Operations Research II: Algorithms

Gradient Descent and Newton's Method

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Road map

- **Introduction**.
- Gradient descent.
- Newton's method.

Algorithms for nonlinear programming

- In many cases, we need to solve NLPs.
  - We rely on **numerical algorithms** for obtaining a numerical solution.
  - Typically the focus on an engineering application.
- To apply an algorithm, we need to first get the values of all parameters.
- Many NLP algorithms run in the following way:
  - **Iterative**: The algorithm moves to a point in one iteration, and then starts the next iteration starting from this point.
  - **Repetitive**: In each iteration, it repeat some steps.
  - **Greedy**: In each iteration, it seeks for some "best" thing achievable in that iteration.
  - **Approximation**: Relying on first-order or second-order approximation of the original program.

Limitations of NLP algorithms

- NLP algorithms certainly have their limitations.
- It may **fail to converge**.
  - An algorithm converges to a solution if further iterations do not modify the current solution "a lot."
  - Sometimes an algorithm may fail to converge at all.
- It may be trapped in a **local optimum**.
  - A serious problem for general NLPs.
  - The starting point matters.
  - Some algorithms play some tricks to "try" several local optima.
- It (typically) requires the domain to be **continuous and connected**.
  - A nonlinear integer program is very hard to solve.
- We will point out these difficulties.
  - Remedies are beyond the scope of this course.

Assumptions

- In today's lecture, we will introduce two algorithms to solve **unconstrained** NLP.
  - **Gradient descent**.
  - **Newton's method**.
- We will solve

$$\min_{x \in \mathbb{R}^n} f(x)$$

  where $f(\cdot)$ is a **twice-differentiable** function.
- Our next step is to learn about **gradients and Hessians**, which are the bases of gradient descent and Newton's method.

Gradients and Hessians

- For a function $f : \mathbb{R}^n \to \mathbb{R}$, collecting its first- and second-order partial derivatives generates its **gradient** and **Hessian**:

Definition 1 (Gradients and Hessians)

For a multi-variate twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, its gradient and Hessian are

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \quad \text{and} \quad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \ddots & \\ \vdots & & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}.$$

- In this course, all Hessians are **symmetric**.

Example

- For $f(x_1, x_2, x_3) = x_1^2 + x_2 x_3 + x_3^3$, the gradient is

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \frac{\partial f(x)}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ x_3 \\ x_2 + 3x_3^2 \end{bmatrix}.$$

- The Hessian is

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f(x)}{\partial x_3 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_3 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_3^2} \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 6x_3 \end{bmatrix}.$$

- What are $\nabla f(3, 2, 1)$ and $\nabla^2 f(3, 2, 1)$?

Road map

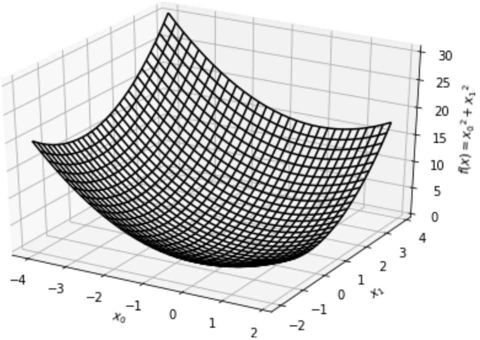
- Introduction.
- **Gradient descent**.
- Newton's method.

Gradient descent

- We first introduce the **gradient descent** method.
- Given a current solution $x \in \mathbb{R}^n$, consider its gradient

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}.$$

- The gradient is an n-dimensional vector. We may try to "improve" our current solution by moving along this direction.



Gradient is an increasing direction

- Is the gradient an improving direction?

Proposition 1

For a twice-differentiable function $f(x)$, its gradient $\nabla f(x)$ is an increasing direction, i.e., $f(x + a\nabla f(x)) > f(x)$ for all $a > 0$ that is small enough.

