Advanced Microeconometrics

Exercise Set 6: Numerical Optimization

Andreas P. A. Pedersen, Week 44 2025

UNIVERSITY OF COPENHAGEN

Today's Plan

- Motivation
- Numerical optimizers
- Newton Raphson: a gradient optimizer
- Your time to shine!

Estimators as solutions to optimization problems

- Most estimation problems involve or can be reformulated as minimising some objective function
- Common objective functions include the sum of squared residuals (OLS, NLS), the sum of absolute deviations (LAD) and log likelihoods
- For instance, the Non-linear Least Squares estimator is the solution to the minimisation problem:

$$\hat{\theta} = \underset{\theta \in \Theta}{argmin} \sum_{i=1}^{N} (y_i - m(\mathbf{x}_i \theta))^2 \qquad (1)$$

  where m(.) can be some nonlinear function of $\mathbf{x}_i \theta$
- Sometimes analytical solutions to these optimisation problems are not readily available. Then numerical optimisers come in handy

Numerical Optimizers

- Generally speaking, there are two types of numerical optimisers: gradient based and non gradient based
- **Gradient based**: faster but require objective function to be smooth
- **Non gradient based**: in some sense more robust (can handle less smooth objective functions), but often slower
- Sometimes a combination of the two works well: start with a non-gradient based when far from the optimum, and switch to a gradient based once close to the optimum
- You can help your optimiser the more you do analytically beforehand, e,g, by ”feeding it” analytical Jacobians, Hessians

Gradient Based Optimisers

- The Newton-Raphson (N-R) Algorithm is gradient based
- N-R approximates the objective f(x) with a 2nd order Taylor. **Why is that a good idea?** The scalar case:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + 1/2f''(x_0)(x - x_0)^2 \tag{2}$$

- N-R finds the minimum of this Taylor expansion in each iteration. Differentating (2) wrt x and setting equal to zero:

$$0 = f'(x_0) + 2/2f''(x_0)(x - x_0) \Rightarrow x = x_0 - f'(x_0)/f''(x_0) \tag{3}$$

- At each iteration i, N-R yields x_{i+1} by updating x_i using $f'(x_i)/f''(x_i)$. **What is the role of the slope of f(x)? What about the curvature?**

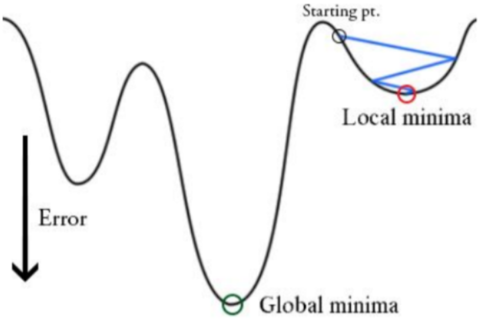
Matrix Notation

- Matrix version of the Newton-Raphson algorithm:

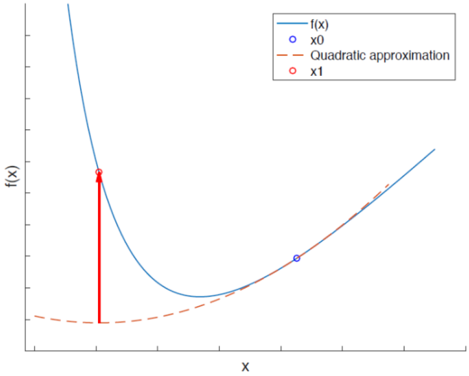
$$\theta_{i+1} = \theta_i - \mathbf{g}(\theta_\mathbf{i})\mathbf{H}(\theta_\mathbf{i})^{-\mathbf{1}} \tag{4}$$

where **H** and **g** are the Hessian and Gradient of the objective function, respectively.

Issues with NR I: Can get stuck in local extrema



Issues with NR II: Overshooting



Note on lambda functions

- You'll be using lambda to define functions. Previously, you've used def...return when making your functions
- lambda functions differ from normal function in a number of ways:
  - are (slightly) faster way to write short functions
  - are anonymous which means that if/when you get an error message, it will refer to the function as lambda, not the function name
  - can not contain statement e.g. asset, raise and so on
  - can not type hints. When defining a function as def f(x:np.ndarray, y : np.ndarray)–>np.ndarray
    we've specified that our function takes numpy arrays as inputs and returns a numpy array. This can be helpful for reading code though you don't get and error message if you input the wrong type. lambda functions don't have this functionality

Your time to shine!

- Solve the problem set
- The estimation.py file gives you a function for computing the forward difference of a function i.e.

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{5}$$

  This can be used to approximate first and second derivatives if you don't want to (or can't) find these analytically