Fundamentals of Optimization

Gradient descent method for unconstrained optimization

Pham Quang Dung

dungpq@soict.hust.edu.vn

Department of Computer Science

Content

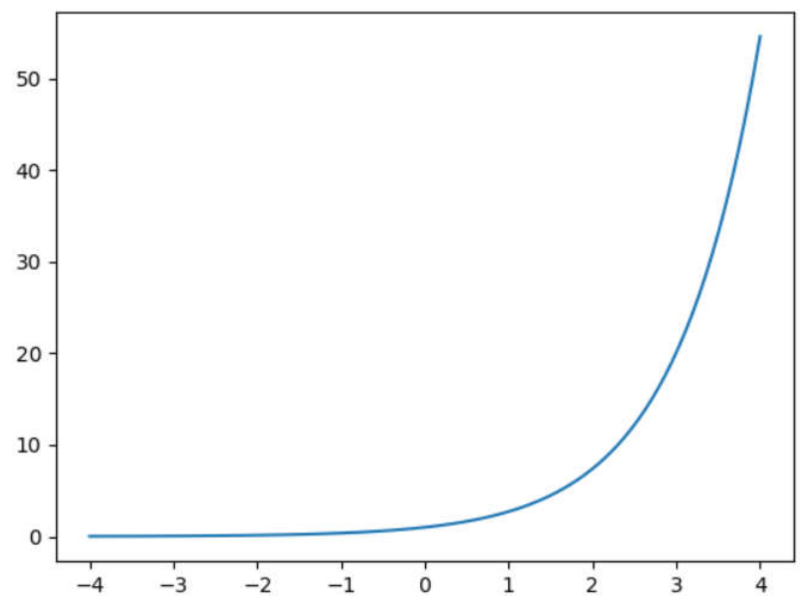
- Unconstrained optimization problems
- Descent method
- Gradient descent method
- Newton method

Gradient descent method

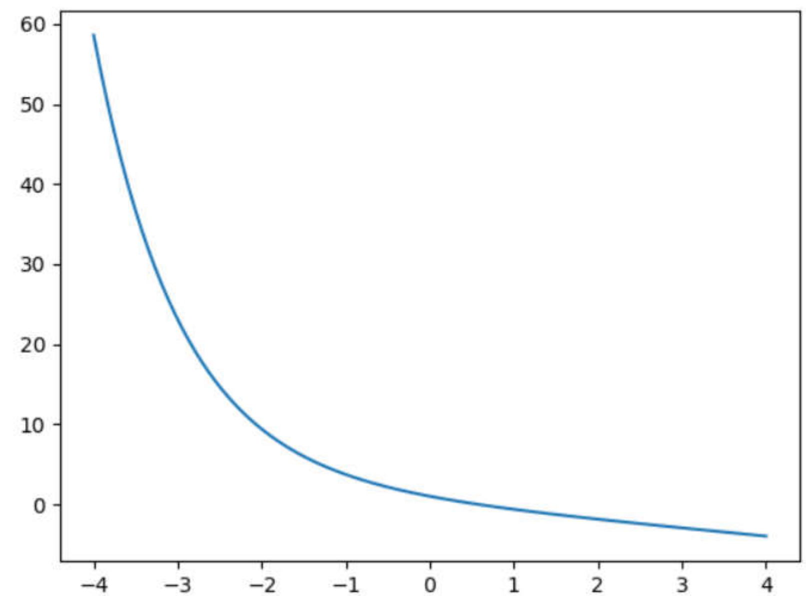
- For unconstrained, smooth convex optimization problem: min $f(x)$
  - f: $R^n \rightarrow R$ is convex and twice differentiable
  - ***dom*** $f = R$: no constraint
  - Assumption: the problem is solvable with $f^* = \min_x f(x)$ and $x^* = \text{argMin}_x\, f(x)$
- To find x, solve equation $\nabla f(x^*) = 0$: 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: $f(x^{(k)}) - f^* \leq \varepsilon$

