3.5 & 8.75 \\
3.5 & 8.75 & 11.375 \\
8.75 & 11.375 & 23.1875
\end{pmatrix} \begin{pmatrix}
c_1 \\
c_2 \\
c_3
\end{pmatrix} = \begin{pmatrix}
8.86 \\
-2.91 \\
8.96
\end{pmatrix}.
\]

If we use the SVD method, we would solve

\[
\begin{pmatrix}
1 & -1.0 & 1.0 \\
1 & -0.5 & 0.25 \\
1 & 0 & 0 \\
1 & 0.5 & 0.25 \\
1 & 1.0 & 1.0 \\
1 & 1.5 & 2.25 \\
1 & 2.0 & 4.0
\end{pmatrix} \begin{pmatrix}
c_1 \\
c_2 \\
c_3
\end{pmatrix} = \begin{pmatrix}
4.1 \\
2.3 \\
1.05 \\
0.2 \\
0.05 \\
0.26 \\
0.9
\end{pmatrix}.
\]

By either means we find that \((c_1, c_2, c_3) = (1.044, -2.044, 0.995)\). Therefore our estimated polynomial is \( p(x) = 1.044 - 2.044x + 0.995x^2 \), which is not too far from the true function \( f(x) = 1 - 2x + x^2 \).

On the next page, the diagram\(^3\) (a) compares the graphs of \( p(x) \) and \( f(x) \). Notice that there is a small persistent error between \( p(x) \) and \( f(x) \). However, the shapes of \( p \) and \( f \) agree well. This is because we restricted ourselves to a quadratic subspace of the space of functions.

In contrast, the diagram (b) shows the interpolating polynomial \( p_0(x) \) that matches the \( f_i \) values at the \( x_i, i = 1, \ldots, 7 \). While we get agreement between \( p_0(x) \) and \( f(x) \) more often, the maximum error is substantially larger than it was for \( p(x) \) and \( f(x) \). Furthermore, being of higher order and by matching the spurious data \( f_i \) exactly, \( p_0(x) \) wiggles around a lot more than \( p(x) \). Thus the general “shape” is not as good.

\(^3\)Both diagrams are courtesy of Mike Erdmann.
3 Curve and Surface Fitting

In computer vision, robotics, and geometric modeling, it is often important to derive some implicit representation of an object from raw shape data obtained using an imaging or touch sensing device. Such an implicit representation often takes the form of the zero set of a polynomial of even degree.\footnote{It is known that the zero sets of polynomials of odd degrees always define unbounded curves and surfaces.}

In practice, quartics or superquartics [1] have often been the choices for describing surfaces of moderate geometric complexities.

An implicit representation in polynomial form has several advantages. Firstly, it gives us a succinct shape description with few parameters so storage of a large amount of raw data is no longer necessary. Secondly, using the polynomial we can compute some algebraic invariants (which are expressions in terms of the polynomial’s coefficients) to recognize the object from a set of known models. Thirdly, we would be able to differentiate the shape to obtain local geometry indices such as curvature and torsion. Fourthly, evaluating the polynomial at a point would easily tell us whether the point is inside, outside, or on the surface. Finally, a polynomial fit handles noisy data effectively in the sense that most shape irregularities due to noise are filtered out.

In surface fitting, a family of quartic polynomials with stably bounded zero sets is often considered. Choice of the family takes into account factors such as the effectiveness of representation and the computational efficiency of fitting. Polynomial coefficients are the parameters to be determined by, say, a least-squares method. For example, we may minimize the sum of squares of the distances from individual data points to the polynomial surface.

Suppose we need to fit a curve or surface over \( n \) data points \( \mathbf{p}_1, \ldots, \mathbf{p}_n \) in \( \mathbb{R}^2 \) or \( \mathbb{R}^3 \). The family of polynomial curves or surfaces that we consider assumes the form

\[
f(x; a_0, \ldots, a_k) = 0, \tag{7}
\]

where \( x = (x, y) \) or \( (x, y, z) \), and \( a_0, \ldots, a_k \) are the coefficients. Denote by \( d_i \) the distance from \( \mathbf{p}_i \) to the surface (7), for \( i = 1, \ldots, n \). Then the fitting problem can be formulated in a least-squares
fashion as

\[
\min_{a_0, \ldots, a_k} \sum_{i=0}^{k} d_i^2.
\]  

(8)

The issue, of course, is that we cannot determine \(d_i\), \(1 \leq i \leq n\), until \(a_0, \ldots, a_k\) are known, which conversely depends on the knowledge of \(d_i\) according to (8). To get out of this chicken-and-egg situation, let us approximate \(d_i\) as follows. Denote by \(q_i\) the closest point to \(p_i\) on the surface to be determined. Obviously, \(f(q_i; a_0, \ldots, a_k) = 0\). The best fitting \(f\) will place \(q_i\) close enough to \(p_i\). So we can approximate the value of \(f\) at \(q_i\) using Taylor’s series at \(p_i\), discarding all terms of the second order and above:

\[
f(q_i; a_0, \ldots, a_k) \approx f(p_i; a_0, \ldots, a_k) + \nabla f(p_i; a_0, \ldots, a_k) \cdot (q_i - p_i).
\]

