# Data Fitting

(Com S 477/577 Notes)

Yan-Bin Jia

Nov 5, 2020

## 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 $f_i$.

| $x_i$ | $f(x_i)$ | $f_i$ |
|---|---|---|
| $-1$ | 4 | 4.1 |
| $-\frac{1}{2}$ | 2.25 | 2.3 |
| 0 | 1 | 1.05 |
| $\frac{1}{2}$ | .25 | .20 |
| 1 | 0 | .05 |
| $\frac{3}{2}$ | .25 | .26 |
| 2 | 1 | .90 |

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]

$$1, \qquad x, \qquad x^2$$

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

$$f_i = a + bx_i + cx_i^2, \qquad i = 1, \ldots, 7, \tag{1}$$

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 $(\boldsymbol{x}_i, f_i)$ at a data point $\boldsymbol{x}_i \in \mathbb{R}^m$ for some $m > 0$. And we would like to reconstruct the function $f(\boldsymbol{x})$. What we have is some parametrized family of functions:

$$F(\boldsymbol{x}) = F(\boldsymbol{x}; c_1, \ldots, c_k). \tag{2}$$

We then choose the parameters $c_1, \ldots, c_k$ based on the observations $\{(\boldsymbol{x}_i, f_i)\}$ in such a way that $F(\boldsymbol{x})$ is "close to" $f(\boldsymbol{x})$. Often, for mathematical simplicity we write $F(\boldsymbol{x})$ as a linear combination of some $k$ basis functions: $F(\boldsymbol{x}) = c_1\phi_1(\boldsymbol{x}) + \cdots + c_k\phi_k(\boldsymbol{x})$. Here $k$ should be large enough so that the information about $f(\boldsymbol{x})$ 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 Linear Least Squares

How are the coefficients $\{c_i\}$ chosen? Ideally, we would like to minimize the difference between $f(\boldsymbol{x})$ and $F(\boldsymbol{x})$. Unfortunately, we do not know $f(\boldsymbol{x})$, so instead we simply minimize the difference between the two functions at the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. There are a variety of norms by which to measure the error:

$$\begin{aligned}
\infty\text{-norm:} \qquad \|f - F\|_\infty &= \max_{1 \le i \le n} \Big| f_i - F(\boldsymbol{x}_i) \Big|, \\
1\text{-norm:} \qquad \|f - F\|_1 &= \sum_{i=1}^{n} \Big| f_i - F(\boldsymbol{x}_i) \Big|, \\
p\text{-norm:} \qquad \|f - F\|_p &= \sqrt[p]{\sum_{i=1}^{n} \Big| f_i - F(\boldsymbol{x}_i) \Big|^p}.
\end{aligned}$$

[1] Other 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 norm notation. In fact, $F(\boldsymbol{x}_i) = F(\boldsymbol{x}_i; c_1, \ldots, c_k)$. 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 $F(\boldsymbol{x})$ has the simple linear form

$$F(\boldsymbol{x}) = c_1\phi_1(\boldsymbol{x}) + \cdots + c_k\phi_k(\boldsymbol{x}). \tag{3}$$

However, as we shall see, in the case of a 2-norm with $F(\boldsymbol{x})$ 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

$$\|f - F\|_2 = \sqrt{\sum_{i=1}^{n} \Big| f_i - F(\boldsymbol{x}_i; c_1, \ldots, c_k) \Big|^2}.$$

Equivalently, we are choosing

$$\boldsymbol{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix}$$

to minimize the function

$$\begin{aligned} \omega(\boldsymbol{c}) &= \sum_{i=1}^{n} \Big( f_i - F(\boldsymbol{x}_i; \boldsymbol{c}) \Big)^2 \\ &= (\boldsymbol{f} - \boldsymbol{\alpha})^T (\boldsymbol{f} - \boldsymbol{\alpha}), \end{aligned} \tag{4}$$

where

$$\boldsymbol{f} = \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{\alpha} = \begin{pmatrix} F(\boldsymbol{x}_1; \boldsymbol{c}) \\ \vdots \\ F(\boldsymbol{x}_n; \boldsymbol{c}) \end{pmatrix}.$$

The vector $\boldsymbol{f} - \boldsymbol{\alpha}$ measures the error at the $n$ data points.