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 $x \in R^n$



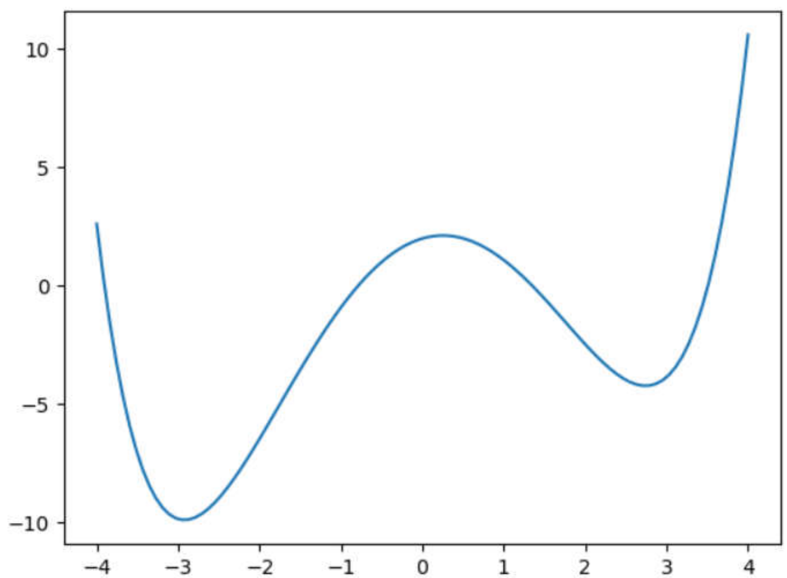
$f(x) = e^x$ has no minimizer



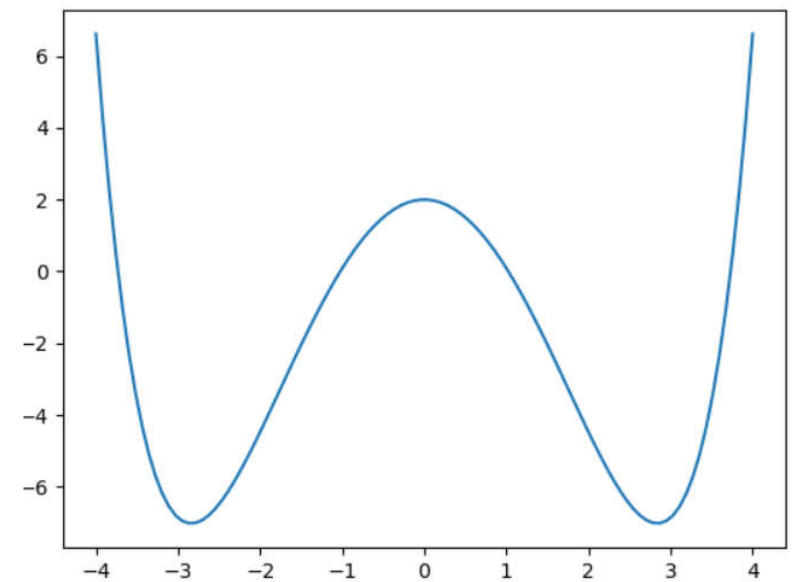
$f(x) = -x + e^{-x}$ has no minimizer

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 one local minimizer and one global minimizer



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

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)
- **Proof** Suppose that x^* is a local minimizer but $\nabla f(x^*) \neq 0$. We can find a vector z such that $< \nabla f(x^*), z> < 0$ (for instance $z = -\nabla f(x^*)$, $< \nabla f(x^*), z> = -\|\nabla f(x^*)\|^2 < 0$)
- For a constant $t \geq 0$, consider vector $x(t) = x^* + tz$, and $\varphi(t) = f(x(t))$
- $\frac{d\varphi(t)}{dt}\Big|_{t=0} = \frac{df(x(t))}{dt}\Big|_{t=0} = < \nabla f(x^*), z> < 0 \rightarrow$ with constant t small enough, $\varphi(t) < \varphi(0)$ or $f(x(t)) < f(x^*)$ (conflict with the assumption that x^* is a local minimizer) □

Local minimizer

Example

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

→ there is no minimizer of $f(x,y)$

Local minimizer

- **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

$$\nabla^2 f(x) = \begin{pmatrix} \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_1 \partial xn} \\ \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f(x)}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial x_2 \partial xn} \\ \cdots & & & \\ \dfrac{\partial^2 f(x)}{\partial x_n \partial x_1} & \dfrac{\partial^2 f(x)}{\partial xn \partial x_2} & \cdots & \dfrac{\partial^2 f(x)}{\partial xn \partial xn} \end{pmatrix}$$

Local minimizer

- Matrix A_{nxn} is called positive definite if

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

Local minimizer

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

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

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

Local minimizer

- **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

$\rightarrow$ 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

- $\| \nabla f(x^k) \| \le \varepsilon$
- $\|x^{k+1} - x^k\| \le \varepsilon$
- 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: $\frac{\partial f}{\partial \alpha_k} = 0$

Gradient descent method

Example

- $f(x_1,x_2,x_3) = x_1^2 + x_2^2 + x_3^2 - x_1x_2 - x_2x_3 + x_1 + x_3$
- $f(x_1,x_2,x_3) = [2x_1 - x_2 + 1, 2x_2 - x_1 - x_3, 2x_3 - x_2 + 1]^T$
- $\nabla^2 f = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$

Gradient descent method

```
import numpy as np
f = lambda x1,x2,x3: x1**2 + x2**2 + x3**2 - x1*x2 - x2*x3 + x1 + x3 # function f
df = lambda x1,x2,x3: [2*x1 + 1 - x2,-x1 + 2*x2 - x3,-x2 + 2*x3 + 1] # gradient
x1,x2,x3 = 0,0,0
for i in range(1000):
        [D1,D2,D3] = df(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
```

Gradient descent method

Step	x	$\alpha_k \nabla f(x^{(k-1)})$	f
Initialization	[0,0,0]	[0.5, 0.0, 0.5]	0
Step 1	[-0.5, 0.0, -0.5]	[0.0, 0.5, 0.0]	-0.5
Step 2	[-0.5, -0.5, -0.5]	[0.25, 0.0, 0.25]	-0.75
Step 3	[-0.75, -0.5, -0.75]	[0.0, 0.25, 0.0]	-0.875
. . .	. . .	. . .	. .
Step 107	[-1.0, -1.0, -1.0]		-1.0

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 $\frac{\partial g}{\partial h} = 0 \rightarrow h = -\nabla^2 f(x)^{-1}\,\nabla f(x)$

Newton method

Generate $x^{(0)}$; // starting point

$k = 0$;

while stop condition not reach{

$x^{(k+1)} \leftarrow x^{(k)} - \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)})$;

$k = k + 1$;

}

Newton method

```
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))
```

Newton method

```
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()
```

Newton method

Step	x	y	f
Initialization	[0,0,0]	[1, 1, 1]	0
Step 1	[-1., -1., -1.]	[-2.46519033e-32 1.11022302e-16 2.22044605e-16]	-1.0000000000000004
Step 2	[-1., -1., -1.]	[0., 0., 0.]	-1