Since the left hand side of the above is zero, we have

\[
\nabla f(p_i; a_0, \ldots, a_k) \cdot (q_i - p_i) \approx -f(p_i; a_0, \ldots, a_k).
\]

Because \(q_i\) and \(p_i\) are very close to each other, the vector \(q_i - p_i\) and the gradient \(\nabla f(p_i; a_0, \ldots, a_k)\) are nearly parallel. The above equation yields an approximation:

\[
d_i = \|q_i - p_i\| \approx \frac{|f(p_i; a_0, \ldots, a_k)|}{\|\nabla f(p_i; a_0, \ldots, a_k)\|}.
\]

With a substitution of the above approximation into (8), we reformulate the fitting problem as

\[
\min_{a_0, \ldots, a_k} \frac{f^2(p_i; a_0, \ldots, a_k)}{\|\nabla f(p_i; a_0, \ldots, a_k)\|^2}.
\]

The reader is referred to [2, 5, 7] for more on surface fitting techniques and experiments.

4 Surface Patch Reconstruction

A robotic hand can reconstruct an unknown surface patch by touch [4]. The idea is to track along three concurrent\(^5\) curves on the surface, obtaining tactile data points \((x_k, y_k, z_k)\), \(1 \leq k \leq n\), in the meantime. Each curve is the intersection of the patch with a separate plane (called the sampling plane) within which the tracking motion is presently constrained. Denote by \(p\) the intersection point of the three curves. Through fitting a parabola to the data points along each curve, its

\(^5\)intersecting at one point
curvature at $p$ is estimated. The surface’s normal curvature at $p$ in the tangent direction of this curve is the product of the (estimated) curvature with the cosine of the angle between the sampling plane and the normal plane containing the tangent direction.

Three normal curvatures are thus obtained. From them and the relative orientations of the corresponding tangent vectors at $p$, we solve for the principal curvatures $\kappa_1$ and $\kappa_2$ and the Darboux frame at the point $p$. Under this frame, the surface patch locally takes the form

$$z(x, y) = \frac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2) + \sum_{3 \leq i+j \leq d} a_{ij} x^i y^j,$$

where the terms of degrees above two are added to describe a larger area of the surface. The coefficients of the polynomial, gathered into a vector $a$, are determined in a least-squares sense

$$\min_a f(a)$$

where

$$f(a) = \frac{1}{n} \sum_{k=1}^n (z(x_k, y_k) - z_k)^2.$$  

However, the resulting surface described by $z(x, y)$ may “fold” many times unrealistically, as indicated by a high total absolute Gaussian curvature:

$$g(a) = \int \int_D |K(x, y)| \cdot \sqrt{1 + z_x^2 + z_y^2} \, dx \, dy.$$  

To get rid of unnecessary “folds”, we minimize the total absolute Gaussian curvature subject to the constraint that the surface fit should “pass through” the sampled data points:

$$\min_a g(a) \quad \text{subject to } f(a) = 0.$$  

The technique of Lagrange multipliers is then applied.

The next figure displays a shell and two views (inside-out and outside-in) of the reconstructed patch\textsuperscript{6} overlaid onto its mesh model generated by NextEngine’s desktop 3-D scanner (accuracy 0.127mm). The data points from tracking are shown in red while the estimated Darboux frame in brown. The average distance of the mesh vertices in the overlay area to the patch is 0.2283mm, indicating a close match.

\textsuperscript{6}In the reconstruction, $d$ was set to be 4.
A Statistical Rationale for Least Squares

The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum.

— Karl Friedrich Gauss

The problem of data fitting can be formally phrased below:

Given observations \( \{(x_i, f_i)\} \) and a family of functions \( F(x) = \sum_{j=1}^{k} c_j \phi_j(x) \), what is the likelihood of a particular set of parameters \( c_1, \ldots, c_k \)?

Instead of trying to answer the above question directly, we seek to answer the following question:

Given \( c_1, \ldots, c_k \), what is the probability that \( \{(x_i, f_i)\} \) could have occurred?

Essentially, we identify the probability given the parameters as the likelihood of the parameters given the data.

Let us treat each \( f_i \) as an independent “random variable” with normal distribution around the “true” value \( F(x_i) \), that is, the “mean” of \( f_i \). This yields the probability

\[
e^{-\frac{(f_i - F(x_i))^2}{2\sigma^2}} df_i,
\]

where \( \sigma \) is the standard deviation of all \( f_i \). But the above probability is mathematically zero, so we replace \( df_i \) with \( \Delta f_i = \Delta f \):

\[
e^{-\frac{(f_i - F(x_i))^2}{2\sigma^2}} \Delta f.
\]

Therefore, the probability of the data set \( \{(x_i, f_i)\} \) is

\[p = \prod_{i=1}^{n} \left( e^{-\frac{(f_i - F(x_i))^2}{2\sigma^2}} \Delta f \right).\]

Maximizing \( p \) is equivalent to minimizing

\[- \ln p = \sum_{i=1}^{n} \frac{(f_i - F(x_i))^2}{2\sigma^2} - n \ln(\Delta f).\]

Since \( n, \sigma, \) and \( \Delta f \) are constants, we end up with the familiar least squares problem:

\[
\min \sum_{i=1}^{n} \left( f_i - F(x_i) \right)^2.
\]

References


