HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

FUNDAMENTALS OF OPTIMIZATION

FUNDAMENTALS OF OPTIMIZATION

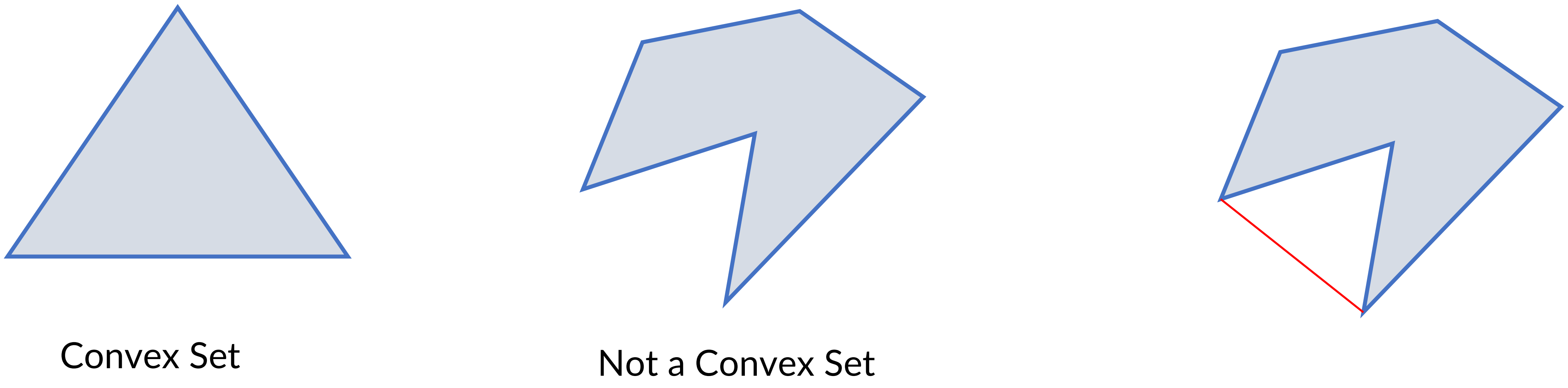
Week 2: Convex Optimization

Outline

1. **Definitions**
    - Convex set
    - Convex combination
    - Convex function
    - Exercises
2. Unconstrained Optimization
3. Constrained Optimization

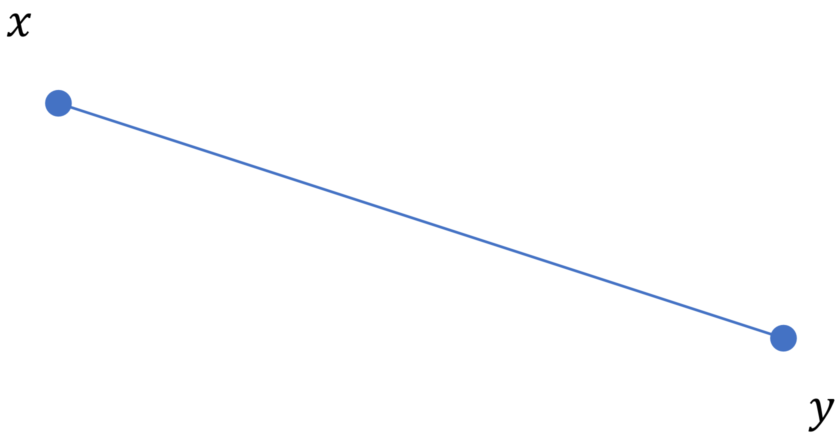
Convex set

- A set of points $C \subseteq \mathbb{R}^n$ is **convex** if for every pair of points $x, y \in C$, any point on the line segment between x and y is also in C
- That is, if $x, y \in C$, then $tx + (1 - t)y \in C$ for all $t \in [0,1]$



Examples of convex set

- **Line Segments:** $C = \{x + t(y - x) \mid t \in [0,1]\}$ for some $\mathrm{x}, y \in \mathbb{R}^n$



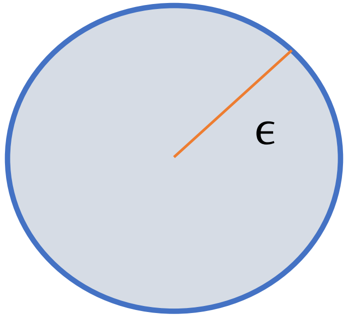
- Lines, planes, hyperplanes, etc. also define convex sets

Examples of convex set

- **Half spaces:** $C = \{x \in \mathbb{R}^n \mid w^T x + b \le 0\}$ for some $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$
- **Balls of Radius ϵ:** $C = \{x \in \mathbb{R}^n \mid \|x\|_2 \le \epsilon\}$ for some $\epsilon \ge 0 \in \mathbb{R}$

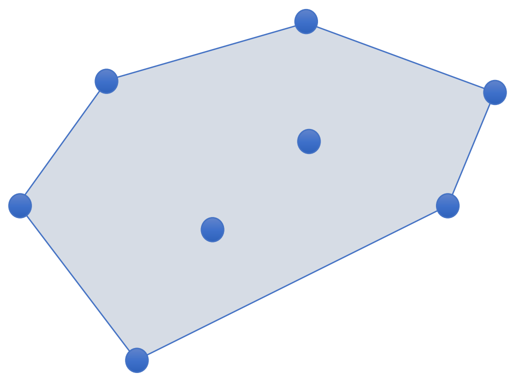
$$\|x\|_2 = \sqrt{x^T x} = \sqrt{\sum_{i=1}^{n} x_i^2}$$

