Lecture 8: Putting least squares fitting to work

Recap

In Lecture 7, we went through the mechanics of least-squares fitting. Today, we want to put it to work by applying least-squares fitting to some specific examples. Recall that our basic framework is to solve a minimization problem if the form:

$$\mathbf{G}\mathbf{m} = \mathbf{d} \tag{1}$$

where $\mathbf{d}$ defines our data, $\mathbf{G}$ is a matrix defining the structure of the model that we fit to the data, and $\mathbf{m}$ is a vector containing the model parameters. We solve this as a linear equation.

Optimal Multi-Parameter Analysis

Let's start with a specific example: Optimal Multi-Parameter (OMP) analysis. Suppose you have oceanographic profile measurements from a certain location. You can imagine that water at this location represents a mixture of multiple end members, and you want to know how much each of the end members has contributed to the water at your location. OMP provides a framework for deciding on fractional contributions. There are a number of variants of this, but to keep things simple, let's suppose that we just want to know the mixture of water masses on a single isopycnal surface, following for example, the analysis of Frants et al (2013).

In this case, our data vector contains water properties on the isopycnal surface:

$$\mathbf{d} = \begin{bmatrix} 1 \\ T \\ S \\ O_2 \\ PO_4 \\ NO_3 \\ Si \\ PV \end{bmatrix}, \tag{2}$$

where each element of $\mathbf{d}$ is a value of one variable measured in the water: temperature, salinity, oxygen, phosphate, nitrate, silicon, and potential vorticity. The first element, 1, tells us that we're going to account for 100% of the water at this location. The matrix $\mathbf{G}$ represents the water masses at each of the end member locations:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 \\ T_1 & T_2 & T_3 \\ S_1 & S_2 & S_3 \\ O_{2_1} & O_{2_2} & O_{2_3} \\ PO_{4_1} & PO_{4_2} & PO_{4_3} \\ NO_{3_1} & NO_{3_2} & NO_{3_3} \\ Si_1 & Si_2 & Si_3 \\ PV_1 & PV_2 & PV_3 \end{bmatrix}, \tag{3}$$

and the model parameters,

$$\mathbf{m} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}, \tag{4}$$

indicate the fractional contributions of each of three water masses.

The first row of $\mathbf{G}$ impoises a requirement that $\alpha+\beta+\gamma=1$, that is that the water at our data location is comprised entirely of a mixture of the three water masses. Formally, we should also require that $alpha$, β, and γ are all positive and less than 1—that is that we cannot have negative contributions of water masses, nor can we have super-concentrations exceeding 100%.

We can spend a bit of time pondering the normalization of each of these values. If temperature is in millidegrees and swings over an enormous range, while salinity has a tiny range, then the big fluctuations in temperature will have an enormous impact on the OMP solution. To solve this, we will want to normalize by the variance of each value, so that each contributor has roughly equal impact on the solution, or so that the weight of each contributor is consistent with the uncertainties in the values.

Another concern is the absolute size of the values. If the data are noise free, the difference between temperature in °C vs temperature in Kelvin won't make much difference, but if there are large uncertainties, the big values in Kelvin could do strange things to our solution.

Once we lay out the problem, and appropriately weight the matrix $\mathbf{G}$, the the solution is exactly what we'd expect:

$$\mathbf{m} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d}. \tag{5}$$

Seasonal cycle of temperature

Last time we looked at idealized data. Now let's put this to work with the Ocean Station Papa temperature data, which have a strong seasonal cycle. To fit a seasonal cycle to the data, we set up a matrix $\mathbf{G}$ of the form:

$$\mathbf{G} = \begin{bmatrix} 1 & \cos\left(\frac{2\pi t_1}{365.25}\right) & \sin\left(\frac{2\pi t_1}{365.25}\right) \\ 1 & \cos\left(\frac{2\pi t_2}{365.25}\right) & \sin\left(\frac{2\pi t_2}{365.25}\right) \\ \vdots & \vdots & \vdots \\ 1 & \cos\left(\frac{2\pi t_N}{365.25}\right) & \sin\left(\frac{2\pi t_N}{365.25}\right) \end{bmatrix}. \tag{6}$$

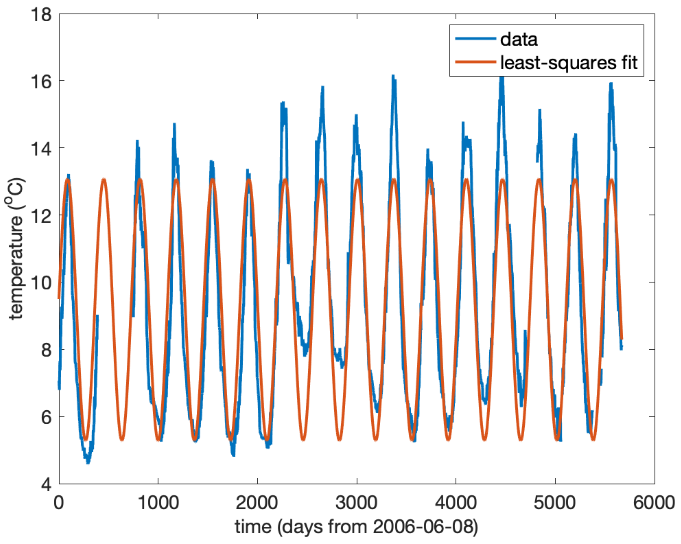
We can solve this:

```
file = 'data/sst50n145w_dy.cdf'
T=ncread(file,'T_25');
time=ncread(file,'time');
QT=ncread(file,'QT_5025');
xx=find(QT==0 | QT>=4);
T(xx)=NaN;
plot(time,squeeze(T),'LineWidth',2)

G=[ones(size(time)) cos(2*pi*time/365.25) sin(2*pi*time/365.25)];
G\squeeze(T)
```

```
xx=find(~isnan(T));
parameters=G(xx,:)\squeeze(T(xx))

% plot results
clf; plot(time,squeeze(T),'LineWidth',2)
hold on
plot(time,G*parameters,'LineWidth',2)
h=gca; set(h,'FontSize',14)
xlabel('time (days from 2006-06-08)','FontSize',14)
ylabel('temperature (^oC)','FontSize',14)
```