Proof. Recall that

$$\lim_{a \to 0} \frac{f(x + ad) - f(x)}{a} = d\nabla f(x).$$

Therefore, we have $\lim_{a \to 0} \frac{f(x+a\nabla f(x)) - f(x)}{a} = \nabla f(x)^{\mathrm{T}} \nabla f(x) > 0$, which means that if a is small enough, $f(x + a\nabla f(x))$ is greater than $f(x)$. $\square$

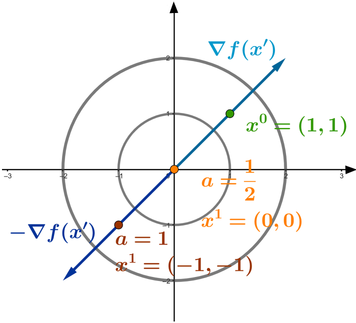
- In fact the gradient is the **fastest increasing direction**.

Gradient is an increasing direction

- Given that the gradient is an increasing direction, we should move along its opposite direction (for a minimization problem).
- Therefore, given a current solution x:
  - In each iteration we update it to
$$x - a\nabla f(x)$$
    for some value $a > 0$. a is called the **step size**.
  - We stop when the gradient of a current solution is 0.
- Question: How to choose an appropriate value of a?
- Before we answer this question, let's see an example.

A bad step size can be very bad

- Let's solve
$$\min_{x \in \mathbb{R}^2} \ f(x) = x_1^2 + x_2^2.$$
- Suppose we starts at $x^0 = (1, 1)$.
  - The gradient in general is $\nabla f(x) = (2x_1, 2x_2)$.
  - The gradient at x^0 is $\nabla f(x^0) = (2, 2)$.
- If we set $a = \frac{1}{2}$, we will move from x^0 to $x^1 = (1, 1) - \frac{1}{2}(2, 2) = (0, 0)$. Optimal!
- If we set $a = 1$, we will move to $x^1 = (1, 1) - (2, 2) = (-1, -1)$.
  - The gradient at x^1 is $\nabla f(x^1) = (-2, -2)$.
  - We move to $x^2 = (-1, -1) - (-2, -2) = (1, 1)$.
  - The algorithm does not converge.



Maximizing the improvement

- How to choose a step size?
- We may instead look for the **largest improvement**.
  - Along our improving direction $-\nabla f(x)$, we solve

$$\min_{a \geq 0} \; f(x - a\nabla f(x))$$

    to see how far we should go to reach the lowest point along this direction.
- We now may describe our **gradient descent algorithm**.
- Step 0: Choose a starting point x^0 and a precision parameter $\epsilon > 0$.
- Step $k + 1$:
  - Find $\nabla f(x^k)$.
  - Solve $a_k = \operatorname{argmin}_{a \geq 0} \; f(x^k - a\nabla f(x^k))$.
  - Update the current solution to $x^{k+1} = x^k - a_k \nabla f(x^k)$,
  - If $||\nabla f(x^{k+1})|| < \epsilon$, stop; otherwise let k become $k + 1$ and continue.[1]

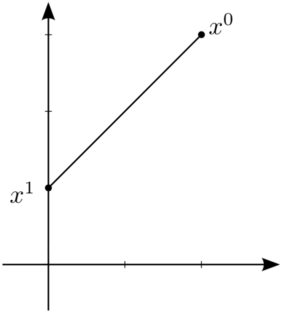
[1] For $x \in \mathbb{R}^n$, $||x|| = \sqrt{x_1^2 + \cdots + x_n^2}$.

Example 1

- Let's solve $\min \ f(x) = 4x_1^2 - 4x_1x_2 + 2x_2^2$.
  - The optimal solution is $x^* = (0, 0)$.
  - We have $\nabla f(x) = (8x_1 - 4x_2, -4x_1 + 4x_2)$
- Step 0: $x^0 = (2, 3)$. $f(x^0) = 10$.
- Step 1:
  - $\nabla f(x^0) = (4, 4)$.
  - $a_0 = \text{argmin}_{a \geq 0} \ f(x^0 - a\nabla f(x^0))$, where

$$\begin{aligned} f(x^0 - a\nabla f(x^0)) &= f(2 - 4a, 3 - 4a) \\ &= 32a^2 - 32a + 10. \end{aligned}$$

    It follows that $a_0 = \frac{1}{2}$.
  - $x^1 = x^0 - a_0\nabla f(x^0) = (2, 3) - \frac{1}{2}(4, 4) = (0, 1)$. Note that $f(x^1) = 2$.
  - $||\nabla f(x^1)|| = ||(-4, 4)|| = 4\sqrt{2}$.

