18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

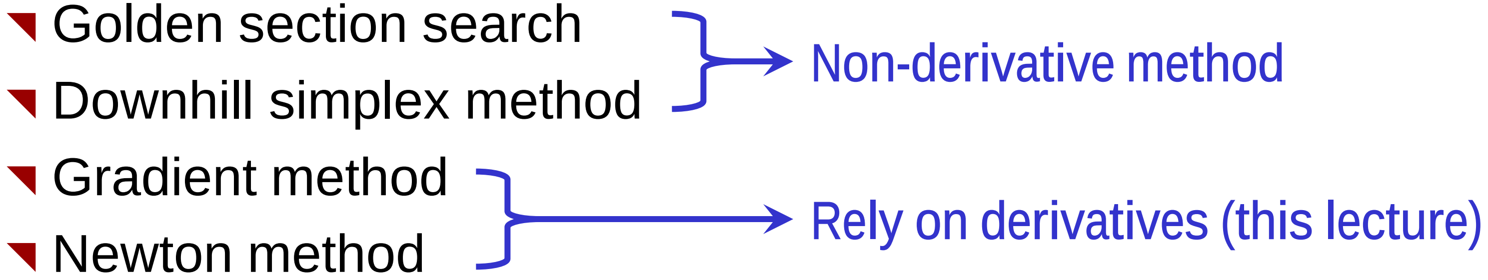
- Unconstrained Optimization
  - Gradient method
  - Newton method

Unconstrained Optimization

- Linear regression with regularization

$$A\alpha = B \quad \Rightarrow \quad \min_{\alpha} \quad \|A\alpha - B\|_2^2 + \lambda \cdot \|\alpha\|_1$$

- Unconstrained optimization: minimizing a cost function without any constraint



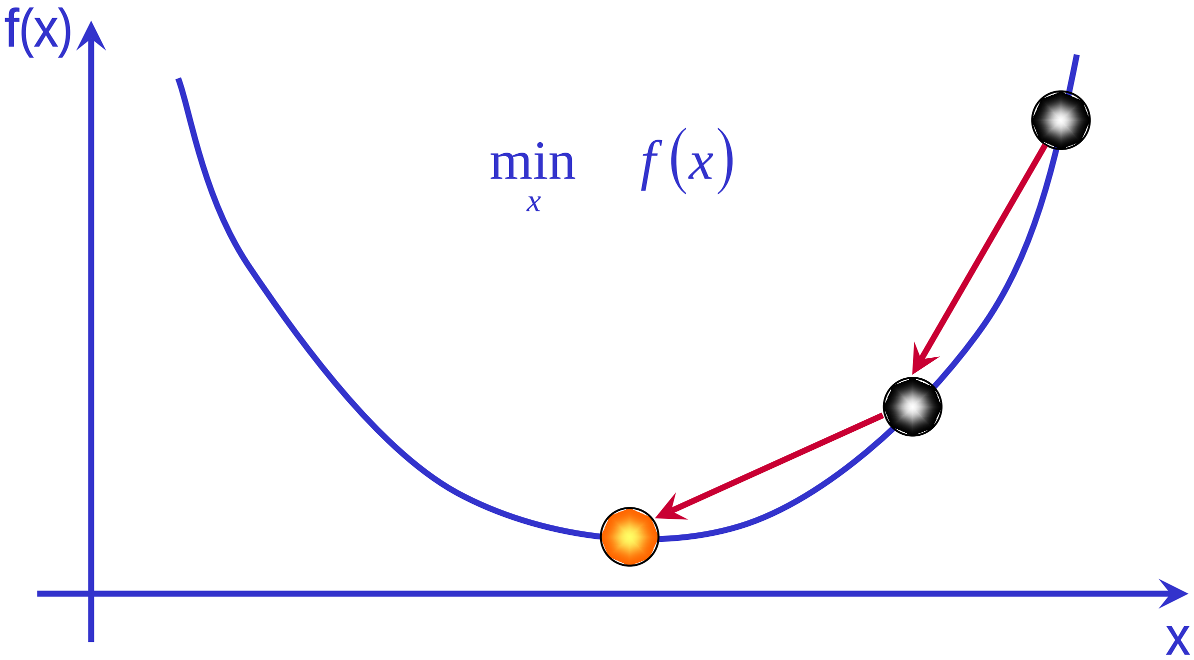
  - Golden section search
  - Downhill simplex method
  - Gradient method
  - Newton method

