A6523
Signal Modeling, Statistical Inference and Data Mining in Astrophysics
Spring 2011

Reading:
Chapter 10 = linear LSQ with Gaussian errors
Chapter 11 = Nonlinear fitting
Chapter 12 = Markov Chain Monte Carlo

Lectures 1-15 are on the web page

Lecture 22
Linear Least Squares (summary)
Non-linear Least Squares

Projects:
• Tuesday 19 April: Present 5 minute description of your project
Summary of Linear Least Squares

Unweighted Least squares: (equal, uncorrelated errors):

Cost function:

\[ Q(\theta) = \epsilon^\top \epsilon \equiv \epsilon^\top I \epsilon = \sum_j \epsilon_j^2. \]

Parameter vector that minimizes \( Q \):

\[ \hat{\theta} = (X^\top X)^{-1} X^\top y \]

Covariance matrix for the parameters:

\[ P \equiv \langle (\hat{\theta} - \theta)(\hat{\theta} - \theta)^\top \rangle = \sigma^2 \left( X^\top X \right)^{-1} \]

Cost function dependence on \( \theta = \hat{\theta} + \delta \theta \):

\[
Q(\theta) = (y - X\theta)^\top (y - X\theta) \\
= (y - X\hat{\theta})^\top (y - X\hat{\theta}) + \underbrace{\delta \theta^\top X^\top X \delta \theta}_{\text{quadratic form } \geq 0}.
\]

The cost function hypersurface is quadratic and has only one minimum.
Weighted Least squares:

Arbitrary covariance matrix $\mathbf{V}$ for $\epsilon$):

Cost function:

$$Q(\theta) = \epsilon^\dagger \mathbf{V}^{-1} \epsilon$$

Parameter vector that minimizes $Q$:

$$\hat{\theta} = (\mathbf{X}^\dagger \mathbf{V}^{-1} \mathbf{X})^{-1} \mathbf{X}^\dagger \mathbf{V}^{-1} \mathbf{y}$$

Covariance matrix for the parameters:

$$\mathbf{P} \equiv \langle (\hat{\theta} - \theta)(\hat{\theta} - \theta)^\dagger \rangle = \left(\mathbf{X}^\dagger \mathbf{V}^{-1} \mathbf{X}\right)^{-1}$$

Cost function dependence on $\theta = \hat{\theta} + \delta \theta$, $Q(\theta)$:

The cost function hypersurface generally has many local minima whereas we want the global minimum.
Nonlinear Least Squares

So far we have considered linear models of the form

\[ \mathbf{y} = \mathbf{X}\theta + \epsilon. \]

But often we want to fit models that are non-linear in the parameters \( \theta \), such as

\[ \mathbf{y} = \mathbf{f}(\mathbf{X}, \theta) + \epsilon, \]

where this vector equation has \( n \) elements and \( \theta \) is a \( k \)-vector.

We cannot solve for the best fit to the data in the same way as for the linear model, but the underlying principle is the same: minimize the sum of squares.
Thus, we minimize the quadratic form

\[ Q(\theta) = \epsilon^\dagger V^{-1} \epsilon. \]

The problem is to find the minimum \( Q \) in a k-space where \( Q \) is a nonmonotonic function of the parameters.

Recall that \( Q \) is of parabolic form for the linear LS problem, so finding any minimum of \( Q \) is the same as finding the minimum. With nonlinear functions, there may be an arbitrary number of local minima that can confuse algorithms for finding the nearest minimum of a function. This multiplicity of minima is the bane of the nonlinear LS problem.
The various strategies for minimizing $Q$ include:

1. A grid search in $\theta$ space: (a brute force approach). There is a dynamic-range problem of searching enough hyper-volume so that the \textit{global minimum} is found, but with sufficiently fine resolution that the global minimum is not missed. The total number of operations grows rapidly with $k$, the number of parameters. Let $\Theta_i$ be the total range searched for the parameter $\theta_i$ (one element of the vector $\theta$) and with a grid sample interval $\delta \theta_i$. The total number of grid points is

$$N = \prod_{i=1}^{k} \left( \frac{\Theta_i}{\delta \theta_i} \right).$$
2. A *ravine search*: Use the gradient of $Q$,

$$\nabla_\theta \equiv \frac{dQ}{d\theta},$$

to find the bottom of a particular valley in $\theta$-space. Choose a length $\delta\theta$ in the direction of the negative gradient, move to a new position, evaluate $Q$ and see if a minimum has been found. If not, iterate. This method clearly finds only the minimum that is nearest to the starting point of the search. This may not be the local minimum unless the starting point has been chosen wisely (or luckily).

