CS 357 - 16 Optimization

Boyang Li (boyangl3)

2 types of optimization: Function $f(\mathbf{x}) : S \to \mathbb{R}$, $S \subset \mathbb{R}^n$, the point $\mathbf{x}^*$ is called the **minimizer** or **minimum** of f if $f(\mathbf{x}^*) \leq f(\mathbf{x}) \, \forall \mathbf{x} \in S$. Generally, there are 2 types of optimization:

1. **Unconstrained:** Find any $\mathbf{x}^*$ to minimize $f(\mathbf{x})$.
2. **Constrained:** Find any $\mathbf{x}^*$ to minimize $f(\mathbf{x})$, and
   (a) $g(\mathbf{x}) = 0$ (Equality Constraints), or
   (b) $g(\mathbf{x}) \leq 0$ (Inequality Constraints)

   While g is another function.

Introduction to maximizer: In order to find the maximizer $\mathbf{x}^*$ for function $f(\mathbf{x})$, we can find the minimizer of function $-f(\mathbf{x})$.

Local and global minima:

- **Local minima:** $\mathbf{x}^*$ is a local minimum if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ in **some subset of domain**.
- **Global minima:** $\mathbf{x}^*$ is a global minimum if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for **all x in the domain**.

Properties for 1D stationary points:

1. **First-order derivative, necessary condition:** $f'(x^*) = 0$
2. **Second-order derivative, sufficient condition:**
   (a) $f''(x^*) > 0$ - Local minima;
   (b) $f''(x^*) < 0$ - Local maxima;
   (c) $f''(x^*) = 0$ - Inflection point.

Unimodal functions:

- **Definition:** A function $f : \mathbb{R} \to \mathbb{R}$ is unimodal on an interval $[a, b]$ if this function has a unique (global) minimum on that interval $[a, b]$.
- **Properties:** Assume $x_1, x_2, x^* \in [a, b]$ and $x_1 < x_2$,
  - ⋄ $x_2 < x^* \Rightarrow f(x_1) > f(x_2)$
  - ⋄ $x^* < x_1 \Rightarrow f(x_1) < f(x_2)$

Golden section search:

- **Motivation:** Reduce the domain to certain $[x_1, x_2]$, find a root of the first order derivative.
- **Iteration steps:**
  1. Starting from initial interval $[a, b]$, take $x_1 = a+(1-\tau)(b-a)$ and $x_2 = a+\tau(b-a)$
  2. Evaluate $f(x_1)$ and $f(x_2)$
  3. Update the interval $[a, b]$
     - If $f(x_1) > f(x_2)$ our new interval would be $[x_1, b]$;
     - If $f(x_1) \leq f(x_2)$ our new interval would be $[a, x_2]$;

  Note: τ is the coefficient that shrinks the interval at constant rate each iteration, in the Golden section search, $\tau = \dfrac{\sqrt{5}-1}{2} \approx 0.618$, which is equal to the Golden Ratio.
- **Convergence:** Linear convergent, $C = \tau \approx 0.618$
- **Cost:** One function evaluation at each iteration because we can reuse x_1 or x_2 as an interior point.
- **"Bracket length":** Define the "bracket length" as $h = b - a$, the "bracket length" after n iterations is $h_n = \tau^n h$.

Newton's method (for optimization):

- **Motivation:** Instead of find the root of $f(x)$, we are finding the root of $f'(x)$.
- **Iteration steps:**
  1. $f'(x_k) + f''(x_k) \cdot h_k = 0 \to h_k = -\dfrac{f'(x_k)}{f''(x_k)}$ (h_k is called Newton step)
  2. $x_{k+1} = x_k + h_k = x_k - \dfrac{f'(x_k)}{f''(x_k)}$ (This step is called Newton update)
- **Convergence:** Quadratic convergence.
- **Cost:** Each iteration we need to evaluate 2 functions, $f'(x_k)$ and $f''(x_k)$.

Hessian matrix (2nd order derivative for N-D functions):

$$\mathbf{H}_f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Properties of N-D stationary points:

1. **Gradient (1st order derivative), necessary condition:** $f'(\mathbf{x}^*) = \vec{0}$
2. **Hessian matrix (2nd order derivative), sufficient condition:**

$\mathbf{H}_f(\mathbf{x}^*)$	**Eigenvalues of** $\mathbf{H}_f(\mathbf{x}^*)$	**Critial point** $\mathbf{x}^*$
Positive definite	All positive	Minimizer
Negative definite	All negative	Maximizer
Indefinite	Indefinite	Saddle point

N-D Steepest descent:

- **Motivation:** The negative of the gradient of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ points downhill (i.e. towards points in the domain having lower values). In other words, given a function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $\mathbf{x}$, the function will decrease its value in the direction of "steepest descent" $-\nabla f$. This hints us to move in the direction of $-\nabla f$ while searching for the minimum until we reach the point where $-\nabla f(\mathbf{x}) = \vec{0}$.