So to recap,

$$\mathbf{Gm} = \mathbf{d} \tag{7}$$

where $\mathbf{d}$ defines our data, $\mathbf{G}$ is a matrix defining the structure of the model that we fit to the data, and $\mathbf{m}$ is a vector containing the model parameters. We solve this as a linear equation.

Further examples

In class we considered four additional cases. You were asked to define the matrix $\mathbf{G}$ required to solve a least-squares problem for each of these cases:

1. *For a time series, fit (a) mean, (b) linear trend, (c,d) M2 tide (amplitude and phase).*
   The wording of this seemed a little confusing, but the goal is to fit all of these at once. For

this you can write a matrix:

$$\mathbf{G} = \begin{bmatrix} 1 & t_1 & \cos\left(\frac{2\pi t_1}{T_{M_2}}\right) & \sin\left(\frac{2\pi t_1}{T_{M_2}}\right) \\ 1 & t_2 & \cos\left(\frac{2\pi t_2}{T_{M_2}}\right) & \sin\left(\frac{2\pi t_2}{T_{M_2}}\right) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_N & \cos\left(\frac{2\pi t_N}{T_{M_2}}\right) & \sin\left(\frac{2\pi t_N}{T_{M_2}}\right) \end{bmatrix} \tag{8}$$

.

2. *For geographically distributed data, fit (a) mean, (b) tilted plane.*
In this case, you're asked to fit variables in x and y. Three points determine a plane, so we don't need too much information—just a fixed constant plus a slope in x and a slope in y:

$$\mathbf{G} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & y_N \end{bmatrix} \tag{9}$$

.

3. *Same as (2) but with a quadratic structure as well.*
In this case we need to add quadratic terms in x and y, and in analogy with the elliptical covariance that we considered a few days ago, we need to allow for an xy co-dependence:

$$\mathbf{G} = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & y_1^2 & x_1y_1 \\ 1 & x_2 & y_2 & x_2^2 & y_2^2 & x_2y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_N & y_N & x_N^2 & y_N^2 & x_Ny_N \end{bmatrix} \tag{10}$$

.

4. *For a time series, fit (a) mean, (b) linear trend, (c,d,e) Gaussian with unknown amplitude, center point, and width.*
This final case was motivated by a research question, but that doesn't make it easy. We're trying to minimize the misfit between data $\mathbf{d}$ and a function of the form:

$$\hat{d}_i = a + bx_i + c\exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right), \tag{11}$$

where the unknowns are a, b, c, μ, and σ, and two of the unknowns are embedded in the exponential. This is not tractable as a linear least-squares fit. We have a few options.

One possibility is to solve a minimization problem for a cost function ϵ:

$$\epsilon = \sum_{i=1}^{N}\left(d_i - \hat{d}_i\right)^2 \tag{12}$$

using a non-linear fitting procedure. Matlab and Python both have functions implemented for this. usually you need to provide a first guess, and work iteratively to sample a range possible values for model parameters $\mathbf{m}$.

A second option would be to decide that this model is too complicated. If we decided that we knew μ and σ, we could do a linear fit for a, b, and c.

Or if we could skip fitting a and b, then we could take the log of the model equation to have:

$$\begin{aligned} \log(\hat{d}_i) &= log(c) + \left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right) && (13)\\ &= log(c) - \frac{x_i^2}{2\sigma^2} + \frac{\mu x_i}{\sigma^2} - \frac{\mu^2}{2\sigma^2} && (14)\\ &= \alpha' + \beta' x_i^2 + \gamma' x_i + \delta' && (15) \end{aligned}$$

Of course we can't quite solve this, since α' and δ' are both scalars, so they need to be combined. And even if we're happy with this structure, it involves minimize a different cost function than the one we began with—not $\epsilon = (\mathbf{d}^T - \hat{\mathbf{d}}^T)(\mathbf{d} - \hat{\mathbf{d}})$, but instead $\epsilon = (\log \mathbf{d}^T - \log \hat{\mathbf{d}}^T)(\log \mathbf{d} - \log \hat{\mathbf{d}})$, which is really representing the ratio of the logs. Is this a problem? It depends on the physics of the problem that you're trying to solve.

Frants, M., S. T. Gille, C. D. Hewes, O. Holm-Hansen, M. Kahru, A. Lombrozo, C. I. Measures, B. G. Mitchell, H. Wang, and M. Zhou, 2013. Optimal multiparameter analysis of source water distributions in the southern Drake Passage, Deep-Sea Res. II, 90, 31-42.