At a minimum of $\omega(\boldsymbol{c})$, the partial derivative of $\omega$ with respect to $\boldsymbol{c}$ vanishes, that is,

$$\frac{\partial \omega}{\partial \boldsymbol{c}} = \boldsymbol{0}.$$

Let us write out these partial derivatives[2]:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{c}} \omega(\boldsymbol{c}) &= -2(\boldsymbol{f} - \boldsymbol{\alpha})^T \frac{\partial \boldsymbol{\alpha}}{\partial \boldsymbol{c}} \\ &= -2(\boldsymbol{f} - \boldsymbol{\alpha})^T \left( \frac{\partial \boldsymbol{\alpha}}{\partial c_1}, \ldots, \frac{\partial \boldsymbol{\alpha}}{\partial c_k} \right) \\ &= -2(\boldsymbol{f} - \boldsymbol{\alpha})^T (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k), \qquad \text{by (3)}, \end{aligned}$$

where, for $1 \le j \le k$,

$$\boldsymbol{\phi}_j = \frac{\partial \boldsymbol{\alpha}}{\partial c_j} = \begin{pmatrix} \phi_j(\boldsymbol{x}_1) \\ \vdots \\ \phi_j(\boldsymbol{x}_n) \end{pmatrix}.$$

[2]For vector calculus, you are referred to Appendix A in the lecture notes titled "Nonlinear Optimization".

For $1 \leq i \leq n$, the vector $\boldsymbol{\phi}_i$ gathers the values of the basis functions $\phi_j$s at the $n$ data points. To summarize, $\omega(\boldsymbol{c})$ is a minimum only if

$$(\boldsymbol{f} - \boldsymbol{\alpha})^T \Phi = \mathbf{0}, \tag{5}$$

where

$$\Phi = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k).$$

Equation (5) is referred to as the *normal equation*.

There are two ways to solve the normal equations (5). In the first approach, we rewrite $\boldsymbol{\alpha}$ as

$$\begin{aligned} \boldsymbol{\alpha} &= c_1\boldsymbol{\phi}_1 + \cdots + c_k\boldsymbol{\phi}_k \\ &= \Phi\boldsymbol{c}, \end{aligned} \tag{6}$$

and substitute the expression into (5) to arrive at a system in terms of the vector $\boldsymbol{c}$:

$$\Phi^T\Phi\boldsymbol{c} = \Phi^T\boldsymbol{f}. \tag{7}$$

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 $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k$ are linearly independent. This is almost always true since the basis functions $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k$ are independent of each other, since there is no reason for their values at the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ to be related. So we solve (7) to obtain

$$\boldsymbol{c} = \left(\Phi^T\Phi\right)^{-1}\Phi^T\boldsymbol{f}.$$

However, the matrix $\Phi^T\Phi$ is often very ill-conditioned, leading to unreliable results.

The second approach observes that the normal equation essentially states that at the minimizing $\boldsymbol{c}$, the error vector $\boldsymbol{e} = \boldsymbol{f} - \boldsymbol{\alpha}$ is perpendicular to the column space of $\Phi$, which is the hyperplane spanned by $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k$. (All these vectors reside in $\mathbb{R}^n$, the space of possible data values at the points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.) The obtained solution $\boldsymbol{\alpha}$ is thus the orthogonal projection of $\boldsymbol{f}$ (the data vector) onto the column space of $\Phi$ (the space of permissible function vectors). This is illustrated in Figure 1. 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 $\boldsymbol{c}$ to the overconstrained system of equations

$$F(\boldsymbol{x}_i; \boldsymbol{c}) = f_i, \qquad 1 \leq i \leq n,$$

written compactly as

$$\Phi\boldsymbol{c} = \boldsymbol{f}, \tag{8}$$

which intends to match the predicted values with the measured values. So we get as close as possible instead. The SVD solution to (8) projects $\boldsymbol{f}$ orthogonally onto the space spanned by $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k$, so it satisfies the normal equation (5). The solution vector $\boldsymbol{c}$ is then our best approximation.