A hybrid approach would combine ravine search with another, pilot search that has identified the rough location of the global minimum.
3. **Parabolic Extrapolation of Q:** Near a minimum, $Q$ may be approximated as a parabolic surface, so expression as such leads to a determination of the minimum. In vector form this is

$$Q = Q_{\text{min}} + (\nabla \theta Q)^\dagger \delta \theta + \frac{1}{2} \left[ \nabla \theta (\nabla \theta Q)^\dagger \delta \theta \right]^\dagger \delta \theta.$$

This can also be written in the form

$$Q = Q_{\text{min}} + \sum_k \left( \frac{\partial Q}{\partial \theta_k} \right)_{\text{min}} \delta \theta_k + \frac{1}{2} \sum_j \sum_k \left( \frac{\partial^2 Q}{\partial \theta_j \partial \theta_k} \right)_{\text{min}} \delta \theta_j \delta \theta_k.$$

Minimizing with respect to the increments $\delta \theta$, we obtain

$$\nabla \delta \theta Q = 0 = \nabla \theta Q + \nabla \theta \left( \nabla \theta Q^\dagger \delta \theta \right).$$

This is a k-vector equation that yields corrections $\delta \theta$ to initial guesses $\theta_0$ which yield

$$Q_0 = (y - \hat{y}(\theta_0))^\dagger V^{-1}(y - \hat{y}(\theta_0)).$$
The *Fisher information matrix* is related to the quadratic term above:

\[
F_{jk} = \frac{1}{2} \left( \frac{\partial^2 Q}{\partial \theta_j \partial \theta_k} \right)_{\text{min}}
\]

and is the inverse of the parameter covariance matrix (in the quadratic approximation).
4. **Linearization of the fitting function, $f(\theta)$**: Linearize $f(\theta)$ according to

$$f(\theta) \approx f_0(\theta_0) + \nabla_\theta f(\theta_0)^\dagger (\theta - \theta_0).$$

Then the model for the data becomes

$$y \approx f_0(\theta_0) + \nabla_\theta f(\theta_0)^\dagger (\theta - \theta_0) + \epsilon,$$

where $\theta_0$ is an initial guess for the parameters. Note that $f(\theta)$ is implicitly a function of some independent variable(s), as with linear LS.

Since the model is linear near the initial guess, one can solve for $\delta \theta \equiv \theta - \theta_0$ using the **linear** LS formalism. Specifically,

$$\delta \theta = \left( X^\dagger V^{-1} X \right)^{-1} X^\dagger V^{-1} y,$$

where $X$ is now the $n \times k$ matrix of values $\nabla_\theta f(\theta_0)$ for the k-dimensional gradient, evaluated at $n$ values of some independent variable (e.g. time, spatial coordinate, frequency, etc.).
Note that, like methods 2. & 3., linearization of the fitting function also will find only the minimum that is closest to the initial guess for the parameters.
Optimization Methods

We have seen a number of instances where we have wanted to maximize or minimize a function. For least-squares’ problems, the cases of interest are:

1. *Linear Models*: $Q$ is concave $\implies$ a single minimum found through a single iteration of the standard LS solution.
2. Non-linear Models: $Q$ is generally complicated with many local minima.

(a) Ravine searches, parabolic extrapolation, linearization of the fitting function are all iterative methods for finding the local minimum near a starting point. There is no guarantee that the global minimum will be found with these methods.

(b) Grid Search: can find the global minimum but at the great cost of evaluating functions at a large number of locations in $\theta$-space. Also, with too-coarse sampling, the global minimum can be missed with this method as well.

(c) Hill-climbing Method: Essentially the same as (a).
(d) **Downhill Simplex:** This method searches the parameter space, or domain, using a geometrical construct called a simplex, a non-coplanar object with $k + 1$ vertices in the k-space. There need not be any computations of derivatives, the method simply changes the shape of the simplex and moves it through the k-space according to values of $Q$ that are encountered at the vertices. It *can* get stuck in false minima, however, so multiple trials with different starting points should be used.
(e) **Simulated Annealing:** Allow trial values of parameters to jump around the domain (i.e. $\theta$-space) according to a temperature-like parameter and application of the Metropolis algorithm. This provides the opportunity for exploring the entire domain and not getting stuck in a local minimum. The temperature is lowered slowly as in annealing of metals, where the lattice finds a nice minimum-energy solution for itself. This method has a high probability of at least finding the neighborhood of the global minimum. Finding the exact minimum through the annealing process is slow. Hybridizing annealing with a method from A. can find the minimum more quickly.
(f) **Genetic Algorithms (GA’s):** Search the domain through genetic-like operations. Let the parameter vector be associated with *chromosomes* made up of *genes* that each represent a specific parameter. The chromosomes are subject to genetic manipulation between generations (iterations). The main genetic processes are:

i. *selection according to fitness* (defined in terms of a better value of the quantity being optimized, i.e. $Q$ in least-squares, likelihood function in ML);

ii. *recombination or crossover:* where selected pairs of chromosomes (parameter vectors) interchange genes (bits).

iii. *mutation:* where genes (bits) are randomly flipped according to some probability. This helps organisms from getting stuck in local minima.
GA’s can search the entire domain efficiently because successful substrings (bit sequences) in the chromosomes (“schema”) grow exponentially according to their fitness relative to the mean fitness. Thus, the genetic approach explores the domain more efficiently than a purely random search of the domain (e.g. Monte Carlo selection of parameter values) or a deterministic grid search because the genetic approach includes memory.