Deterministic Optimization

Unconstrained Optimization: Derivative Based

Shabbir Ahmed
Anderson-Interface Chair and Professor
School of Industrial and Systems Engineering

Newton's Method

Newton's Method

Learning objectives:

- Identify the Newton's method and Quasi-Newton method

Unconstrained Optimization: Derivative Based

$$(P): \quad \min f(x) \quad \text{s.t. } x \in \mathbb{R}^n$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuous and twice differentiable.

- Lesson 1: Optimality Conditions
- Lesson 2: Gradient Descent
- Lesson 3: Newton's Method

Descent Methods

$$(P): \quad \min f(x) \;\; \text{s.t.} \;\; x \in \mathbb{R}^n.$$

Basic paradigm of descent methods:
- Choose an initial solution x^0.
- Choose a descent direction d^0.
- Choose a step size α_0.
- Update the solution $x^1 = x^0 + \alpha_0 d^0$.
- If some stopping criteria is met, STOP; else repeat with current solution.

Newton's Method

- Let x^k be the current iterate and consider a second order Taylor approximation of $f(x)$ at x^k, i.e. $g(x) = f(x^k) + \nabla f(x^k)^\top (x - x^k) + \frac{1}{2}(x - x^k)\nabla^2 f(x^k)(x - x^k)$.

- Choose the next iterate as the solution that minimizes the approximate function $g(x)$.

  Setting $\nabla g(x) = 0$, we get the linear system

$$\nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0.$$

Newton's Method

- If the Hessian is non-singular, a solution to the above system is well-defined, and we set

$$x^{k+1} = x^k - [\nabla^2 f(x_k)]^{-1} \nabla f(x^k).$$

- In this case the improving direction is $d^k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x^k)$, and step size $\alpha = 1$.

Example: Newton's Iteration

$$\min f(x) = (x_1+1)^4 + x_1 x_2 + (x_2+1)^4$$

- Let $x^0 = [0,1]^\top$, and $f(x^0) = 17.0$.
- We have

  $$\nabla f(x) = [4(x_1+1)^3 + x_2, \quad x_1 + 4(x_2+1)^3]^\top \quad \text{and}$$

  $$\nabla^2 f(x) = \begin{bmatrix} 12(x_1+1)^2 & 1 \\ 1 & 12(x_2+1)^2 \end{bmatrix}$$

Example: Newton's Iteration

- At x^0, $\nabla f(x^0) = [5, \quad 32]^\top$ and

$$\nabla^2 f(x^0) = \begin{bmatrix} 12 & 1 \\ 1 & 48 \end{bmatrix}$$

- Therefore, $x^1 = [-0.3617, \; 0.3409]^\top$ and $f(x^1) =$ 3.2755.

Behavior of Newton's Method

- If started close enough to a local minimum and the Hessian is positive definite, then the method has quadratic convergence.

  However, in general ...

- Not guaranteed to converge. The Newton direction may not be improving at all!

  In the example, if we start from $x^0 = [-1, 1]^\top$ with $f(x^0) = 15$, the next iterate is $x^1 = [-2, 18]^\top$, with $f(x^1) = 130,286$!!

Behavior of Newton's Method

- If the Hessian is singular (or close to singular) at some iteration, we cannot proceed.

- Computing gradient as well as the Hessian and its inverse is expensive.

Quasi-Newton Methods

- Blend of gradient descent and Newton's method.
- Avoids computation of the Hessian and its inverse.
- Update iterate using

$$x^{k+1} = x^k - \alpha_k H_k \nabla f(x^k),$$

  where α_k is determined by line-search and H_k is an approximation to $[\nabla^2 f(x^k)]^{-1}$.

Quasi-Newton Methods

- Required properties:
  - H_k should be symmetric and positive definite.
  - Since H_{k+1}^{-1} approximates the Hessian,

$$H_{k+1}(\nabla f(x^{k+1}) - \nabla f(x^k)) = x^{k+1} - x^k.$$

- Along with the iterate, the matrix H_k is updated in each iteration using some update formulas that preserve above properties.

Quasi-Newton Methods

- A widely used formula is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula:

  $$H_{k+1} = H_k - \frac{d_k g_k^\top H_k + H_k g_k d_k^\top}{d_k^\top g_k} + \left(1 + \frac{g_k^\top H_k g_k}{d_k^\top g_k}\right)\frac{d_k d_k^\top}{d_k^\top g_k},$$

  where $g_k = \nabla f(x^{k+1}) - \nabla f(x^k)$ and $d_k = x^{k+1} - x^k$.

- Nonlinear programming solvers in Matlab (`fminunc`) uses variants of quasi-Newton methods with BFGS updates.

Summary

- Newton's method uses second order information
- It converges fast if started close to an optimal solution, otherwise it may not converge
- Quasi-Newton methods uses approximate Hessian estimates