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

$$\phi_1(x) = 1, \qquad \phi_2(x) = x, \qquad \text{and} \qquad \phi_3(x) = x^2.$$

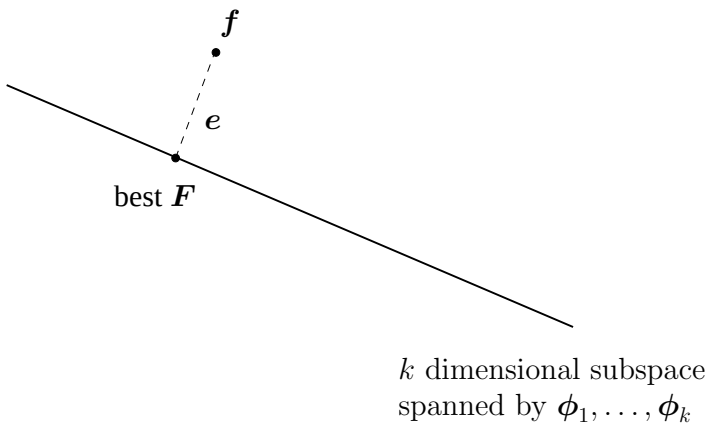



**Figure 1**: Linear least-squares determines the values of the parameters $c_1, \ldots, c_k$ by projecting the measurement vector $\boldsymbol{f}$ onto the subspace spanned by the vectors $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_k$ from evaluating the individual basis functions at all the data points.

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 $(\boldsymbol{x}_i, f_i)$. We have

$$\boldsymbol{f} = \begin{pmatrix} 4.1 \\ 2.3 \\ 1.05 \\ 0.2 \\ 0.05 \\ 0.26 \\ 0.9 \end{pmatrix}, \qquad \boldsymbol{\phi}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \qquad \boldsymbol{\phi}_2 = \begin{pmatrix} -1.0 \\ -0.5 \\ 0 \\ 0.5 \\ 1.0 \\ 1.5 \\ 2.0 \end{pmatrix}, \qquad \boldsymbol{\phi}_3 = \begin{pmatrix} 1.0 \\ 0.25 \\ 0 \\ 0.25 \\ 1.0 \\ 2.25 \\ 4.0 \end{pmatrix}.$$

If we use (7), we would solve

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

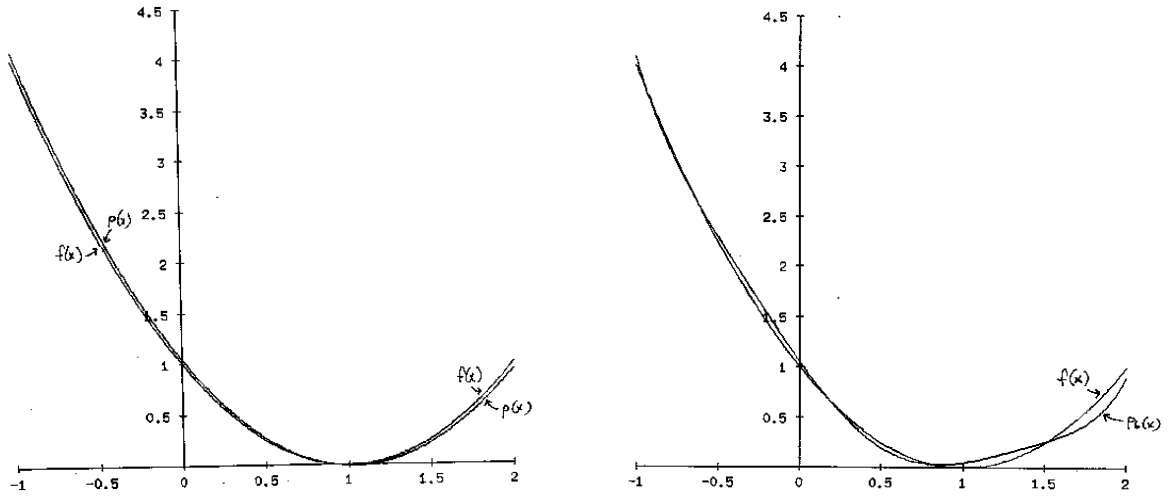
If we use the SVD method, we would solve