Non-derivative method

Rely on derivatives (this lecture)

Gradient Method

- If a cost function is smooth, its derivative information can be used to search optimal solution



$$\min_{x} \quad f(x)$$

Gradient Method

- For illustration purpose, we start from a one-dimensional case

$$\min_{x} \quad f(x)$$

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\underbrace{\lambda^{(k)}}_{\text{Step size}} \cdot \underbrace{\left.\frac{df}{dx}\right|_{x^{(k)}}}_{\text{Derivative}} \qquad \left(\lambda^{(k)} > 0\right)$$

$$\boxed{x^{(k+1)} = x^{(k)} - \lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \qquad \left(\lambda^{(k)} > 0\right)}$$

Gradient Method

- One-dimensional case (continued)

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \quad \& \quad f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \Delta x^{(k)}$$

$$\Downarrow$$

$$f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \left[-\lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}\right]$$

$$\Downarrow$$

$$f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] - \lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}^{2}$$

$\lambda^{(k)} > 0$

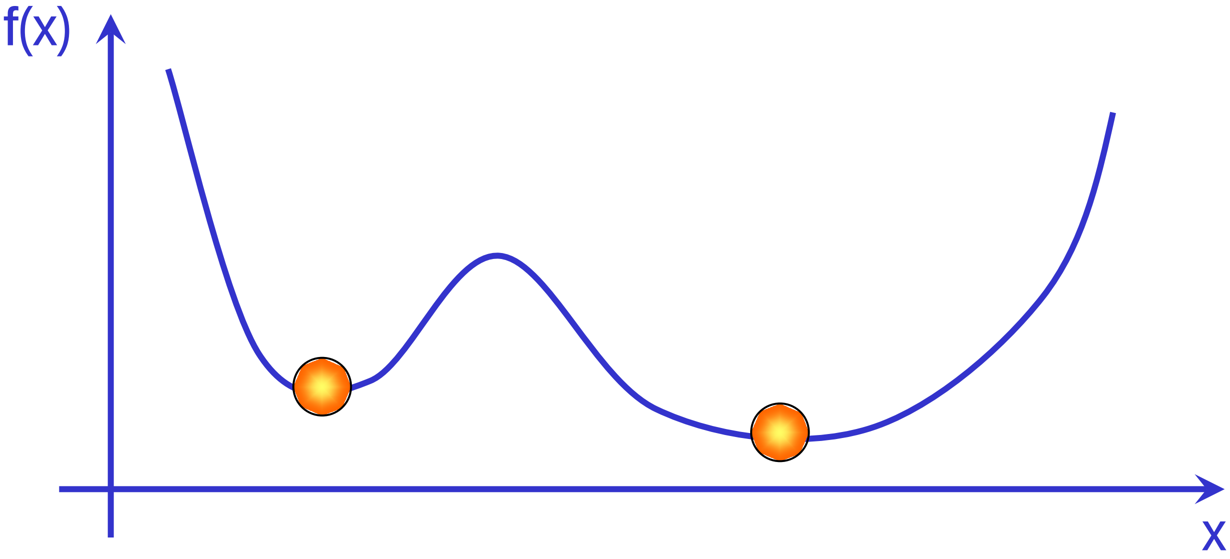
The cost function f(x) keeps decreasing if the derivative is non-zero

Gradient Method

- One-dimensional case (continued)

$$\Delta x = -\lambda \cdot df / dx = 0$$

Derivative is zero at local optimum (gradient method converges)



Gradient Method

- Two-dimensional case

$$\min_{x_1, x_2} \quad f(x_1, x_2)$$

$$\nabla f(x_1, x_2) = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix}$$

$$\begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix} = -\lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

$$\boxed{\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]}$$

Gradient Method

- Two-dimensional case (continued)

$$\begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = -\lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix}$$

⇓

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] - \lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

⇓

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] - \lambda^{(k)} \cdot \left\|\nabla f\left[x_1^{(k)}, x_2^{(k)}\right]\right\|_2^2 \qquad \lambda^{(k)} > 0$$

The cost function $f(x_1, x_2)$ keeps decreasing if the gradient is non-zero

Gradient Method

- N-dimensional case

$$\min_{X} \quad f\left(X\right)$$

$$\nabla f\left(X\right)=\begin{bmatrix}\partial f/\partial x_1\\ \partial f/\partial x_2\\ \vdots\end{bmatrix}$$

$$\boxed{X^{\left(k+1\right)}=X^{\left(k\right)}-\lambda^{\left(k\right)}\cdot\nabla f\left[X^{\left(k\right)}\right]}$$

