Overview

- Unconstrained Optimization
  - Gradient method
  - Newton method
Unconstrained Optimization

- Linear regression with regularization
  \[ A\alpha = B \]
  \[ \min_{\alpha} \|A\alpha - B\|_2^2 + \lambda \|\alpha\|_1 \]

- Unconstrained optimization: minimizing a cost function without any constraint
  - Golden section search
  - Downhill simplex method
  - Gradient method
  - Newton method
  \[ \text{Non-derivative method} \]
  \[ \text{Rely on derivatives (this lecture)} \]
If a cost function is smooth, its derivative information can be used to search optimal solution.
Gradient Method

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

\[ \min_x f(x) \]

\[ \Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\lambda^{(k)} \cdot \frac{df}{dx}
\bigg|_{x^{(k)}} \quad (\lambda^{(k)} > 0) \]

Step size  Derivative

\[ x^{(k+1)} = x^{(k)} - \lambda^{(k)} \cdot \frac{df}{dx}
\bigg|_{x^{(k)}} \quad (\lambda^{(k)} > 0) \]
Gradient Method

One-dimensional case (continued)

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

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

\[ f\left[x^{(k+1)}\right] \approx f\left[x^{(k)}\right] - \lambda^{(k)} \cdot \left( \frac{df}{dx}\bigg|_{x^{(k)}} \right)^2 \quad \lambda^{(k)} > 0 \]

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

One-dimensional case (continued)

\[ \Delta x = -\lambda \cdot \frac{df}{dx} = 0 \]

Derivative is zero at local optimum (gradient method converges)
Gradient Method

- Two-dimensional case

\[
\min_{x_1, x_2} f(x_1, x_2)
\]

\[
\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix}
\]

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

\[
\begin{bmatrix}
x_1^{(k+1)} \\
x_2^{(k+1)}
\end{bmatrix} = \begin{bmatrix}
x_1^{(k)} \\
x_2^{(k)}
\end{bmatrix} - \lambda^{(k)} \cdot \nabla f[x_1^{(k)}, x_2^{(k)}]
\]
Gradient Method

Two-dimensional case (continued)

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

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

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

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

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

\(\lambda^{(k)} > 0\)
Gradient Method

- N-dimensional case

\[
\min_{X} f(X)
\]

\[
\nabla f(X) = \begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2} \\
\vdots
\end{bmatrix}
\]

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

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

- 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**

\[
\min_x f(x)
\]

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

First-order derivative is zero at local optimum

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

One-dimensional case (continued)

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

\[
\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = -\frac{d^2 f}{dx^2} \bigg|_{x^{(k)}}^{-1} \cdot \frac{df}{dx} \bigg|_{x^{(k)}}
\]

\[
x^{(k+1)} = x^{(k)} - \frac{d^2 f}{dx^2} \bigg|_{x^{(k)}}^{-1} \cdot \frac{df}{dx} \bigg|_{x^{(k)}}
\]
Newton Method

- One-dimensional case (continued)

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

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

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

Positive (convex function)
Newton Method

One-dimensional case (continued)

Gradient method

Newton method

\[ x^{(k+1)} = x^{(k)} - \lambda^{(k)} \cdot \frac{df}{dx} \bigg|_{x^{(k)}} \]

\[ x^{(k+1)} = x^{(k)} - \frac{d^2 f}{dx^2} \bigg|_{x^{(k)}}^{-1} \cdot \frac{df}{dx} \bigg|_{x^{(k)}} \]

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

\[ \lambda^{(k)} = \frac{d^2 f}{dx^2} \bigg|_{x^{(k)}}^{-1} > 0 \quad \text{(convex function)} \]
One-dimensional case (continued)

The step size estimation is based on the following linear approximation for first-order derivative

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

The approximation is exact if and only if \( f(x) \) is quadratic and, therefore, \( \frac{df}{dx} \) 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 $\beta$ is typically introduced

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

Two-dimensional case

\[
\min_{x_1, x_2} f(x_1, x_2)
\]

\[
\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} \quad \& \quad \nabla^2 f(x_1, x_2) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}
\]

Hessian matrix

\[
\nabla f[x_1^{(k+1)}, x_2^{(k+1)}] \approx \nabla f[x_1^{(k)}, x_2^{(k)}] + \nabla^2 f[x_1^{(k)}, x_2^{(k)}] \begin{bmatrix} x_1^{(k+1)} - x_1^{(k)} \\ x_2^{(k+1)} - x_2^{(k)} \end{bmatrix}
\]
Newton Method

Two-dimensional case (continued)

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

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

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \nabla^2 f [x_1^{(k)}, x_2^{(k)}]^{-1} \cdot \nabla f [x_1^{(k)}, x_2^{(k)}]$$
Two-dimensional case (continued)

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

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

Positive definite (convex function)
Two-dimensional case (continued)

Gradient method

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

Newton method

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

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

- Newton method can be extended to high-dimensional cases.

- The following Hessian matrix is \( N \times N \) if we have \( N \) variables:

\[
\nabla^2 f(X) = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\
\frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_1 \partial x_N} & \frac{\partial^2 f}{\partial x_2 \partial x_N} & \cdots & \frac{\partial^2 f}{\partial x_N^2}
\end{bmatrix}
\]

- 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

Summary

- Unconstrained Optimization
  - Gradient method
  - Newton method