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

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 $f(x) = 1 - 2x + x^2$.

In Figure 2, the diagram[3] (a) compares the graphs of $F(x)$ and $f(x)$. Notice that there is a small persistent error between $F(x)$ and $f(x)$. However, the shapes of $F$ and $f$ agree well. This is because we restricted ourselves to a quadratic subspace of the space of functions.

[3]Both diagrams are courtesy of Mike Erdmann.




(a) (b)

**Figure 2**: (a) Fitting a parabola over noisy data points. (b) Interpolating at the same points with a polynomial of degree 6.

In contrast, the diagram (b) shows the interpolating polynomial $p_6(x)$ that matches the $f_i$ values at the $\boldsymbol{x}_i$, $i = 1, \ldots, 7$. While we get agreement between $p_6(x)$ and $f(x)$ more often, the maximum error is substantially larger than it was for $F(x)$ and $f(x)$. Furthermore, being of higher order and by matching the spurious data $f_i$ exactly, $p_6(x)$ wiggles around a lot more than $F(x)$. Thus the general "shape" is not as good.

# 3 Line Fitting

In computer vision, often we need to extract straight edges from an image in order to locate objects in the image. Such an edge is usually constructed via fitting to image points that are approximately collinear. More precisely, given $n$ points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ such that $\boldsymbol{p}_i = (x_i, y_i)^T$, $1 \leq i \leq n$, we would like to find a line that best fits these points. One approach is to represent the fitting line as $y = ax + b$ and minimize the sum of the squares of the *vertical offsets* of the points from the line. That is, we minimize the following function:

$$g(a, b) = \sum_{i=1}^{n} (ax_i + b - y_i)^2,$$

over $a$ and $b$ to determine the line. There are two drawbacks, however. First, a vertical line has the equation $x = x_0$ for some $x_0$, which cannot be put into the form of $y = ax + b$. Second, the vertical offset $|ax_i + b - y_i|$ does not well reflect the proximity of the point $\boldsymbol{p}_i$ to the fitting line. When the data distribution suggests a fitting line to be nearly vertical, even a point very close to the line can have a very large vertical offset.

Thus, a best fitting line should be regarded as one that minimizes the sum of the squared distances from the line to the $n$ points. Such a distance is referred to as the *perpendicular offset*.

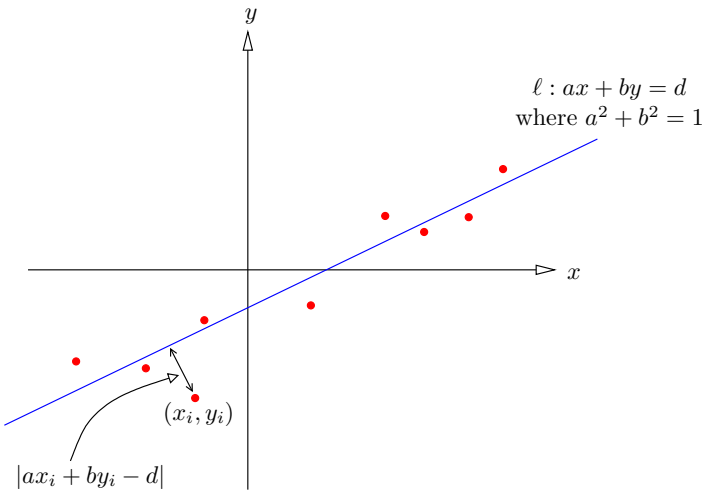



**Figure 3**: Fitting a line over points.

Below we offer a method for constructing such a line.[4] A line $\ell$ can be described by $ax + by = d$, where $a^2 + b^2 \neq 0$. Here, the vector $(a, b)$ is normal to the line. If we normalize it as a unit vector, then $ax + by - d$ is the (signed) distance from the point $\binom{x}{y}$ to the line. Thus, we should minimize