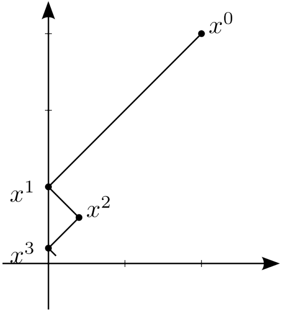


Example 1

- Step 2:
  - $\nabla f(x^1) = (-4, 4)$.
  - $a_1 = \text{argmin}_{a \geq 0} \; f(x^1 - a\nabla f(x^1))$, where

    $$\begin{aligned} f(x^1 - a\nabla f(x^1)) &= f(0 + 4a, 1 - 4a) \\ &= 160a^2 - 32a + 2. \end{aligned}$$

    It follows that $a_1 = \frac{1}{10}$.
  - $x^2 = x^1 - a_1 \nabla f(x^1) = (0, 1) - \frac{1}{10}(-4, 4) = (\frac{2}{5}, \frac{3}{5})$. Note that $f(x^2) = \frac{2}{5}$.
  - $||\nabla f(x^2)|| = ||(\frac{4}{5}, \frac{4}{5})|| = \frac{4\sqrt{2}}{5}$.



Example 2

- Let's solve $\min_{x \in \mathbb{R}^2} \; f(x) = x_1^2 - 2x_1x_2 + 2x_2^2 + 2x_1$.
  - We have $\nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 - 2x_2 + 2 \\ -2x_1 + 4x_2 \end{bmatrix}$.
  - We are searching for a point x^* that satisfies $\nabla f(x^*) = 0$. This implies that $(x_1^*, x_2^*) = (-2, -1)$.
- Step 0: $x^0 = (0, 0)$.
- Step 1:
  - We have

$$\nabla f(x_1^0, x_2^0) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}.$$

  - We may obtain an optimal value for a_0 by solving

$$a_0 = \operatorname*{argmin}_{a \geq 0} f(x^0 - a\nabla f(x^0)) = 4a^2 - 4a,$$

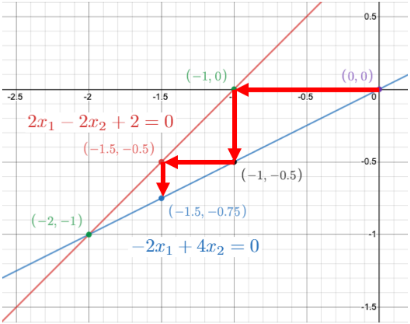
it follows that $a_0 = \frac{1}{2}$. Therefore, $x^1 = (0, 0) - \frac{1}{2}(2, 0) = (-1, 0)$.

Example 2

- Step 2:
  - We have
$$\nabla f(x_1^1, x_2^1) = \begin{bmatrix} 0 \\ 2 \end{bmatrix}.$$
  - We may obtain an optimal value for a_1 by solving
$$a_1 = \operatorname*{argmin}_{a \geq 0} f(x^1 - a\nabla f(x^1)) = 8a^2 - 4a - 1,$$
it follows that $a_1 = \frac{1}{4}$. Therefore, $x^2 = (-1, 0) - \frac{1}{4}(0, 2) = (-1, -\frac{1}{2})$.

Example 2

- By depicting the search route from x^0 to x^1 to x^2, and obtaining optimal x^* by the FOC, we know that the algorithm search only one direction once a time.
- Thus, we can predict x^3 is on $2x_1 - 2x_2 + 2 = 0$ with $x_2 = -\frac{1}{2}$, so $x^3 = (-\frac{3}{2}, -\frac{1}{2})$. And $x^4 = (-\frac{3}{2}, -\frac{3}{4})$. By doing more iterations, you can get the optimal solution $x^* = (-2, -1)$.



Road map

- Introduction.
- Gradient descent.
- **Newton's method**.

Newton's method

- The gradient descent method is a **first-order** method.
  - It relies on the gradient to improve the solution.
- A first-order method is intuitive, but sometimes too slow.
- A **second-order** method relies on the Hessian to update a solution.
- We will introduce one second-order method: **Newton's method**.
- Let's start from Newton's method for solving a **nonlinear equation**.