  - This is called the Euclidean norm or Euclidean distance because $\|x - y\|_2$ is equal to the length of the line segment between the points x and y



Convex combination

- We say that $y \in \mathbb{R}^n$ is a **convex combination** of the points $x^{(1)}, \dots, x^{(k)} \in \mathbb{R}^n$ if $y = \lambda_1 x^{(1)} + \cdots + \lambda_k x^{(k)}$ for some choice of $\lambda_1, \dots, \lambda_k \in [0,1]$ such that $\lambda_1 + \cdots + \lambda_k = 1$
- Let C be the set of all points y that can be obtained as a convex combination of the $x^{(1)}, \dots, x^{(k)}$
  - In the special case $k = 2$, C is just a line segment
  - C is a convex set called the **convex hull** of the $x^{(1)}, \dots, x^{(k)}$



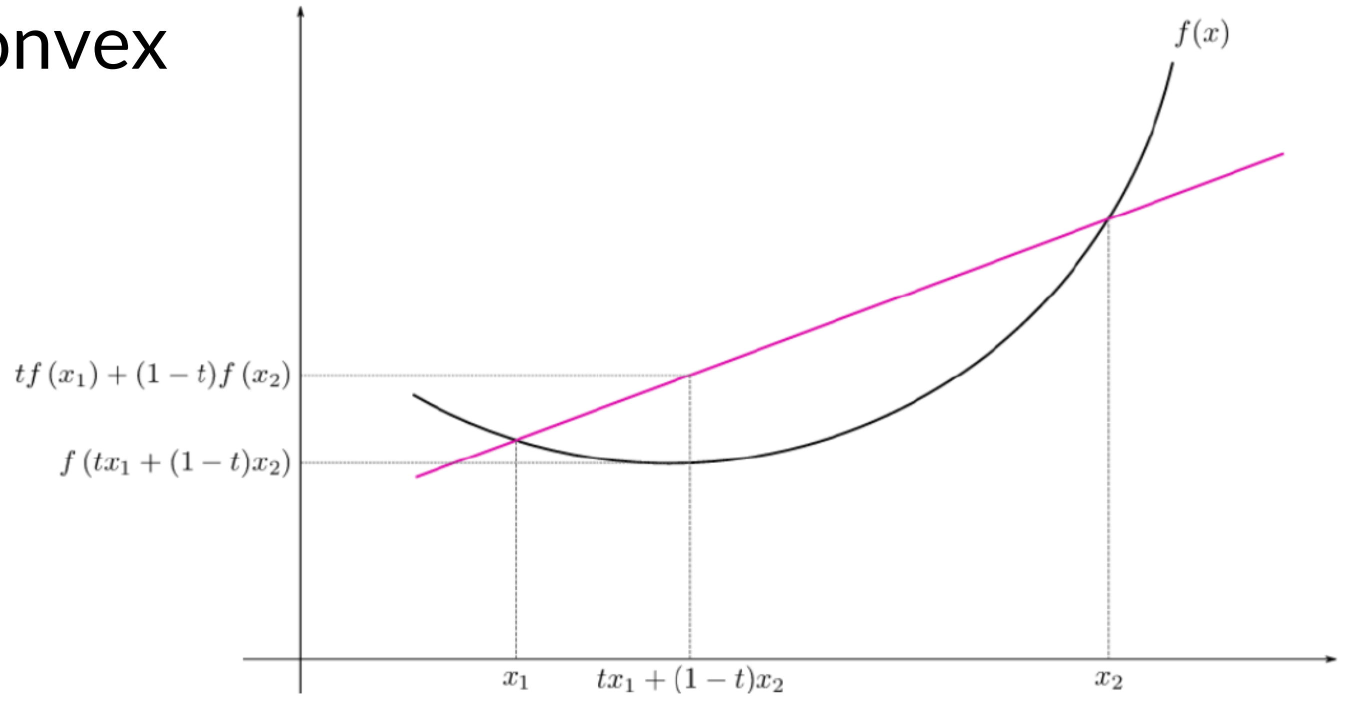
Convex function

- A **function** $f: C \to \mathbb{R}$ is **convex** if C is a convex set and

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

for all $x, y \in C$ and $t \in [0,1]$

- f is called **concave** if $-f$ is convex



Exercises

1. $S = \{(x, y) \mid 0 \le x \le a, 0 \le y \le b\}$ is a convex set ?
2. *Union* of two convex sets is a convex set ?
3. $S = \{x \in R^n \mid ||x - a|| \le r\}$ is a convex set ?
4. $S = \{(x, y, z) | \ z \ge x^2 + y^2\}$ is a convex set ?

$S = \{(x, y) \mid 0 \le x \le a, 0 \le y \le b\}$ is a convex set ?

