Computational Methods in Economics

Numerical Optimization

February 13, 2020

Problem and Solution

Definition

$$\min_{x} f(x) \tag{1}$$

- $x \in \mathbb{R}^n$
- f is a smooth function.

Existence: Weierstrass theorem

A point or a vector x^* is a global minimizer if $f(x^*) \leq f(x) \, \forall \, x$.

Maximization Vs Minimization

Let $-f$ denote the function whose value at any value at any x is $-f(x)$. Then,

1. x is the maximum of f if and only if x is a minimum of $-f$
2. z is a minimum of f if and only if z is a maximum of $-f$

Necessary conditions

If $x^\star$ is optimal

- 1st order necessary condition: the gradient $f'(x^\star)$ is zero.
- 2nd order condition: the hessian $f''(x^\star)$ is positive and semi-definite.

Sufficient condition

If $x^\star$ is such that $f'(x^\star) = 0$, and $f''(x^\star)$ is positive definite, then $x^\star$ is a local minimum ($f(x) \geq f(x^\star)$)

Likelihood setup

The likelihood function is defined by:

$$\mathcal{L}_n(\theta) = \prod_{i=1}^{n} f(X_i; \theta) \tag{2}$$

The necessary conditions for optimization yield the regularity conditions

Feasible example: Poisson distribution

$$y_i \sim f(\lambda, y_i) = \frac{\exp(-\lambda)\lambda^{y_i}}{y_i!} \tag{3}$$

The likelihood is given by:

$$\mathcal{L}(y;\lambda) = \prod_{i=1}^{N} \frac{\exp(-\lambda)\lambda^{y_i}}{y_i!} = \frac{\exp(-N\lambda)\lambda^{\sum_i^N y_i}}{\prod_{i=1}^{N} y_i!} \tag{4}$$

$$\log \mathcal{L}(y;\lambda) = -N\lambda + \sum_i^N y_i \log(\lambda) - \sum_i^N \log(y_i!) \tag{5}$$

$$\frac{\partial \log \mathcal{L}(y;\lambda)}{\partial \lambda} = 0 \implies \widehat{\lambda} = \frac{\sum_i^N y_i}{N} \tag{6}$$

Unfeasible example:

Any nonlinear model

Numerical optimization

- Local optimization: the best minimum/maximum in a vicinity - usually defined by a convergence criteria.
- Global optimization: Best of all local minimas/maximas.

Numerical optimization - Local Optimization

Overview

1. **Line Search**: Starting from an initial value, choose a direction and search along this direction to find a new iterate
2. **Trust region**: Use previous estimates of the objective function, to construct a **synthetic** or **model** function whose behavior near the current point is similar to the objective function, and search only over a region, *trust region*, with the underlying idea that the model function is a good approximate over the trust region.

Gradient/Hessian

- No close form solution usually.
- Numerical approximation

$$f'(x) = \frac{f(x+\epsilon) - f(x)}{\epsilon} \tag{7}$$

- More consistent approach

$$f'(x) = \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon} \tag{8}$$

Convergence

Let an optimization algorithm generate some sequence $\{x_k\}$. This algorithm is said to **converge globally** when $\{x_k\}$ converge to "what is wished" (the point that satisfies the optimality conditions) for any initial iterate x_1.

Class of problems

- Unconstrained problems
- Problems with equality constraints

Line Search

Idea:

$$x_{k+1} = x_k + \alpha_k d_k \tag{9}$$

where d_k is a direction to be evaluated, and α_k a scaling parameter.

The variants of numerical optimization

1. Steepest descent: $d_k = -\nabla f(x_k)$
2. Newton direction: $d_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$
3. Quasi-Newton direction: $d_k = -(B(x_k))^{-1} \nabla f(x_k)$
4. Derivative free.

Properties

- Robustness: Perform well for various problems and starting values.
- Efficiency:
- Accuracy: Identify a solution with precision, not sensitive to starting values

Recover the scaling parameter

- Solve the function $\phi(\alpha) = f(x_k + \alpha d_k)$

Quasi-Newton methods

How to approximate the hessian such that:

- Reduce the computation time (Use only gradient instead of hessian)
- Increase convergence rate

Conjugate Gradient- FR

- Given x0
- Evaluate $f_0 = f(x0)$, $\nabla f_0 = \nabla f(x_0)$
- Set $d_0 = -\nabla f_0$, $k = 0$
- **While** $\nabla f_k \neq 0$
    - Set $x_{k+1} = x_k + \alpha_k d_k$
    - Evaluate ∇f_{k+1}, then:
        - $\beta_{k+1}^{FR} = \dfrac{\nabla f'_{k+1} \nabla f_{k+1}}{\nabla f'_k \nabla f_k}$
        - $d_{k+1} = -\nabla f_{k+1} + \beta_{k+1}^{FR} d_k$
        - $k = k + 1$
- **end(while)**

BFGS

- Given x0
- Evaluate $f_0 = f(x0)$, $\nabla f_0 = \nabla f(x_0)$, $H_0 = I$
- Set $k = 0$
- **While** $||\nabla f_k|| > \epsilon$
    - Compute direction $d_k = -H_k \nabla f_k$
    - Set $x_{k+1} = x_k + \alpha_k d_k$
    - Evaluate ∇f_{k+1}, then:
        - set $s_k = x_{k+1} - x_k$, $y_k = \nabla f_{k+1} - \nabla f_k$ and $\rho_k = \dfrac{1}{y_k' s_k}$
        - Update $H_{k+1} = (I - \rho_k s_k y_k') H_k (I - \rho_k y_k s_k') + \rho_k s_k s_k'$
- **end(while)**

Derivative Free

- Nelder-Mead
- Simulated annealing
- BOBYQA, COBYLA, ...