Newton Method

- Gradient method relies on first-order derivative information
  - Each iteration is simple, but it converges to optimum slowly
  - I.e., require a large number of iteration steps

- The step size $\lambda^{(k)}$ can be optimized by one-dimensional search for fast convergence

$$\min_{\lambda^{(k)}} \quad f\left\{X^{(k)} - \lambda^{(k)} \cdot \nabla f\left[X^{(k)}\right]\right\}$$

- Alternative algorithm: Newton method
  - Rely on both first-order and second-order derivatives
  - Each iteration is more expensive
  - But it converges to optimum more quickly, i.e., requires a smaller number of iterations to reach convergence

Newton Method

- One-dimensional case

$$\min_{x} \quad f(x)$$

$$\left.\frac{df}{dx}\right|_{x^{(k+1)}} \approx \left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \left[x^{(k+1)} - x^{(k)}\right]$$

First-order derivative is zero at local optimum

$$0 = \left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \left[x^{(k+1)} - x^{(k)}\right]$$

Newton Method

- One-dimensional case (continued)

$$\left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \left[x^{(k+1)} - x^{(k)}\right] = 0$$

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$$

$$\boxed{x^{(k+1)} = x^{(k)} - \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}}$$

Newton Method

- One-dimensional case (continued)

$$\Delta x^{(k)} = -\left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \quad \& \quad f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \Delta x^{(k)}$$

$$\Downarrow$$

$$f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] + \left.\frac{df}{dx}\right|_{x^{(k)}} \cdot \left[-\left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \right]$$

$$\Downarrow$$

$$f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] - \underline{\left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1}} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}^{2}$$

Positive (convex function)

Newton Method

- One-dimensional case (continued)

$$x^{(k+1)} = x^{(k)} - \lambda^{(k)} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$$

Gradient method

$$x^{(k+1)} = x^{(k)} - \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}}$$

Newton method

- Newton method gives an estimation of the optimal step size $\lambda^{(k)}$ using second-order derivative

$$\lambda^{(k)} = \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} > 0$$

(convex function)

Newton Method

- One-dimensional case (continued)
  - The step size estimation is based on the following linear approximation for first-order derivative

$$\left.\frac{df}{dx}\right|_{x^{(k+1)}} \approx \left.\frac{df}{dx}\right|_{x^{(k)}} + \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}} \cdot \left[x^{(k+1)} - x^{(k)}\right]$$

  - The approximation is exact if and only if f(x) is quadratic and, therefore, df/dx is linear

$$f(x) = ax^2 + bx + c \quad \Rightarrow \quad \frac{df}{dx} = 2ax + b$$

Newton Method

- One-dimensional case (continued)
  - If the actual f(x) is not quadratic, the following step size estimation may be non-optimal

$$\lambda^{(k)} = \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1}$$

  - Using this step size may result in bad convergence
  - In these cases, a damping factor β is typically introduced

$$x^{(k+1)} = x^{(k)} - \beta \cdot \left.\frac{d^2 f}{dx^2}\right|_{x^{(k)}}^{-1} \cdot \left.\frac{df}{dx}\right|_{x^{(k)}} \qquad (0 < \beta < 1)$$

Newton Method

- Two-dimensional case

$$\min_{x_1, x_2} \quad f\left(x_1, x_2\right)$$

$$\Downarrow$$

$$\nabla f\left(x_1, x_2\right) = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix} \quad \& \quad \nabla^2 f\left(x_1, x_2\right) = \underline{\begin{bmatrix} \partial^2 f / \partial x_1^2 & \partial^2 f / \partial x_1 \partial x_2 \\ \partial^2 f / \partial x_1 \partial x_2 & \partial^2 f / \partial x_2^2 \end{bmatrix}}$$

Hessian matrix

$$\Downarrow$$

$$\nabla f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx \nabla f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right] \cdot \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix}$$

Newton Method

- Two-dimensional case (continued)

$$\nabla f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx \nabla f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right] \cdot \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix} = 0$$

⇓

$$\begin{bmatrix} \Delta x_1^{(k+1)} \\ \Delta x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix} = -\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

