Statistics and Data Analysis in MATLAB
Kendrick Kay, kendrick.kay@wustl.edu
February 28, 2014

Lecture 4: Model fitting

1. The basics

- Suppose that we have a set of data and suppose that we have selected the type of model to apply to the data (see Lecture 3). Our task now is to fit the model to the data—that is, adjust the free parameters of the model such that the model describes the data as well as possible.
- Before proceeding, we must first establish a metric for the *goodness-of-fit* of the model. A common metric is *squared error*, that is, the sum of the squares of the differences between the data and the model fit:

$$\text{squared error} = \sum_{i=1}^{n} \left(d_i - m_i\right)^2$$

where n is the number of data points, d_i is the ith data point, and m_i is the model fit for the ith data point. We will see the motivation for squared error later in this lecture.
- Given the metric of squared error, our job is to determine the specific set of parameter values that minimize squared error. The solution to this problem depends on the type of model that we are trying to fit.

2. The case of linear models

- Model fitting in the case of linear (and linearized) models can be given a nice geometric interpretation. We can view the data as a single point in n-dimensional space and we can view the regressors as vectors in this space emanating from the origin. Potential model fits are given by points in the subspace spanned by the vectors. (The subspace consists of all points that can be expressed as a weighted sum of the regressors.) The model fit that minimizes squared error is the point that lies closest in a Euclidean sense to the data. (This is because the Euclidean distance between two points is a simple monotonic transformation—the square root—of the sum of the squares of the differences in the two points' coordinates.) The residuals of the model fit can be viewed as a vector that starts at the model fit and ends at the data.
- With these geometric insights in mind, we can now derive the solution to our fitting problem. Recall that linear models can be expressed as

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{n}$$

where $\mathbf{y}$ is a set of data points ($n \times 1$), $\mathbf{X}$ is a set of regressors ($n \times p$), $\mathbf{w}$ is a set of weights ($p \times 1$), and $\mathbf{n}$ is a set of residuals ($n \times 1$). Let $\hat{\mathbf{w}}^{\text{OLS}}$ denote the set of weights that provide the optimal model fit; these weights are called the *ordinary least-squares (OLS)* estimate. At the optimal model fit, the residuals must be orthogonal to each of the regressors. (If the residuals were correlated with a given regressor, then a better model fit could be obtained by moving in the direction of that regressor.) This orthogonality condition implies that the dot product between each regressor and the residuals must equal zero:

$$\mathbf{X}^{\text{T}}(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}^{\text{OLS}}) = \mathbf{0}$$

where $\mathbf{0}$ is a vector of zeros ($n \times 1$). Expanding and solving, we obtain:

$$\mathbf{X}^{\text{T}}\mathbf{y} - \mathbf{X}^{\text{T}}\mathbf{X}\hat{\mathbf{w}}^{\text{OLS}} = \mathbf{0}$$

and

$$\hat{\mathbf{w}}^{\text{OLS}} = (\mathbf{X}^{\text{T}}\mathbf{X})^{-1}\mathbf{X}^{\text{T}}\mathbf{y}$$

where $\mathbf{A}^{-1}$ indicates the matrix inverse of $\mathbf{A}$. Thus, we see that the set of weights that minimize squared error can be computed using an analytic expression involving simple matrix operations.
- Note that if there are more regressors than data points ($p > n$), the inversion of the correlation matrix $\mathbf{X}^{\text{T}}\mathbf{X}$ is ill-defined, and there is no OLS solution. Intuitively, the idea is that if there are more regressors than data points, then there are infinitely many solutions, all of which achieve zero error.