Newton's method for a nonlinear equation

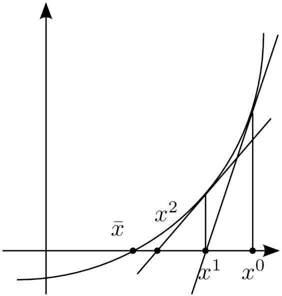
- Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable. We want to find $\bar{x}$ satisfying $f(\bar{x}) = 0$.
- For any x^k, let

$$f_L(x) = f(x^k) + f'(x^k)(x - x^k)$$

be the **linear approximation** of f at x^k.
  - This is the **tangent line** of f at x^k or the first-order **Taylor expansion** of f at x^k.
- We move from x^k to x^{k+1} by setting

$$f_L(x^{k+1}) = f(x^k) + f'(x^k)(x^{k+1} - x^k) = 0.$$

- We will keep iterating until $|f(x^k)| < \epsilon$ or $|x^{k+1} - x^k| < \epsilon$ for some predetermined $\epsilon > 0$.



Newton's method for single-variate NLPs

- Let f be twice differentiable. We want to find $\bar{x}$ satisfying $f'(\bar{x}) = 0$.
- For any x^k, let

$$f'_L(x) = f'(x^k) + f''(x^k)(x - x^k)$$

  be the **linear approximation** of f' at x^k.
- To approach $\bar{x}$, we move from x^k to x^{k+1} by setting

$$f'_L(x^{k+1}) = f'(x^k) + f''(x^k)(x^{k+1} - x^k) = 0.$$

- We will keep iterating until $|f'(x^k)| < \epsilon$ or $|x^{k+1} - x^k| < \epsilon$ for some predetermined $\epsilon > 0$.
- Note that $f'(\bar{x})$ does not guarantee a global minimum.
  - That is why showing f is convex is useful!

Another interpretation

- Let f be twice differentiable. We want to find $\bar{x}$ satisfying $f'(\bar{x}) = 0$.
- For any x^k, let

$$f_Q(x) = f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2}f''(x^k)(x - x^k)^2$$

  be the **quadratic approximation** of f at x^k.
  - This is the second-order **Taylor expansion** of f at x^k.
- We move from x^k to x^{k+1} by moving to the **global minimum** of the quadratic approximation, i.e.,

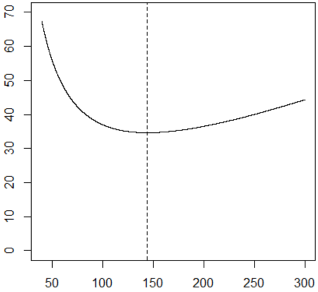
$$x^{k+1} = \operatorname*{argmin}_{x \in \mathbb{R}} \ f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2}f''(x^k)(x - x^k)^2,$$

- Differentiating the above objective function with respect to x, we have

$$f'(x^k) + f''(x^k)(x^{k+1} - x^k) = 0 \quad \Leftrightarrow \quad x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)}.$$

Example: the NLP

- Let
$$f = \frac{KD}{x} + \frac{hx}{2},$$
where $K = 5$, $D = 500$, and $h = 0.24$.
- The global minimum is
$$x^* = \sqrt{\frac{2KD}{h}} \approx 144.34.$$

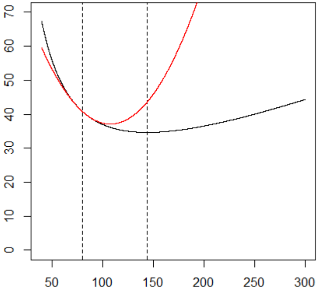


Example: quadratic approximation

- At any x^k, the quadratic approximation is

$$f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2} f''(x^k)(x - x^k)^2$$

$$= \left(\frac{KD}{x^k} + \frac{hx^k}{2} \right) + \left(\frac{-KD}{(x^k)^2} + \frac{h}{2} \right)(x - x^k) + \frac{1}{2} \left(\frac{2KD}{(x^k)^3} \right)(x - x^k)^2.$$