Let (x_1, y_1) and (x_2, y_2) be two arbitrary points in S. This means:

$$0 \le x_1 \le a, \quad 0 \le y_1 \le b, \quad 0 \le x_2 \le a, \quad 0 \le y_2 \le b.$$

Consider any convex combination of these points:

$$x_\lambda = \lambda x_1 + (1 - \lambda) x_2, \quad y_\lambda = \lambda y_1 + (1 - \lambda) y_2,$$

where $\lambda \in [0, 1]$.

- Since $x_1, x_2 \in [0, a]$, the weighted sum x_λ also satisfies $0 \le x_\lambda \le a$.
- Similarly, since $y_1, y_2 \in [0, b]$, the weighted sum y_λ satisfies $0 \le y_\lambda \le b$.

Thus, every point on the line segment between (x_1, y_1) and (x_2, y_2) remains in S, proving that S is convex.

Union of two convex sets is a convex set ?

Consider two convex sets:

$$S_1 = \{(x, y) \mid x \geq 0, y = 1\}$$

$$S_2 = \{(x, y) \mid x \leq 0, y = -1\}$$

- S_1 is a horizontal line at $y = 1$, which is convex.
- S_2 is a horizontal line at $y = -1$, which is also convex.

Now, take two points:

- $A = (1, 1) \in S_1$
- $B = (-1, -1) \in S_2$

The line segment joining A and B is given by:

$$(x_\lambda, y_\lambda) = \lambda(1, 1) + (1 - \lambda)(-1, -1) = (2\lambda - 1, 2\lambda - 1), \quad \lambda \in [0, 1].$$

For values of λ strictly between 0 and 1, the intermediate points **do not belong to** S_1 or S_2. Hence, the union $S_1 \cup S_2$ is **not convex**.

Convex properties

- **Nonnegative weighted sums** of convex functions are convex, i.e., if $f_1: \mathbb{R}^n \to \mathbb{R}$ and $f_2: \mathbb{R}^n \to \mathbb{R}$ are convex functions and $c_1, c_2 \geq 0$, then
$$g(x) = c_1 f_1(x) + c_2 f_2(x)$$
  is a convex function.

- **Pointwise maximum** of convex functions are convex, i.e., if $f_1: \mathbb{R}^n \to \mathbb{R}$ and $f_2: \mathbb{R}^n \to \mathbb{R}$ are convex functions, then
$$g(x) = \max(f_1(x), f_2(x))$$
  is a convex function.

Convex properties

- A differentiable function $f: \mathbb{R}^n \to \mathbb{R}$ is convex on a convex set C if and only if

$$f(x) \geq f(y) + \nabla f(y)^T (x - y)$$

  for all $x, y \in C$

Exercises

- Which of the following functions are convex?
  - $\exp(x)$
  - $\exp(-x)$
  - $\log(x)$
  - $\sin(x)$
  - x^2
  - x^8
  - $\max(x, 0)$
  - $\sqrt{x}$
  - $|x|$

Exercises

- Which of the following functions are convex?
  - $\exp(x)$
  - $\exp(-x)$
  - $\log(x)$
  - $\sin(x)$
  - x^2
  - x^8
  - $\max(x, 0)$
  - $\sqrt{x}$
  - $|x|$

Outline

1. **Definitions**
2. Unconstrained Optimization
   - Introduction to unconstrained optimization
   - Descent method
   - Gradient descent method
   - Newton method
3. Constrained Optimization

Unconstrained convex optimization

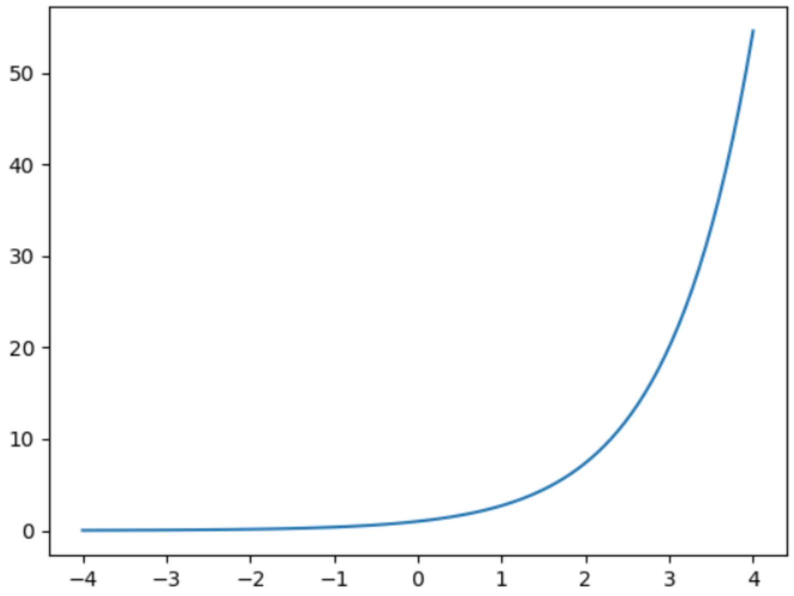
- Unconstrained, smooth convex optimization problem:

$$\min f(x)$$

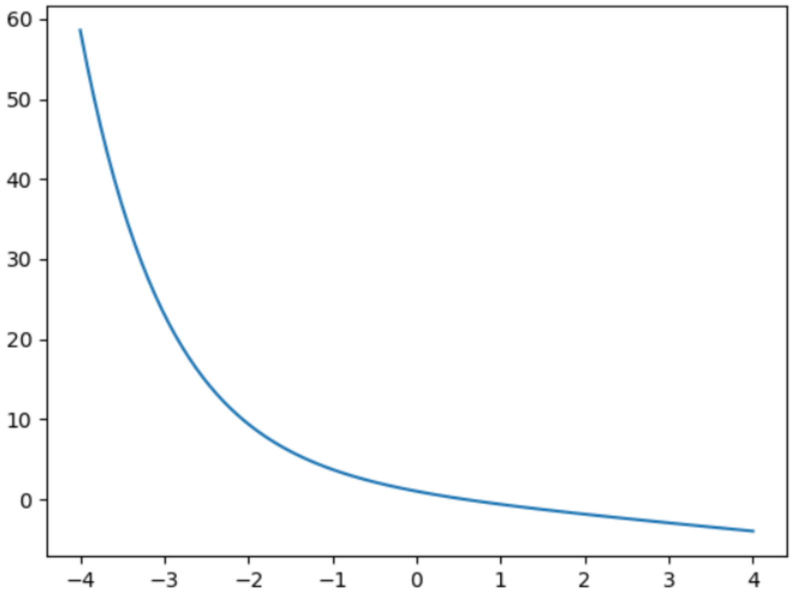
  - f: $R^n \rightarrow R$ is convex and twice differentiable
  - ***dom*** $f = R$: no constraint
  - Assumption: the problem is solvable with $f^* = \min_x f(x)$ and $x^* = \mathrm{argMin}_x f(x)$
- To find x, solve equation $\nabla f(x^*) = 0$: not easy to solve analytically
- Iterative scheme is preferred: compute minimizing sequence $x^{(0)}, x^{(1)}, \ldots$ s.t. $f(x^{(k)}) \rightarrow f(x^*)$ as $k \rightarrow \infty$
- The algorithm stops at some point $x(k)$ when the error is within acceptable tolerance: $f(x^{(k)}) - f^* \le \varepsilon$

Local minimizer

- x^* is a local minimizer for f: $R^n \rightarrow R$ if $f(x^*) \leq f(x)$ for $||x^*-x|| \leq \varepsilon$ ($\varepsilon > 0$ is a constant)
- x^* is a global minimizer for f: $R^n \rightarrow R$ if $f(x^*) \leq f(x)$ for all $x \in R^n$