3. The case of nonlinear models

- Given that nonlinear models encompass a broad diversity of models, there is little hope of writing down a single expression that will solve the fitting problem for an arbitrary nonlinear model.
- To fit nonlinear models, we cast the problem as a search problem in which we have a parameter space, a cost function, and our job is to search through the space to find the point that minimizes the cost function. Since we are using the cost function of squared error, we can think of our job as trying to find the minimum point on an *error surface*. If there is only one parameter, the error surface is a function defined on one dimension (i.e. a curvy line); if there are two parameters, the error surface is a function defined on two dimensions (i.e. a bumpy sheet); etc.
- To search through the parameter space, the usual approach is to use local, iterative optimization algorithms that start at some point in the space (the *initial seed*), look at the error surface in a small neighborhood around that point, move in some direction in an attempt to reduce the error, and then repeat this process until improvements are sufficiently small (e.g. until the improvement is less than some small number). There are a variety of optimization algorithms, and they basically vary with respect to how exactly they make use of first-order derivative information (gradients) and second-order derivative information (curvature). The Levenberg-Marquardt algorithm is a popular and effective algorithm and is implemented in the MATLAB Optimization Toolbox.
- As a simple example of an optimization method, let us consider how to perform gradient descent for a linear model (reusing the earlier example $\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{n}$). Assuming the error metric is squared error, then the derivative of the error surface with respect to the jth weight is

$$\frac{\partial}{\partial w_j}\text{error} = \frac{\partial}{\partial w_j}\left((\mathbf{y}-\mathbf{X}\mathbf{w})^{\text{T}}(\mathbf{y}-\mathbf{X}\mathbf{w})\right) = \frac{\partial}{\partial w_j}\left(\sum_i (y_i - \mathbf{X}_i\mathbf{w})^2\right) = \sum_i 2(y_i - \mathbf{X}_i\mathbf{w})(-X_{i,j})$$

where w_j is the jth element of $\mathbf{w}$, y_i is the ith element of $\mathbf{y}$, $\mathbf{X}_i$ is the ith row of $\mathbf{X}$, and $X_{i,j}$ is the (i,j)th element of $\mathbf{X}$. Collecting the derivatives for different weights into a vector, we obtain the gradient of the error surface ($p \times 1$):

$$\nabla\text{error} = -2\mathbf{X}^{\text{T}}(\mathbf{y}-\mathbf{X}\mathbf{w})$$

So, what this tell us is that given a set of weights $\mathbf{w}$, we know how to compute the amount by which the error will change if we were to tweak any of the weights (e.g., if we were to increment the first weight by 0.01, then the error surface will increase by approximately 0.01 times the first element in the gradient vector). This suggests a simple algorithm: first, set all the weights to some initial value (e.g. all zeros); then, compute the gradient and update the weights by subtracting some small fraction of the gradient; and repeat the weight-updating process until the error stops decreasing. We will see that this algorithm can be implemented in MATLAB quite easily.

- A potential problem with local, iterative optimization is *local minima*, that is, locations on the error surface that are the minimum within some local range but which are not the absolute minimum that can be achieved (which is known as the *global minimum*).
- For linear models, the error surface is shaped like a bowl and there are no local minima. The lack of local minima makes parameter-search easy—as long as an algorithm can adjust parameters to reduce the error, we will eventually get to the optimal solution. Geometrically, there is a nice intuition for why linear models have no local minima: assuming there are at least as many data points as regressors, there is exactly one point in the subspace spanned by the regressors that is closest to the data, and as parameter values deviate from this point, the distance from the data grows monotonically.
- For nonlinear models, the error surface may be "bumpy" with local minima. Because of local minima, the solution found by an algorithm may not be the best possible solution.
- The severity of the problem of local minima depends on the nature of the data, the nature of the model, and the specific optimization algorithm used, so it is difficult to make any general statements. Strategies for dealing with local minima include (1) starting with different initial seeds and selecting the best resulting model, (2) exhaustively sampling the parameter space, and (3) using alternative optimization techniques such as genetic algorithms.