  We know the direction we need to move to approach the minimum but we still do not know the distance we need to move in order to approach the minimum. If $\mathbf{x}_k$ was our earlier point then we select the next guess by moving it in the direction of the negative gradient:

  $$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha(-\nabla f(\mathbf{x}_k)).$$

  The next problem would be to find the α, and we use the 1-dimensional optimization algorithms to find the required α. Hence, the problem translates to:

  $$\mathbf{s} = -\nabla f(\mathbf{x}_k) \min_{\alpha} \left(f\left(\mathbf{x}_k + \alpha \mathbf{s}\right)\right)$$

- **Iteration steps:**

  1. Evaluate deepest descent: $\mathbf{s}_k = -\nabla f(\mathbf{x}_k)$
  2. Perform a line search to obtain α_k (for example, Golden Section Search):

     $$\alpha_k = \underset{\alpha}{\operatorname{argmin}} f(\mathbf{x}_k + \alpha \mathbf{s}_k)$$

  3. Update: $\mathbf{x}_k = \alpha_k \mathbf{s}_k$

- **Convergence:** Linear convergence.
- **Side note:** $\nabla f(\mathbf{x}_{k+1})$ is orthogonal to $\nabla f(\mathbf{x}_k)$.

N-D Newton's method:

- **Motivation:** Instead of find the root of $f(\mathbf{x})$, we are finding the root of $\nabla f(\mathbf{x})$.
- **Iteration steps:**
  1. $\nabla f(\mathbf{x}_k) + \mathbf{H}(\mathbf{x}_k) \cdot \mathbf{s}_k = 0 \to \mathbf{s}_k = -\mathbf{H}(\mathbf{x}_k)^{-1} \cdot \nabla f(\mathbf{x}_k)$ ($\mathbf{s}_k$ can also be calculated by solving the system $\mathbf{H}(\mathbf{x}_k) \cdot \mathbf{s}_k = -\nabla f(\mathbf{x}_k)$).
  2. $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1} \cdot \nabla f(\mathbf{x}_k)$
- **Convergence:** Quadratic convergence.
- **Cost:** The cost per iteration is $\mathcal{O}(n^3)$, the cost of evaluate Hessian matrix is $\mathcal{O}(n^2)$.
- **Drawbacks:**
  - Need to evaluate 2nd order derivative
  - Only converges locally
  - Works poorly when Hessian is nearly indefinite

SUMMARY: Computational cost (only CORRECT statements):

✓ Secant method reuses result from previous iterations to save computational resources.

✓ Secant method and bisection method have similar computational cost per iteration after several iterations.

✓ As the number of iteration increases, Newton's method is the most computationally expensive one among the three methods.

SUMMARY: Convergence (only CORRECT statements):

✓ Secant method has superlinear convergence and has a lower cost for each iteration compared to Newton's methods.

✓ Bisection method has linear convergence if $f(x)$ is continuous within an interval $[a, b]$ such that $f(a)f(b) < 0$.

✓ Newton's method has quadratic convergence when it is close to the root.

✓ Of all three methiods, Newton's method has the fastest convergence.

SUMMARY: Newton's method (1-D, N-D, find root, optimization):

	Root/solution finding	**Optimization**
1D	$x_{k+1} = x_k - \dfrac{f(x)}{f'(x)}$	$x_{k+1} = x_k - \dfrac{f'(x)}{f''(x)}$
N-D	$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbb{J}(\mathbf{x}_k)^{-1} \cdot f(\mathbf{x}_k)$	$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}(\mathbf{x}_k)^{-1} \cdot \nabla f(\mathbf{x}_k)$

- **Convergence:** All Newton's methods have quadratic convergence.
- **Cost:** For each iteration, we need to evaluate 2 functions.
  - For N-D root finding, the cost of find $\mathbf{s}$ by solving a linear system is $\mathcal{O}(n^3)$; the cost of evaluation Jacobian matrix is $\mathcal{O}(n^2)$
  - For N-D optimization, the cost per iteration is $\mathcal{O}(n^3)$, the cost of evaluate Hessian matrix is $\mathcal{O}(n^2)$.
- **Drawbacks:**
  - Function need to be differentiable
  - Expensive cost to evaluate the function and derivative
  - Only converges locally (and may converges to a maxima or inflection point)
- **Concept questions on the exam (only CORRECT statements are listed):**
  - ✓ Newton's method may fail to find the minimum if the start guess is close to a maximum.
  - ✓ Newton's method will fail if the second derivative of $f(x)$ at some point x_i is 0.
  - ✓ Newton's method will not always converge.
  - ✓ The cost of solving the nonlinear system of n equations for one iteration is $\mathcal{O}(n^3)$.
  - ✓ Newton's method for solving n nonlinear equations typically has quadratic convergence when close enough to the root.
  - ✓ Newthon's method can usually achieve quadratic convergence when the start guess is near the minimum.
  - ✓ The cost of evaluation Jacobian matrix is $\mathcal{O}(n^2)$
  - ✓ The cost of calculating the Hessian matrix is $\mathcal{O}(n^2)$
  - ✓ Newton method needs to evaluate derivatives.
  - ✓ Newton's method requires two function calls per iteration.