⇓

$$\boxed{\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]}$$

Newton Method

- Two-dimensional case (continued)

$$\begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = -\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] + \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix}$$

$$\Downarrow$$

$$f\left[x_1^{(k+1)}, x_2^{(k+1)}\right] \approx f\left[x_1^{(k)}, x_2^{(k)}\right] - \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]^T \cdot \underbrace{\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1}}_{\text{Positive definite (convex function)}} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

Newton Method

- Two-dimensional case (continued)

$$\Delta X^{(k)} = -\lambda^{(k)} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

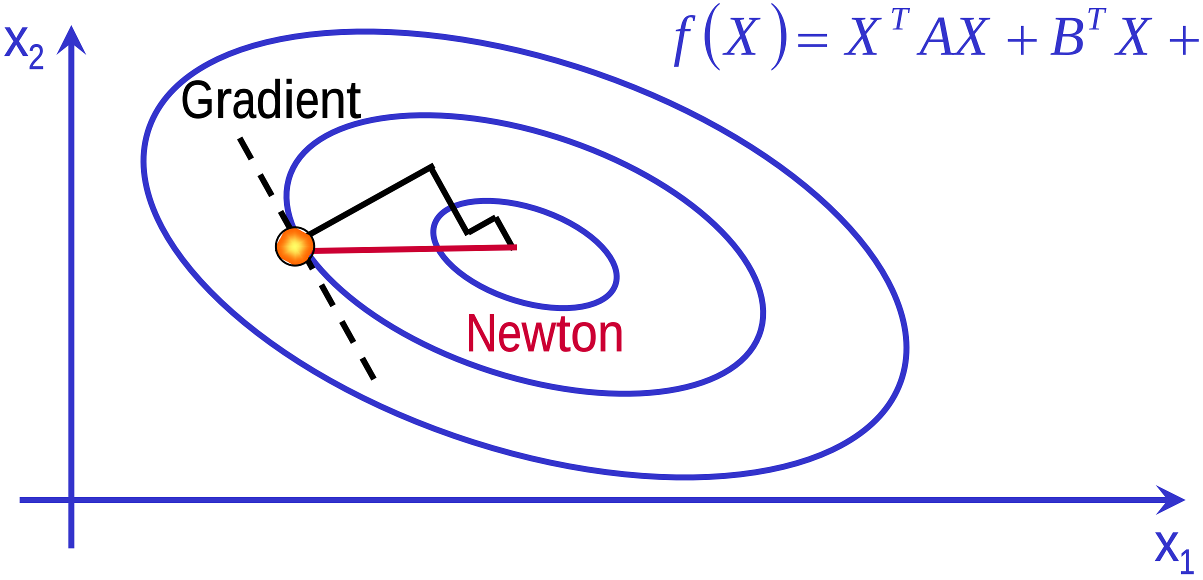
Gradient method

$$\Delta X^{(k)} = -\nabla^2 f\left[x_1^{(k)}, x_2^{(k)}\right]^{-1} \cdot \nabla f\left[x_1^{(k)}, x_2^{(k)}\right]$$

Newton method

  - Gradient method and Newton method do not move along the same direction

$$f(X) = X^T A X + B^T X + C$$



Newton Method

- Newton method can be extended to high-dimensional cases

- The following Hessian matrix is N×N if we have N variables

$$\nabla^2 f(X) = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \cdots & \partial^2 f/\partial x_1 \partial x_N \\ \partial^2 f/\partial x_1 \partial x_2 & \partial^2 f/\partial x_2^2 & \cdots & \partial^2 f/\partial x_2 \partial x_N \\ \vdots & \vdots & \vdots & \vdots \\ \partial^2 f/\partial x_1 \partial x_N & \partial^2 f/\partial x_2 \partial x_N & \cdots & \partial^2 f/\partial x_N^2 \end{bmatrix}$$

- Numerically computing the Hessian matrix and its inverse (by Cholesky decomposition) can be quite expensive for large N

$$X^{(k+1)} = X^{(k)} - \nabla^2 f\left[X^{(k)}\right]^{-1} \cdot \nabla f\left[X^{(k)}\right]$$

Newton Method

- A number of modified algorithms were developed to address this complexity issue
  - E.g., quasi-Newton method

- The key idea is to approximate the Hessian matrix and its inverse so that:
  - The computational cost is significantly reduced
  - Fast convergence can still be achieved

- More details can be found at

Numerical Recipes: The Art of Scientific Computing, 2007

Summary

- Unconstrained Optimization
  - Gradient method
  - Newton method