4. The motivation for squared error

- Given a probability distribution, we can quantify the probability, or likelihood, of a set of data. Moreover, for different probability distributions, we can ask which probability distribution maximizes the likelihood of the data. This procedure is known as *maximum likelihood estimation* and provides a means for choosing from amongst different models. For example, given a set of data, out of all of the possible Gaussian distributions, the one that maximizes the likelihood of the data has a mean and standard deviation equal to the mean and standard deviation of the data.
- Let us apply maximum likelihood estimation to the case of regression models. Suppose that the true underlying probability distribution for each data point is a Gaussian whose mean is equal to the model prediction and whose standard deviation is some fixed value. In other words, suppose that the data are generated by the model plus *independent, identically distributed (i.i.d.)* Gaussian noise. Then, the likelihood of a given set of data can be written as follows:

$$\text{likelihood}(d \mid m) = \prod_{i=1}^{n} p(d_i) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(d_i - m_i)^2}{2\sigma^2}}$$

where d represents the data, m represents the model, n is the number of data points, d_i is the ith data point, σ is the standard deviation of the Gaussian noise, and m_i is the model prediction for the ith data point. The model estimate, m, that we want is the one that maximizes the likelihood of the data:

$$\arg\max_m \left(\text{likelihood}(d \mid m)\right)$$

Because the logarithm is a monotonic function, the desired model estimate is also given by

$$\arg\max_m \left(\text{log-likelihood}(d \mid m)\right)$$

which in turn is equivalent to

$$\arg\min_m \left(\text{negative-log-likelihood}(d \mid m)\right)$$

Now, let's substitute in the likelihood expression:

$$\arg\min_m \left(-\log \left(\prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(d_i - m_i)^2}{2\sigma^2}} \right) \right)$$

Simplifying, we obtain

$$\arg\min_m \left(\sum_{i=1}^{n} \left(-\log \left(\frac{1}{\sigma\sqrt{2\pi}} \right) + \frac{(d_i - m_i)^2}{2\sigma^2} \right) \right)$$

We can drop the first term since it has no dependence on *m*:

$$\arg\min_m \left(\sum_{i=1}^{n} \frac{(d_i - m_i)^2}{2\sigma^2} \right)$$

We can drop the denominator since it has no dependence on *m*:

$$\arg\min_m \left(\sum_{i=1}^{n} (d_i - m_i)^2 \right)$$

Finally, we can rewrite this more simply:

$$\arg\min_m (\text{squared error})$$

Thus, we see that to maximize the likelihood of the data, we should choose the model that minimizes squared error. This shows that there is good motivation to use squared error, namely, that assuming i.i.d. Gaussian noise, the maximum likelihood estimate of the parameters of a model is the set of parameters that minimizes squared error.

5. An alternative error metric: absolute error

- The assumption that the noise is Gaussian may be inaccurate for two reasons. One, the measurement noise may be non-Gaussian. Two, the model being applied may not be the correct model (e.g., fitting a linear model when the true effect is quadratic). This has the consequence that unmodeled effects may be subsumed in the noise and may cause the noise to be non-Gaussian. Thus, although squared error has a nice motivation, we might desire to use a different error metric.
- One alternative metric is the sum of the absolute values of the differences between the data and the model fit:

$$\text{absolute error} = \sum_{i=1}^{n} |d_i - m_i|$$

where *n* is the number of data points, d_i is the *i*th data point, and m_i is the model fit for the *i*th data point. This choice of error metric is useful as it reduces the impact of outliers (which can be roughly defined as unreasonably extreme data points). For a theoretical motivation, it can be shown that the parameter estimate that minimizes absolute error is the maximum likelihood estimate under the assumption of Laplacian noise. (The Laplace distribution is just like the Gaussian distribution except that an exponential is taken of the absolute difference from the mean instead of the squared difference from the mean.)
- The difference between squared error and absolute error can be framed in terms of the difference between the mean and the median: the mean of a set of data points is the number that minimizes squared error (that is, the sum of the squares of the differences between the number and each of the data points), whereas the median of a set of data points is the number that minimizes absolute error (that is, the sum of the absolute differences between the number and

each of the data points). Thus, we can view absolute error as an error metric that is potentially more robust than squared error. Note, however, there are some disadvantages of absolute error: first, there is no analytic solution for linear models, and second, error surfaces quantifying absolute error may be less well-behaved (e.g. more local minima) than error surfaces quantifying squared error.