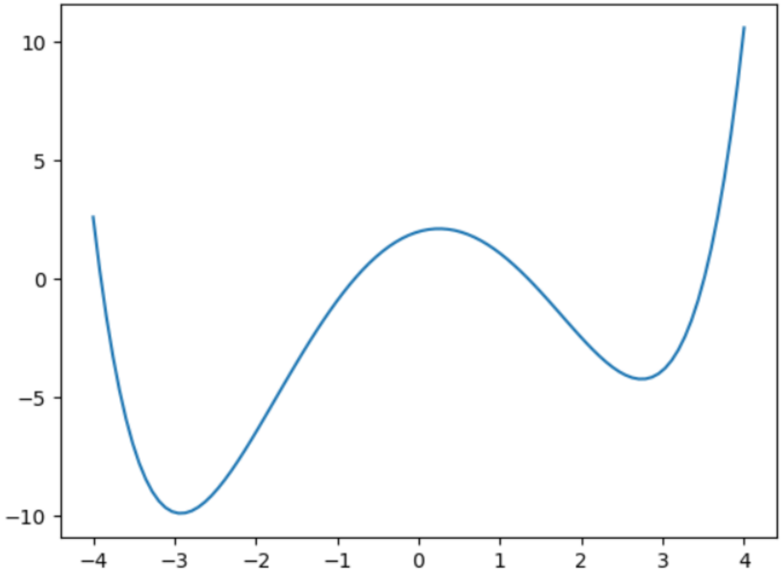
$f(x) = e^x$ has no minimizer



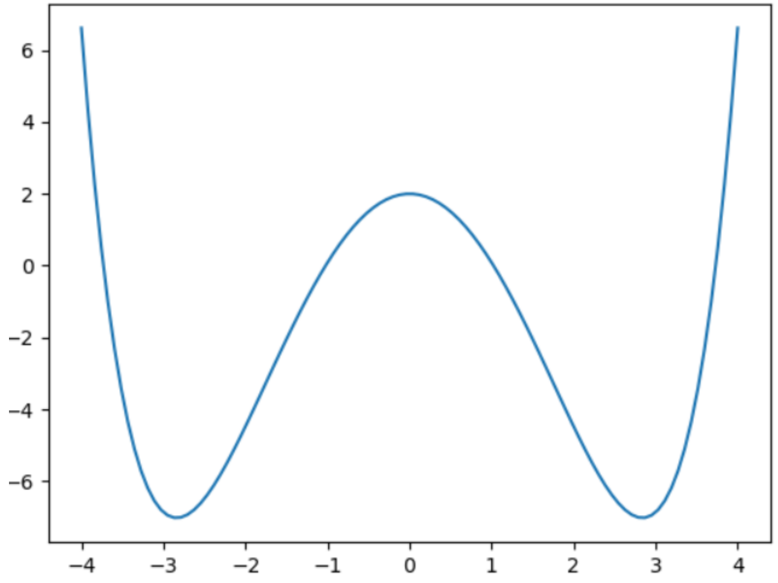
$f(x) = -x + e^{-x}$ has no minimizer

Local minimizer

- x^* is a local minimizer for f: $R^n \rightarrow R$ if $f(x^*) \leq f(x)$ for $||x^*-x|| \leq \varepsilon$ ($\varepsilon > 0$ is a constant)
- x^* is a global minimizer for f: $R^n \rightarrow R$ if $f(x^*) \leq f(x)$ for all $x \in R^n$



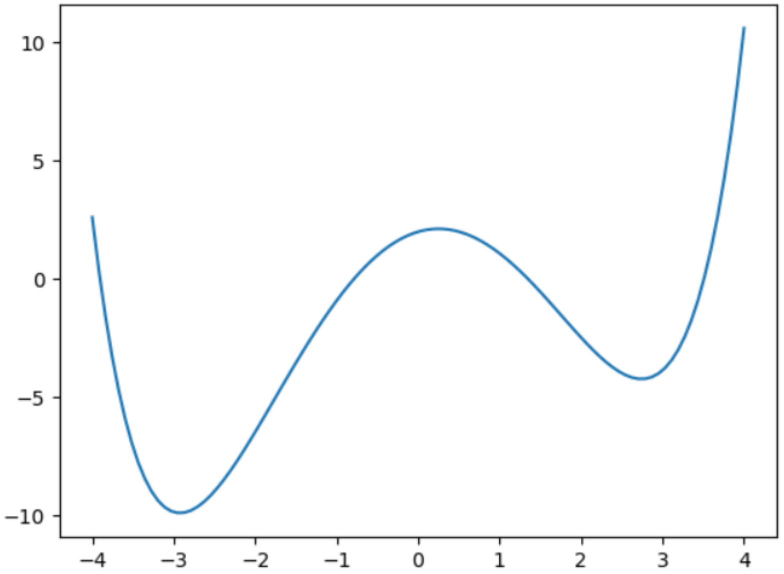
$f(x) = e^x + e^{-x} - 3x^2 + x$ has two local minimizers and one global minimizer



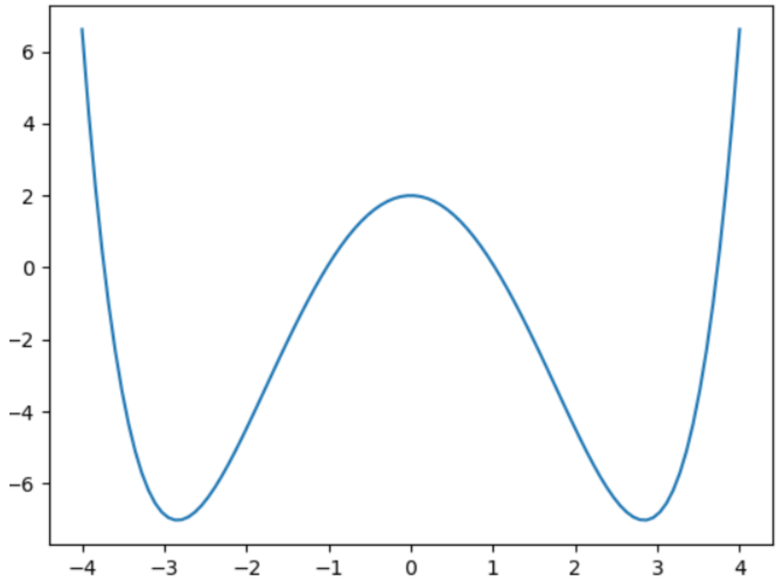
$f(x) = e^x + e^{-x} - 3x^2$ has two global minimizers

Local minimizer

- x^* is a local minimizer for $f: R^n \rightarrow R$ if $f(x^*) \le f(x)$ for $||x^*-x|| \le \varepsilon$ ($\varepsilon > 0$ is a constant)
- **Theorem** (Necessary condition for local minimum) If x^* is a local minimizer for $f: R^n \rightarrow R$, then $\nabla f(x^*) = 0$ (x^* is also called *stationary point* for f)



$f(x) = e^x + e^{-x} - 3x^2 + x$ has two local minimizers and one global minimizer



$f(x) = e^x + e^{-x} - 3x^2$ has two global minimizers

Example

- $f(x, y) = x^2 + y^2 - 2xy + x$
- $\nabla f(x, y) = \begin{pmatrix} 2x - 2y + 1 \\ 2y - 2x \end{pmatrix} = 0$ has no solution

→ there is no minimizer of $f(x, y)$

Local minimizer

- **Theorem** (Necessary condition for local minimum) If x^* is a local minimizer for f: $R^n \rightarrow R$, then $\nabla f(x^*) = 0$ (x^* is also called *stationary point* for f)
- **Theorem** (Sufficient condition for a local minimum) Assume x^* is a stationary point and that $\nabla^2 f(x^*)$ is positive definite, then x^* is a local minimizer
- $\nabla^2 f(x) =$

$$\begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \cdots & & & \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{pmatrix}$$