$$f(a, b, d) = \sum_{i=1}^{n} (ax_i + by_i - d)^2 \tag{9}$$

subject to the constraint that $a^2 + b^2 = 1$. Using a Lagrange multiplier $\mu$, we construct the Lagrangian

$$F(a, b, d, \mu) \quad = \quad f(a, b, d) + \mu(a^2 + b^2 - 1). \tag{10}$$

At a minimizing $(a, b, c, \mu)$, all the partial derivatives of $F$ must vanish. In particular, $\partial F/\partial d = 0$, which yields

$$2 \sum_{i=1}^{n} (ax_i + by_i - d) = 0.$$

This gives u

$$d = a\bar{x} + b\bar{x}, \tag{11}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \text{and} \qquad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

In other words, $(\bar{x}, \bar{y})^T$ is the centroid of the $n$ points.

[4]The description is revised from the two (more or less the same) solutions to this problem given in [2, pp. 294–295,pp. 666-667].

Substituting (11), we now rewrite the least-squares function (9) as

$$\begin{aligned} f(a,b,d) &= \sum_{i=1}^{n}\Big((ax_i-\bar{x})+b(y_i-\bar{y})\Big)^2 \\ &= (a,b)M^TM\begin{pmatrix}a\\b\end{pmatrix}, \end{aligned} \tag{12}$$

where

$$M=\begin{pmatrix} x_1-\bar{x} & y_1-\bar{y} \\ \vdots & \vdots \\ x_n-\bar{x} & y_n-\bar{y} \end{pmatrix}.$$

So the objective becomes to minimize (12) over all unit vectors. Clearly, the matrix $M^TM$ is symmetric and positive semi-definite. Let $\lambda_1$ and $\lambda_2$ be its eigenvalues, $0 \le \lambda_1 \le \lambda_2$, and $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ be the corresponding unit eigenvectors that are orthogonal to each other. Then, $(a,b)^T = \nu_1\hat{\boldsymbol{v}}_1 + \nu_2\hat{\boldsymbol{v}}_2$ for some $\nu_1$ and $\nu_2$, and we have

$$\begin{aligned} (a,b)M^TM(a,b)^T &= (\nu_1\hat{\boldsymbol{v}}_1+\nu_2\hat{\boldsymbol{v}}_2)^TM^TM(\nu_1\hat{\boldsymbol{v}}_1+\nu_2\hat{\boldsymbol{v}}_2) \\ &= (\nu_1\hat{\boldsymbol{v}}_1+\nu_2\hat{\boldsymbol{v}}_2)^T(\lambda_1\nu_1\hat{\boldsymbol{v}}_1+\lambda_2\nu_2\hat{\boldsymbol{v}}_2) \\ &= \nu_1^2\lambda_1+\nu_2^2\lambda_2. \end{aligned}$$

Apparently, the above expression is minimized when $\nu_2 = 0$, which implies $(a,b)^T = \hat{\boldsymbol{v}}_1$. In other words, the minimizing $(a,b)$ is a unit eigenvector corresponding to the smallest eigenvalue of $M^TM$.

# 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

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

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

$$\min \sum_{i=1}^{n} \Big(f_i - F(x_i)\Big)^2.$$

## References


[1] M. Erdmann. Lecture notes for *16-811 Mathematical Fundamentals for Robotics*. The Robotics Institute, Carnegie Mellon University, 1998.

[2] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Pearson Education Inc., 2nd edition, 2012.

[3] W. H. Press, *et al. Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2002.