Optimization and Bootstrapping Methods

Lecture 4

Kevin McAlister

University of Michigan

February 3, 2017

Optimization

Optimization

- Statistics frequently has problems of optimization:
  - Maximum Likelihood
  - Minimum Variance Estimators
  - Mode Finding
- We can sometimes solve these problems analytically.
- As methods get more complicated, we need to rely more and more on computational procedures.

Optimization

- R is a fantastic tool for solving these problems.
- Base R has a great function for potentially high dimensional optimization problems, `optim()`
- `optim()` is useful for unconstrained or box constrained optimization.
- If the problem has constraints, check out the R package `alabama`.

optim()

- Pull up the vignette for the optim() package.
- There are a lot of arguments!
- In general, only three of the arguments need to be explicitly set:
  1. Initial Values
  2. Function to Optimize
  3. Method

Functions for Optimization

- The bulk of `optim()` relies on the function for which you are trying to find optimal parameters.
- Recall analytical optimization, where we have a function $f(\theta)$ and θ is a k-dimensional vector.

$$\frac{\partial f(\theta)}{\partial \theta_1}(\theta_1^*) = \frac{\partial f(\theta)}{\partial \theta_2}(\theta_2^*) = ... = \frac{\partial f(\theta)}{\partial \theta_k}(\theta_k^*) = 0$$

- Analytical approaches require both knowing the functional form of the objective function and, also, that we are able to take derivatives.
- As k gets large, this can be a real pain.

Functions for Optimization

- Imagine a brute force approach to optimization where we are able to test all possible combination of values of theta. The combination
- Very time consuming. Often not viable.
- A more reasonable approach is to evaluate the function at a proposal, θ_p. Then, evaluate points in a sphere around the initial proposal. Move to the point within the sphere which shows the biggest decrease from θ_p. Repeat this process until you reach a point where no points in the sphere have a smaller evaluation than the original point. This is the minimum.
- This is what `optim()` does, more or less.

Functions for Optimization

- Suppose that we have a function which we are trying to minimize with two parameters:

$$\underset{x,y}{\text{argmin}}(x, y)$$

- In R, we need to write this function in following way:

```
fun.to.min <- function(theta,...){
    x <- theta[1]
    y <- theta[2]
    #Evaluate the function for x and y
    ...
    return(f(x,y))
}
```

- This function takes in a value for x and a value for y and returns the functional evaluation.
- If we want to maximize a function, we can just stick a negative in front of the functional evaluation.

Functions for Optimization

- Note that there is only one argument for all k values of θ!
- `optim()` (and really almost all functions in base `R`) perform their operations on the first argument of the function. The other arguments need to be specified in the functional call.
- This will make more sense when we work through an example.

Functions for Optimization

- Built into each function is an assumption that there exists a set of locally unique minima.
- This means that if the minimum value of our function above exists at x^*, y^*, then:

$$f(x^*, y^*) < f(x^* + a, y^* + b); -\epsilon < a, b < \epsilon$$

- This assumption implies that we can only effectively optimize functions which are continuous.

Initial Values

- `optim()` requires that we pass it k starting values for the minima search.
- This argument is simple. Pass a k-dimensional vector with starting values.
- These values must be within the support of the function.
- Starting values that are close to the true optima will let the function succeed with fewer iterations. Fewer iterations means less time.
- This doesn't matter too much if the problem is relatively small.

Methods

- The final argument of interest is the method used to find the minimum of the function.
- We'll (briefly) chat about 3:
  1. Nelder-Mead: Uses a pseudo-gradient algorithm to search for minima in $\mathbb{R}^k$. Finds local minima. Starting values will dictate where we end up if there are multiple optima.
  2. L-BFGS-B: Similar to Nelder-Mead. Finds local minima. Only searches within a "box" defined by the user. Starting values still matter unless we know that our box only includes one minimum.
  3. SANN (Simulated Annealing): Uses pseudo-gradients and annealing to find the global minimum. Can be **VERY** time consuming if k is large.

Example!

- Recall that we can write linear regression as a minimization problem:

$$\beta^* = \underset{\beta}{\text{argmin}} \|(Y - X\beta)\|$$

- OLS is really just finding the set of coefficients that minimizes the sum of squared residuals.
- We know that this problem has a unique solution. We don't need to do a global search.
- On Canvas, there is an `R` script called PS514L4E1.R
- Start by installing `mvtnorm`. Load it in your R session: `library(mvtnorm)`.
- The beginning of the script generates a matrix of X values and corresponding Y values given a known set of coefficients.
- The middle is a function that returns SSR for a set of coefficients
- The end uses `optim()` to find the set of coefficients that minimized SSE.
- Fill in the missing pieces and run the script. Are the coefficients returned by this function close to the actual values?