Local minimizer

- Matrix A_{nxn} is called positive definite if

$$A^i = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,i} \\ a_{2,1} & a_{2,2} & \ldots & a_{2,i} \\ & \ldots\ldots & & \\ a_{i,1} \ldots & a_{i,2} & \ldots & a_{i,i} \end{pmatrix}, \quad \det(A^i) > 0,\ i = 1,\ldots,n$$

Examples

Example $f(x,y) = e^{x^2+y^2}$

$$\nabla f(x) = \begin{pmatrix} 2xe^{x^2+y^2} \\ 2ye^{x^2+y^2} \end{pmatrix} = 0 \text{ has unique solution } x^* = (0,0)$$

$$\nabla^2 f(x^*) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} > 0 \rightarrow (0,0) \text{ is a minimizer of f}$$

Examples

- **Example** $f(x,y) = x^2 + y^2 - 2xy - x$

$$\nabla f(x) = \begin{pmatrix} -2x + 2y + 1 \\ -2x - 2y \end{pmatrix} = 0$$

has unique solution $x^* = (-1/4, 1/4)$

$$\nabla^2 f(x^*) = \begin{pmatrix} -2 & 2 \\ -2 & -2 \end{pmatrix}$$ is not positive definite

→ cannot conclude x^*

Descent method

Determine starting point $x^{(0)} \in R^n$;

$k \leftarrow 0$;

While (stop condition not reach){

Determine a search direction $p_k \in R^n$;

Determine a step size $\alpha_k > 0$ s.t. $f(x^{(k)} + \alpha_k p_k) < f(x^{(k)})$;

$x^{(k+1)} \leftarrow x^{(k)} + \alpha_k p_k$;

$k \leftarrow k+1$;

}

Stop condition may be

- $||\nabla f(x^k)|| \leq \varepsilon$
- $||x^{k+1} - x^k|| \leq \varepsilon$
- $k > K$ (maximum number of iterations)

Gradient descent method

- Gradient descent schema

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

```
init x^(0);
k = 1;
while stop condition not reach {
        specify constant α_k;
        x^(k) = x^(k-1) - α_k ∇f(x^(k-1));
        k = k + 1;
}
```

- α_k might be specified in such a way that $f(x^{(k-1)} - \alpha_k \nabla f(x^{(k-1)}))$ is minimized: $\frac{\partial f}{\partial \alpha_k} = 0$

Minimize $f(x) = x^4 - 2x^3 - 64x^2 + 2x + 63$

```python
# compute the function value
def f(x):
    return x**4 - 2*x**3 - 64*x**2 + 2*x + 63

# compute the derivative value
def grad(x):
    return 4*x**3 - 6*x**2 - 128 * x + 2

# Gradient descent algorithm with given alpha and initial point
def myGD(alpha, x0):
    x = [x0]
    # loop to evaluate a series of candidate
    for it in range(100000):
        # x[k+1] = x[k] - alpha * f'(x[k])
        x_new = x[-1] - alpha*grad(x[-1])

        # check stop condition (f'(x[k+1]) <= epsilon)
        if abs(grad(x_new)) < 1e-3:
            break

        # append x[k+1] into list
        x.append(x_new)

    # return a list of evaluated candidates and the iteration at which the algorithm stops
    return (x, it)
```

Minimize $f(x) = x^4 + 3x^2 - 10x + 4$

```python
def grad(x):
    return 4*x**3+ 6*x - 10

def f(x):
    return x**4 + 3* x**2  - 10 * x + 4

def myGD(alpha, x0):
    x = [x0]
    for it in range(1000):
        x_new = x[-1] - alpha*grad(x[-1])
#         if abs(grad(x_new)) < 1e-3:
#             break

        if(abs(x[-1] - x_new) < 1e-3):
            break

        x.append(x_new)
    return (x, it)
```

Minimize $f(x) = x^2 + 5sin(x)$

```
def grad(x):
    return 2*x+ 5*np.cos(x)

def f(x):
    return x**2 + 5*np.sin(x)

def myGD(delta, x0):
    x = [x0]
    for it in range(100):
        x_new = x[-1] - delta*grad(x[-1])
        if abs(grad(x_new)) < 1e-3:
            break
        x.append(x_new)
    return (x, it)
```

Minimize $f(x, y) = x^2 + y^2 + xy - x - y$

```
def grad(x, y):
    return (2*x + y - 1, 2*y + x - 1)

def f(x, y):
    return x**2 + y**2 + x*y - x - y

def myGD(delta, x0, y0):
    X = [(x0, y0)]
    for it in range(1000):
        x_new = X[-1][0] - delta*grad(X[-1][0], X[-1][1])[0]
        y_new = X[-1][1] - delta*grad(X[-1][0], X[-1][1])[1]
        if abs(grad(x_new, y_new)[0]) < 1e-6 and abs(grad(x_new, y_new)[1]) < 1e-6:
            break
        X.append((x_new, y_new))
    return (X, it)
```

Minimize $f(x) = x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 + x_1 + x_3$