- E.g., at $x^0 = 80$, it is (approximately)

$$40.85 - 0.27(x - 80) + 0.0098(x - 80)^2.$$

Example: one iteration

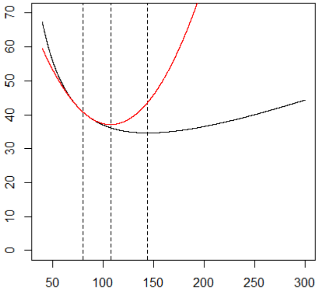
- At any x^k, the quadratic approximation may be obtained.
- Its global minimum x^{k+1} satisfies

$$\left(\frac{-KD}{(x^k)^2} + \frac{h}{2}\right) + \left(\frac{2KD}{(x^k)^3}\right)(x^{k+1} - x^k) = 0.$$

- E.g., at $x^0 = 80$, we have

$$-0.27 + 0.0098(x^1 - 80) = 0,$$

i.e., $x^1 \approx 101.71$.

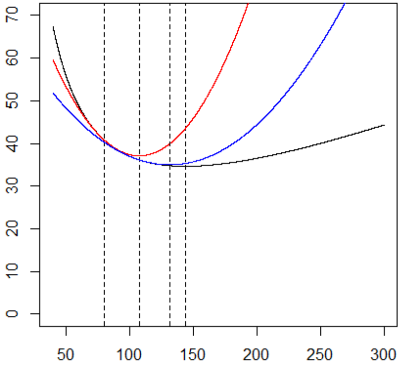


Example: one more iteration

- Note that from x^k we may simply move to

$$x^{k+1} = x^k - \frac{\frac{-KD}{(x^k)^2} + \frac{h}{2}}{\frac{2KD}{(x^k)^3}}.$$

- From $x^1 = 101.71$, we will move to $x^2 = 131.58$.
- We get closer to $x^* = 144.34$.



Newton's method for multi-variate NLPs

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable.
- For any x^k, let

$$f_Q(x) = f(x^k) + \nabla f(x^k)^{\mathrm{T}}(x - x^k) + \frac{1}{2}(x - x^k)^{\mathrm{T}}\nabla^2 f(x^k)(x - x^k)$$

  be the quadratic approximation of f at x^k.
  - Note that we use the **Hessian** $\nabla^2 f(x^k)$.
- We move from x^k to x^{k+1} by moving to the global minimum of the quadratic approximation:

$$\nabla f(x^k) + \nabla^2 f(x^k)(x^{k+1} - x^k) = 0,$$

  i.e.,

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

Example

- Let's minimize $f(x) = x_1^4 + 2x_1^2x_2^2 + x_2^4$.
  - The optimal solution is $x^* = (0, 0)$.
  - $\nabla f(x) = \begin{bmatrix} 4x_1^3 + 4x_1x_2^2 \\ 4x_1^2x_2 + 4x_2^3 \end{bmatrix}$ and $\nabla^2 f(x) = \begin{bmatrix} 12x_1^2 + 4x_2^2 & 8x_1x_2 \\ 8x_1x_2 & 12x_2^2 + 4x_1^2 \end{bmatrix}$.
- Suppose that $x^0 = (b, b)$ for some $b > 0$.
  - We have $\nabla f(x^0) = \begin{bmatrix} 8b^3 \\ 8b^3 \end{bmatrix}$ and $\nabla^2 f(x^0) = \begin{bmatrix} 16b^2 & 8b^2 \\ 8b^2 & 16b^2 \end{bmatrix}$.
  - Therefore, we have

$$\begin{aligned} x^1 &= x^0 - \left[\nabla^2 f(x^0)\right]^{-1} \nabla f(x^0) \\ &= \begin{bmatrix} b \\ b \end{bmatrix} - \frac{1}{192b^2} \begin{bmatrix} 16 & -8 \\ -8 & 16 \end{bmatrix} \begin{bmatrix} 8b^3 \\ 8b^3 \end{bmatrix} = \begin{bmatrix} \frac{2}{5}b \\ \frac{2}{5}b \end{bmatrix}. \end{aligned}$$

  - In fact, we have $x^k = \left((\frac{2}{5})^k b, (\frac{2}{5})^k b\right)$.

Remarks

- For Newton's method:
  - Newton's method does not have the step size issue.
  - It in many cases is faster.
  - For a quadratic function, Newton's method find an optimal solution in one iteration.
  - It may fail to converge for some functions.
- More issues in general:
  - Convergence guarantee.
  - Convergence speed.
  - Non-differentiable functions.
  - Constrained optimization.