Bootstrapping

Back to Monte Carlo

- Recall that last week we spent time going over basic Monte Carlo.
- Random draws from known distributions result in draws from a potentially unknown joint distribution.
- Let's extend this notion to working with data.
- Assume that we have N observations of P covariates, X. We also have N observations of a response variable, Y.
- When modelling, we make sets of assumptions about the distribution of Y. Generally, we don't have to make strong assumptions about the distribution of X.
- What if our assumptions are wrong?
- What if we don't believe that the asymptotic assumptions about the distributions of our estimators hold?

Motivating Example

- I have a data set of state health care expenditures for all states in the U.S.
- N = 50
- If I want to do inference on the mean, I assume that N is large and appeal to CLT s.t.:

$$\mu \sim N\left(\bar{X}, \frac{\bar{s}}{\sqrt{N}}\right)$$

- Is 50 large enough?
- CLT has a balancing condition about the variance of a random variable. What if California, NY, Texas, etc. cause our empirical PDF to have a fat tail?
- Worth a look.

Bootstrapping

- Last week, we took draws from a known distribution.
- Frequentist statistics rely on the notion that repeated sampling is possible.
- If I only have one sample, how can I best approximate taking multiple samples from the same population?
- Assume that my sample is good and representative of the true generating function for the population.
- Randomly draw N samples from my current sample. In other words, sample from the observed sample *with replacement*.

A Short Example

- Let's look at PS514L4E2.R
- What is this code doing?
- Let's pay special attention to `sample(x,length(x),replace=T)`
  - `sample()` takes a vector and samples *n* elements from that vector.
  - The first argument is always the vector to be sampled.
  - The second argument is the number of elements to sample.
  - The third argument is `replace = T`, which says sample *with* replacement.
  - There is a fourth argument, `prob = c()`, where we can assign probabilities of selecting to each element in the vector. The default is equal probability.

A Short Example

- Let's change the sample size in the initial generation of our variable to a much smaller number.
- `xx <- rchisq(47,4500)`
- Run the same code again. Look at the overlayed density plots. What do you see?
- Are the empirical means and standard errors close to one another?
- Because N is small and the variance of our generating distribution is relatively large, CLT is not setting in at $N = 50$.
- One situation where bootstrapping is advantageous is when we have small sample sizes.

When Bootstrapping Is Important

- Bootstrapping can be used to computationally model error structures of estimators.
- If asymptotic theory is sufficient, then bootstrapping is not necessary.
- What if the asymptotics of our estimators of interest are not known?

Quantile Regression

- Recall that OLS is nothing more than a minimization problem:

$$\beta^* = \underset{\beta}{\text{argmin}} \| Y - X\beta \|$$

- Minimizing the sum of squared errors.
- Modelling the conditional *mean* of Y given X.
- OLS is not robust to high influence points. Outliers can wreck the interpretation of OLS estimators.
- One way around this is to regress on a robust estimator of an effect.
- Quantiles are incredibly robust.
- Model the conditional quantile of Y given X.
  - A one unit change in X results in a β unit change in the p^{th} percentile of Y.

Quantile Regression

- We can run a quantile regression by solving the minimization problem:

$$\beta^* = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} \begin{cases} q(y_i - x_i\beta), & \text{if } y_i - x_i\beta \geq 0 \\ (q-1)(y_i - x_i\beta), & \text{if } y_i - x_i\beta < 0 \end{cases}$$

  where q is the conditional quantile we are trying to model.
- To model on the median, set $q = .5$.

Quantile Regression

- Using this minimizer, we can write a function in R to calculate the optimal β for our system.
- What are the errors on β in this case?
- Utilize a bootstrapping scheme!

Quantile Regression

- Open PS514L4E3.R
- What's going on?
- Look at the function for quantile regression. It is very similar to how we previously solved Linear Regression with `optim()`. The only difference is that our loss function has changed.
- Examine the plot comparing median regression, linear regression with the influential points, and linear regression without the influential points. How does median regression compare?
- Let's get the standard errors on the coefficients in the median regression.
- Run the bootstrapping scheme at the end of this script. What are we doing here?
- Note that the bootstrapping follows the same formula as before:
  1. Take a sample of size N from the original dataset with replacement.
  2. Calculate the median regression solution for those data points.
  3. Repeat this process a large number of times.