- α_k might be specified in such a way that $f(x^{(k-1)} - \alpha_k \nabla f(x^{(k-1)}))$ is minimized: $\frac{\partial f}{\partial \alpha_k} = 0$

```
def grad(x1, x2, x3):
    return [2*x1 + 1 - x2, -x1 + 2*x2 - x3, -x2 + 2*x3 + 1]

def f(x1, x2, x3):
    return x1**2 + x2**2 + x3**2 - x1*x2 - x2*x3 + x1 + x3

def myGD(v1, v2, v3):
    x1 = v1
    x2 = v2
    x3 = v3
    for it in range(1000):
        print(f(x1, x2, x3))
        [D1,D2,D3] = grad(x1,x2,x3)
        A = 2*x1*D1 + 2*x2*D2 + 2*x3*D3 - x1*D2 - x2*D1 - x2*D3 -x3*D2 + D1 + D3
        B = 2*D1*D1 + 2*D2*D2 + 2*D3*D3 -2*D1*D2 - 2*D2*D3
        if B == 0:
            break
        alpha = A/B

        x1 = x1 - alpha*D1
        x2 = x2 - alpha*D2
        x3 = x3 - alpha*D3

        val = grad(x1, x2, x3)
        if (val[0]**2 + val[1]**2 + val[2]**2) < 1e-6:
            break

        X.append([x1, x2, x3])
    return (X, it)
```

Newton method

- Second-order Taylor approximation *g* of *f* at *x* is

$$f(x+h) \approx g(x+h) = f(x) + h\,\nabla f(x) + \frac{1}{2}h^2\ \nabla^2 f(x)$$

- Which is a convex quadratic function of *h*
- g(x+h) is minimized when $\frac{\partial g}{\partial h} = 0 \rightarrow h = -\nabla^2 f(x)^{-1}\,\nabla f(x)$

```
Generate x^(0); // starting point
k = 0;
while stop condition not reach{
   x^(k+1) ← x^(k) - ∇²f(x^(k))^-1 ∇f(x^(k));
   k = k + 1;
}
```

Minimize $f(x) = x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 + x_1 + x_3$

```
import numpy as np

def newton(f,df,Hf,x0):
    x = x0
    for i in range(10):
        iH = np.linalg.inv(Hf(x))
        D = np.array(df(x)).T #transpose matrix: convert from list to
                                  #column vector
        print('df = ',D)
        y = iH.dot(D) #multiply two matrices
        if np.linalg.norm(y) == 0:
            break
        x = x - y
        print('Step ',i,': ',x,' f  = ',f(x))
```

Minimize $f(x) = x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 + x_1 + x_3$

```
def main():
    print('main start....')
    f = lambda x: x[0] ** 2 + x[1] ** 2 + x[2] ** 2 - x[0] * x[1] - x[1] *
                    x[2] + x[0] + x[2]  # function f to be minimized
    df = lambda x: [2 * x[0] + 1 - x[1], -x[0] + 2 * x[1] - x[2], -x[1] + 2
                       * x[2] + 1]  # gradient
    Hf = lambda x: [[2,-1,0],[-1,2,-1],[0,-1,2]]# Hessian
    x0 = np.array([0,0,0]).T
    newton(f,df,Hf,x0)

if __name__ == '__main__':
    main()
```

Outline

1. Definitions
2. Unconstrained Optimization
3. **Constrained Optimization**
    - General constrained optimization problem
    - Lagrange multiplier method

General constrained optimization problem

- Optimization problem in the standard form

(*P*)	$minimize\ f(x)$
s.t.	$g_{i(x)} = 0, \quad \forall i \in \{1, 2, \dots, m\}$
	$x \in X \subseteq R^n$

with $x \in R^n$, and assume $D = (\cap_{i=1}^{m} \boldsymbol{dom}\ g_i)$ is not empty.

- Denote f^* the optimal value of $f(x)$
- If $f, g_i\ (i = 1,2, \dots, m)$ are convex functions.

Lagrangian function

- Optimization problem in the standard form

$$
\begin{array}{ll}
(P) & minimize\ f(x) \\
\text{s.t.} & g_{i(x)} = 0, \quad \forall i \in \{1, 2, \dots, m\} \\
 & x \in X \subseteq R^n
\end{array}
$$

- Langragian function of the above problem is defined as follows, $L: R^n \times R^m \rightarrow R$

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \times g_i(x)$$

Lagrangian function - properties

- Optimization problem in the standard form

(P) $\quad minimize\ f(x)$

s.t. $\quad g_{i(x)} = 0, \quad \forall i \in \{1, 2, \dots, m\}$

$\quad x \in X \subseteq R^n$

- Langragian function of the above problem is defined as follows, $L: R^n \times R^m \rightarrow R$

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \times g_i(x)$$

- **Theorem**: The optimal value of the optimization problem is the following property: $\nabla x, \lambda\ L(x, \lambda) = 0$

Lagrangian multiplier method

- **The method of Lagrange multipliers** is a technique in mathematics to find the local maxima or minima of a function $f(x_1, x_2, \dots, x_n)$ subject to constraints $g_i(x_1, \dots, x_n) = 0, \forall i \in \{1, \dots, m\}$.
- **Method**:
  - Step 1: Solving the following system

$$\nabla f(x_1, \dots, x_n) = \sum_{i=1}^{m} \lambda_i g_i(x_1, \dots, x_n)$$
$$g_1(x_1, \dots, x_n) = 0$$
$$\dots$$
$$g_m(x_1, \dots, x_n) = 0$$

  - Step 2: we get particular values of $x_1, x_2, \dots, x_n$, which can be plugged in $f(x_1, x_2, \dots, x_n)$ to get the extremum value if it exists.

Example 1:

- **Problem**: Find the maximum and minimum of $f(x,y) = 5x - 3y$ subject to the constraint $x^2 + y^2 = 136$.
- **Solution**:
  - Region of possible solutions lies on a disk of radius $\sqrt{136}$ which is a closed and bounded region, $-\sqrt{136} \le x, y \le \sqrt{136}$
  - The Lagrangian function $L(x,y,\lambda) = 5x - 3y - \lambda(x^2 + y^2 - 136)$
  - As the result of the theorem, we have

$$\begin{aligned} 5 &= 2\lambda x \\ -3 &= 2\lambda y \\ x^2 + y^2 &= 136 \end{aligned}$$

  - Notice that, we can't have $\lambda = 0$ since that would not satisfy the first two equations. So, since we know that $\lambda \neq 0$ we can solve the first two equations for x and y respectively. This gives,

$$x = \frac{5}{2\lambda} \qquad\qquad y = -\frac{3}{2\lambda}$$

Example 1

- **Problem**: Find the maximum and minimum of $f(x, y) = 5x - 3y$ subject to the constraint $x^2 + y^2 = 136$.
- **Solution**:
  - Plugging these into the constraint gives

$$\frac{25}{4\lambda^2} + \frac{9}{4\lambda^2} = \frac{17}{2\lambda^2} = 136$$

  - We can solve this for λ

$$\lambda^2 = \frac{1}{16} \qquad \Rightarrow \qquad \lambda = \pm\frac{1}{4}$$

  - Now, that we know λ we can find the points that will be potential maximums and/or minimums.
  - If $\lambda = -\frac{1}{4}$ we get $x = -10, y = 6$
  - if $\lambda = \frac{1}{4}$ we get $x = 10, y = -6$
  - So,

$$f(-10, 6) = -68 \qquad \text{Minimum at } (-10, 6)$$
$$f(10, -6) = 68 \qquad \text{Maximum at } (10, -6)$$

Example 2:

- **Problem**:
  - Objective function: $maximize\ f(x,y) = xy$
  - Constraint: $g(x,y) = 10x + 20y - 400 = 0$
- **Solution**
  - Form the Lagrange function:

$$L(x,y,\mu) = f(x,y) - \mu\big(g(x,y)\big)$$
$$L(x,y,\mu) = xy - \mu(10x + 20y - 400)$$

  - Set each first order partial derivative equal to zero:

$$\frac{\partial L}{\partial x} = y - 10\mu = 0$$

$$\frac{\partial L}{\partial y} = x - 20\mu = 0$$

$$\frac{\partial L}{\partial \mu} = -(10x + 20y - 400) = 0$$

Example 2:

- **Problem**:
  - Objective function: $maximize\ u(x, y) = xy$
  - Constraint: $g(x, y) = 10x + 20 - 400 = 0$
- **Solution**:
  - Set each first order partial derivative equal to zero:

$$\frac{\partial L}{\partial x} = y - 10\mu = 0$$

$$\frac{\partial L}{\partial y} = x - 20\mu = 0$$

$$\frac{\partial L}{\partial \mu} = -(10x + 20y - 400) = 0$$

  - So,

$$10x + 20y = 400$$

$$40y = 400$$

$$\mathbf{y = 10}$$

$$\mathbf{x = 2y = 20}$$

Example 3:

- **Problem:**
  - Objective function: $maximize\ f(x, y) = x + y$
  - Constraint: $g(x, y) = x^2 + y^2\ - 2 = 0$
- **Solution:**
  - Form the Lagrange function:

$$\mathcal{L}(x, y, \lambda) = x + y + \lambda(x^2 + y^2 - 2)$$

  - Set each first order partial derivative equal to zero:
    - $\frac{\partial L}{\partial x} = 1 + 2\lambda x = 0$
    - $\frac{\partial L}{\partial y} = 1 + 2\lambda y = 0$
    - $\frac{\partial L}{\partial \lambda} = x^2 + y^2 - 2 = 0$

Example 3

- **Problem:**
  - Objective function: $maximize\ f(x, y) = x + y$
  - Constraint: $g(x, y) = x^2 + y^2 - 2 = 0$
- **Solution:**
  - Set each first order partial derivative equal to zero:
    - $\frac{\partial L}{\partial x} = 1 + 2\lambda x = 0$
    - $\frac{\partial L}{\partial y} = 1 + 2\lambda y = 0$
    - $\frac{\partial L}{\partial \lambda} = x^2 + y^2 - 2 = 0$
  - We have , $(x, y) \in \{(1,1), (-1, -1)\}$
  - So, $(x, y) = (1,1)$

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !