南昌大学
NANCHANG UNIVERSITY

大数据优化：理论、算法及其应用

——来源于《最优化计算方法》（高教社）

★★★ 主讲人：彭振华 ★★★

联系方式：zhenhuapeng@ncu.edu.cn
zhenhuapeng@whu.edu.cn
15870605317（微信同号）

研究兴趣：1. 非凸非光滑优化算法与理论
2. 智能决策
3. 智能计算与机器学习

数学与计算机学院

2026年

课件资源：https://zhenhuapeng.github.io//coursematerials/

目录

南昌大学
NANCHANG UNIVERSITY

无约束优化算法

第 四 部分

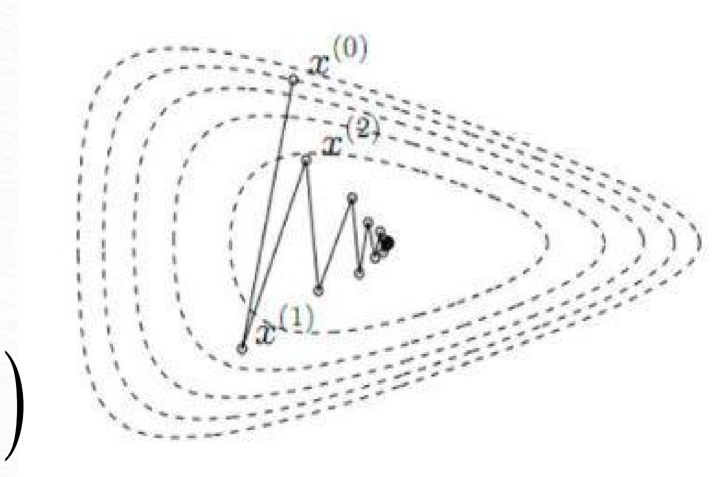
- 线搜索方法
- 梯度类算法
- 共轭梯度算法
- (拟)牛顿算法
- 信赖域算法与最小二乘

线搜索方法

$$\min_x \ f(x)$$

- $x^{k+1} = x^k + \alpha_k d^k$

步长　　搜索方向



$$f\left(x^{k+1}\right) = f\left(x^k + \alpha_k d^k\right) = f\left(x^k\right) + \alpha_k \nabla f\left(x^k\right)^T d^k + o\left(\alpha_k \left\| d^k \right\|\right)$$
$$< f\left(x^k\right)$$

- **下降方向d^k：$\nabla f(x^k)^T d^k < 0$.**

下降方向d^k的选择千差万别，但步长α_k的选取方式非常相似

精确线搜索算法　　非精确线搜索算法

线搜索方法

➢ 精确线搜索算法

$$\min_x \ f(x)$$

$$\alpha_k = \arg\min_{\alpha>0} \phi(\alpha) = f\left(x^k + \alpha d^k\right) \Longrightarrow \nabla f\left(x^k + \alpha_k d^k\right)^T d^k = 0$$

$$\min_x \frac{1}{2} x^T Q x + c^T x, \text{其中} Q \succ 0 \Longrightarrow \alpha_k = \frac{-\nabla f\left(x^k\right)^T d^k}{\left(d^k\right)^T Q d^k}$$

➢ 非精确线搜索算法

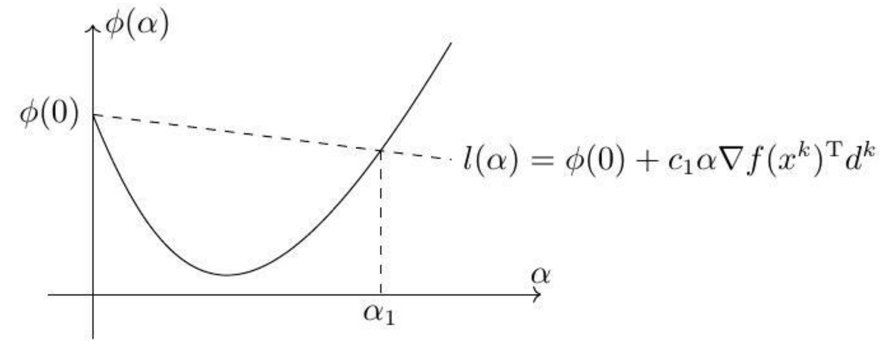
是不是任何步长最终都能收敛，只是快慢的问题？

$$\min_x x^2, x^0 = 1$$

$$\alpha_{k,1} = \frac{1}{3^{k+1}} \Longrightarrow x_{k,1} = \frac{1}{2}\left(1 + \frac{1}{3^k}\right)$$

$$\alpha_{k,2} = 1 + \frac{2}{3^{k+1}} \Longrightarrow x_{k,2} = \frac{(-1)^k}{2}\left(1 + \frac{1}{3^k}\right)$$

- **非精确线搜索算法**
- **Armijo 准则**



$$f\left(x^k+\alpha d^k\right)\le f\left(x^k\right)+c_1\alpha\nabla f\left(x^k\right)^T d^k, c_1\in(0,1)$$

参数 c_1 通常选为一个很小的正数, 例如 $c_1 = 10^{-3}$

回退法选取$\alpha_k = \gamma^{j0}\alpha$, 其中参数$\gamma \in (0, 1)$

Algorithm 1 线搜索回退法

1: 选择初始步长 $\hat{\alpha}$, 参数 $\gamma, c \in (0,1)$. 初始化 $\alpha \leftarrow \hat{\alpha}$.
2: **while** $f(x^k+\alpha d^k) > f(x^k)+c\alpha\nabla f(x^k)^{\mathrm{T}}d^k$ **do**
3: 令 $\alpha \leftarrow \gamma\alpha$.
4: **end while**
5: 输出 $\alpha_k = \alpha$.

实际应用中通常也会给α 设置一下界, 防止步长过小

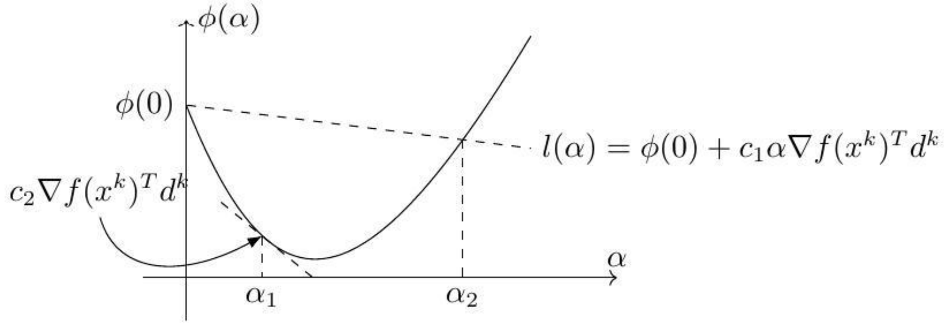
线搜索方法

```
import numpy as np
def armijo_line_search(f, grad_f, x, direction, alpha=1.0, beta=0.5, c1=1e-3, max_iter=100):
    fx = f(x)
    grad = grad_f(x)
    slope = np.dot(grad, direction)

    for _ in range(max_iter):
        candidate = x + alpha * direction
        if f(candidate) <= fx + c1 * alpha * slope:
            return alpha
        alpha *= beta
    return alpha

def h(t): return t**3 - 2*t + 1
def h_grad(t): return 3*t**2 - 2
t_min = 0
direction = -h_grad(t_min)
step_size = armijo_line_search(h, h_grad, t_min, direction)
print(f"最优步长：{step_size:.6f}")
```

- **非精确线搜索算法**



- **Wolfe 准则**

$$f\left(x^k+\alpha d^k\right) \le f\left(x^k\right) + c_1\alpha\nabla f\left(x^k\right)^T d^k, c_1 \in (0,1)$$

$$\nabla f\left(x^k+\alpha d^k\right)^T d^k \ge c_2\nabla f\left(x^k\right)^T d^k, c_2 \in (0,1), c_1 < c_2$$

- `if np.logical_and(f(candidate) <= fx + c1 * alpha * slope, np.dot(grad_f(candidate),direction) >= c2 * slope):`

- **非单调线搜索准则(Grippo)**

$$f\left(x^k+\alpha d^k\right) \le \max_{0\le j\le \min\{k,M\}} f\left(x^{k-j}\right) + c_1\alpha\nabla f\left(x^k\right)^T d^k, c_1 \in (0,1)$$

线搜索方法

```
import numpy as np
def grippo_line_search(f, grad_f, x, direction, m=10, alpha_init=1.0, beta=0.5, c=1e-3, max_iter=100):
    history = [f(x)]  # 函数值历史记录
    grad = grad_f(x)
    slope = np.dot(grad, direction)
    alpha = alpha_init
    for _ in range(max_iter):
        candidate = x + alpha * direction
        f_current = f(candidate)
        f_max = max(history[-min(m, len(history)):])
        if f_current <= f_max + c * alpha * slope:
            history.append(f_current)
            return alpha
        alpha *= beta
    return alpha
def quadratic(x):
    return x[0]**2 + 10*x[1]**2
def quadratic_grad(x):
    return np.array([2*x[0], 20*x[1]])
x0 = np.array([5.0, 1.0])
direction = -quadratic_grad(x0)
step_size = grippo_line_search(quadratic, quadratic_grad, x0, direction, m=5)
print(f"最优步长：{step_size:.6f}")
```

- **Zoutendijk定理**：考虑$x^{k+1}=x^k+\alpha_k d^k$，在迭代过程中Wolfe准则满足. 假设目标函数f下有界、连续可微且梯度L-利普希茨连续，则

Zoutendijk条件 $$\sum_{k=0}^{\infty}\cos^2\theta_k\left\|\nabla f\left(x^k\right)\right\|^2<+\infty,\text{其中}\cos\theta_k=\frac{-\nabla f\left(x^k\right)^T d^k}{\left\|\nabla f\left(x^k\right)\right\|\left\|d^k\right\|}$$

- **proof.**

Wolfe第二个条件 $$\nabla f\left(x^k+\alpha_k d^k\right)^T d^k\geq c_2\nabla f\left(x^k\right)^T d^k$$

$$\left(\nabla f\left(x^{k+1}\right)-\nabla f\left(x^k\right)\right)^T d^k\geq\left(c_2-1\right)\nabla f\left(x^k\right)^T d^k$$

梯度L-利普希茨 $$\left(\nabla f\left(x^{k+1}\right)-\nabla f\left(x^k\right)\right)^T d^k\leq\alpha_k L\left\|d^k\right\|^2$$

$$\alpha_k\geq\frac{c_2-1}{L}\frac{\nabla f\left(x^k\right)^T d^k}{\left\|d^k\right\|^2}$$

Wolfe第一个条件

$$f\left(x^k+\alpha_k d^k\right)\le f\left(x^k\right)+c_1\alpha_k\nabla f\left(x^k\right)^T d^k$$

$$\alpha_k \ge \frac{c_2-1}{L}\frac{\nabla f\left(x^k\right)^T d^k}{\left\|d^k\right\|^2}$$

$$f\left(x^{k+1}\right)\le f\left(x^k\right)+c_1\frac{c_2-1}{L}\frac{\left(\nabla f\left(x^k\right)^T d^k\right)^2}{\left\|d^k\right\|^2}$$

$$f\left(x^{k+1}\right)\le f\left(x^k\right)+c_1\frac{c_2-1}{L}\cos^2\theta_k\left\|\nabla f\left(x^k\right)\right\|^2$$

$$f\left(x^{k+1}\right)\le f\left(x^0\right)-c_1\frac{1-c_2}{L}\sum_{j=0}^{k}\cos^2\theta_j\left\|\nabla f\left(x^j\right)\right\|^2$$

$$\sum_{j=0}^{k}\cos^2\theta_j\left\|\nabla f\left(x^j\right)\right\|^2\le\frac{L\left(f\left(x^0\right)-f\left(x^{k+1}\right)\right)}{c_1\left(1-c_2\right)}$$

因为函数f是下有界的,得证!

- **（线搜索算法的收敛性）**假设对任意的k, 存在常数$\gamma > 0$, 使得

$$\theta_k < \pi/2 - \gamma$$

则在Zoutendijk定理成立的条件下, 有

$$\lim_{k\to\infty} \nabla f(x^k) = 0.$$

proof. 假设结论不成立, 即存在子列$\{k_l\}$ 和正常数$\delta > 0$, 使得

$$\|\nabla f(x^{k_l})\| \geq \delta$$

$\cos\theta_k > \sin\gamma > 0$

$$\sum_{k=0}^{\infty} \cos^2\theta_k \left\|\nabla f\left(x^k\right)\right\|^2 \geq \sum_{l=1}^{\infty} \cos^2\theta_{k_l} \left\|\nabla f\left(x^{k_l}\right)\right\|^2 \geq \sum_{l=1}^{\infty} \sin^2\gamma\,\delta^2 \to +\infty$$

梯度下降算法

$$\min_x \; f(x)$$

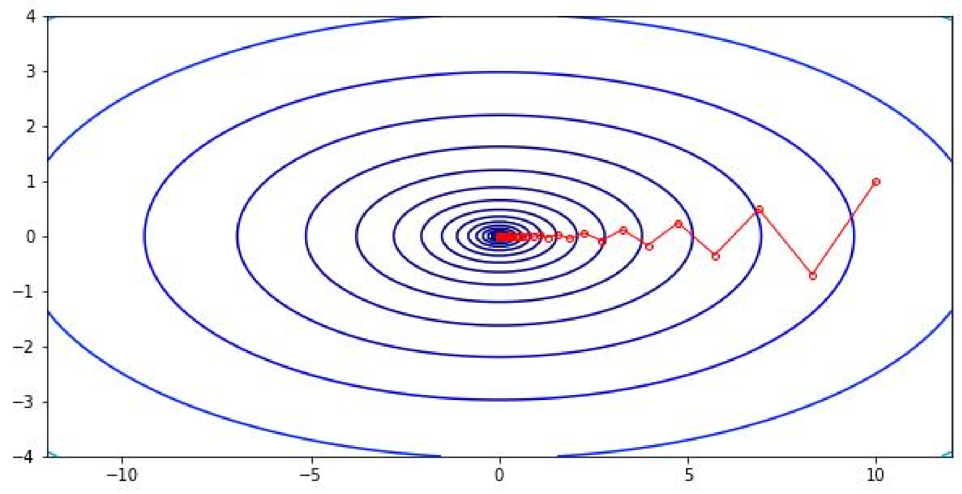
$$f\left(x^k+\alpha d^k\right)=f\left(x^k\right)+\alpha\nabla f\left(x^k\right)^T d^k+o\left(\alpha\left\|d^k\right\|\right)$$

$$\min_{d^k} f\left(x^k\right)+\alpha\nabla f\left(x^k\right)^T d^k+o\left(\alpha\left\|d^k\right\|\right)$$

$$s.t.\ \left\|d^k\right\|=1.$$

- **当α足够小时，$d^k = -\nabla f(x^k)$**
- **梯度下降算法**

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



- **例题 $\min_{x,y}\ x^2 + 10y^2$　$\alpha_k = 0.085$　初始点$(x^0, y^0) = (10, 1)$**
- **练习：$\min_{x,y}\ (x^2 + y^2) / 2$　$\alpha_k = 1$　初始点$(x^0, y^0) = (1, 1)$**

梯度下降算法

```python
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x[0]**2 + 10*x[1]**2
def grad_f(x):
    return np.array([2*x[0], 20*x[1]])

def gradient_descent(x, alpha, iter = 300, tol = 1e-6):
    x_history = [x]
    k = 0
    while k < iter:
        x = x - alpha * grad_f(x)
        x_history.append(x)
        k = k + 1
        if np.linalg.norm(grad_f(x)) < tol:
            break
    return x_history, k
```

```python
x0 = np.array([10, 1])
alpha = 0.085
gd_history, k = gradient_descent(x0, alpha)

x = np.linspace(-12, 12, 100)
y = np.linspace(-4, 4, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + 10*Y**2  # 定义目标函数
# 绘制等高线图
plt.figure(figsize=(10, 5))
plt.contour(X, Y, Z, levels=np.logspace(-2, 3, 20), cmap='jet')

plt.plot([x[0] for x in gd_history], [x[1] for x in gd_history],
marker='o',markersize=4, linewidth=1, color = 'red',
markerfacecolor='none', label='Gradient Descent')

plt.show()
```

梯度下降算法

➢ 精确线搜索方法

练习：$\min_x 2x_1^2+2x_2^2+2x_1x_2-4x_1-6x_2$，初始点(1,1)

$$\min_x \ f(x)$$

$$\alpha_k=\arg\min_{\alpha>0}\phi(\alpha)=f\left(x^k+\alpha d^k\right) \longrightarrow \nabla f\left(x^k+\alpha_k d^k\right)^T d^k=0$$

$$\min_x \frac{1}{2}x^TQx+c^Tx, \text{其中} Q\succ 0 \longrightarrow \alpha_k=\frac{-\nabla f\left(x^k\right)^T d^k}{\left(d^k\right)^T Qd^k}$$

$d^k = -\nabla f(x^k)$

$$\alpha_k=\frac{\left\|\nabla f\left(x^k\right)\right\|^2}{\nabla f\left(x^k\right)^T Q\nabla f\left(x^k\right)}$$

对于正定二次函数，基于精确线搜索方法的梯度下降算法关于迭代点列$\{x^k\}$是Q-线性收敛的

$$\left\|x^{k+1}-x^*\right\|_A^2\le\left(\frac{\lambda_{\max}-\lambda_{\min}}{\lambda_{\max}+\lambda_{\min}}\right)^2\left\|x^k-x^*\right\|_A^2$$

梯度下降算法

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x[0]**2 + 10*x[1]**2
def grad_f(x):
    return np.array([2*x[0], 20*x[1]])

def step_exact(x):
    return np.linalg.norm(grad_f(x)) ** 2 /
np.dot(np.dot(np.array([[2, 0],[0,
20]]),grad_f(x)),grad_f(x))

def gradient_descent(x, iter = 300, tol = 1e-6):
    x_history = [x]
    k = 0
    while k < iter:
        alpha = step_exact(x)
        x = x - alpha * grad_f(x)
        x_history.append(x)
        k = k + 1
        if np.linalg.norm(grad_f(x)) < tol:
            break
    return x_history, k
```

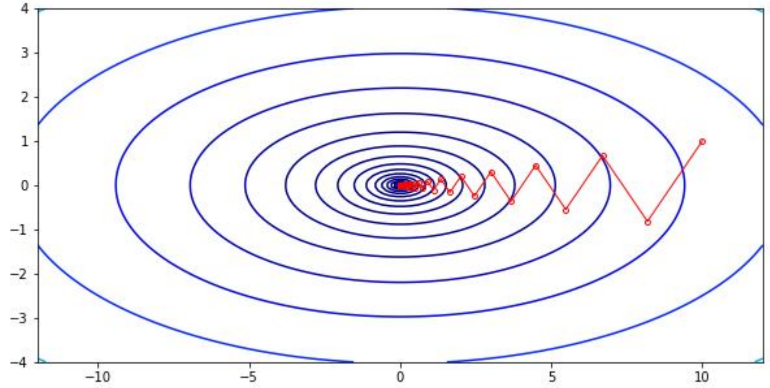
```
x0 = np.array([10, 1])
gd_history, k = gradient_descent(x0)
print("最优解为", gd_history[-1])
print("梯度范数值为", np.linalg.norm(grad_f(gd_history[-1])))

x = np.linspace(-12, 12, 100)
y = np.linspace(-4, 4, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + 10*Y**2  # 定义目标函数
# 绘制等高线图
plt.figure(figsize=(10, 5))
plt.contour(X, Y, Z, levels=np.logspace(-2, 3, 20), cmap='jet')

plt.plot([x[0] for x in gd_history], [x[1] for x in gd_history],
marker='o',markersize=4, linewidth=1, color = 'red', markerfacecolor='none',
label='Gradient Descent')

plt.show()
```

```
最优解为 [3.19952043e-07 3.19952043e-08]
梯度范数值为 9.049610381772918e-07
```



梯度下降算法

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

- 设函数 $f(x)$ 为凸的梯度L - 利普希茨连续函数
- 极小值$f^* = f(x^*) = \inf_x f(x)$ 存在且可达.
- 如果步长α_k 取为常数α 且满足$0 < \alpha \leq 1/L$

结论：点列$\{x^k\}$的函数值收敛到最优值, 且在函数值的意义下收敛速度为$O(1/k)$.

梯度下降算法

➢ **proof.** $x^{k+1} = x^k - \alpha_k \nabla f(x^k)$

二次上界性

$$f(y) \le f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|^2, \forall x, y \in dom\ f.$$

$$f\left(x-\alpha\nabla f\left(x\right)\right) \le f(x) - \alpha\left(1-\frac{L\alpha}{2}\right)\|\nabla f(x)\|^2.$$

0<α<$1/L$

$$f\left(x-\alpha\nabla f\left(x\right)\right) \le f(x) - \frac{\alpha}{2}\|\nabla f(x)\|^2$$

f是凸函数

$$\le f^* + \nabla f\left(x\right)^T\left(x-x^*\right) - \frac{\alpha}{2}\|\nabla f(x)\|^2$$

$$= f^* + \frac{1}{2\alpha}\left(\|x-x^*\|^2 - \|x-\alpha\nabla f(x)-x^*\|^2\right)$$

$$f\left(x^i\right) - f^* \le \frac{1}{2\alpha}\left(\|x^{i-1}-x^*\|^2 - \|x^i-x^*\|^2\right)$$

梯度下降算法

$$f\left(x^i\right)-f^* \le \frac{1}{2\alpha}\left(\left\|x^{i-1}-x^*\right\|^2-\left\|x^i-x^*\right\|^2\right)$$

$$\sum_{i=1}^{k} f\left(x^i\right)-f^* \le \frac{1}{2\alpha}\sum_{i=1}^{k}\left(\left\|x^{i-1}-x^*\right\|^2-\left\|x^i-x^*\right\|^2\right)$$

$$=\frac{1}{2\alpha}\left(\left\|x^{0}-x^*\right\|^2-\left\|x^k-x^*\right\|^2\right)$$

$$\le \frac{1}{2\alpha}\left\|x^{0}-x^*\right\|^2$$

$f(x^i)$ 是非增的

$$f\left(x^k\right)-f^* \le \frac{1}{k}\sum_{i=1}^{k} f\left(x^i\right)-f^* \le \frac{1}{2k\alpha}\left\|x^{0}-x^*\right\|^2$$

梯度下降算法

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

- 设函数 $f(x)$ 为凸的梯度L - 利普希茨连续函数
- 极小值$f^* = f(x^*) = \inf_x f(x)$ 存在且可达.
- 如果步长α_k 取为Armijo步长，$\alpha_k \geq \alpha_{min} := \min\{1, \gamma / L\}$

结论：点列$\{x^k\}$的函数值收敛到最优值, 且在函数值的意义下收敛速度为$O(1/k)$.

梯度下降算法

$$f\left(x^k-\alpha_i\nabla f\left(x^k\right)\right)\le f\left(x^k\right)-c_1\alpha_i\left\|\nabla f\left(x^k\right)\right\|^2$$

Armijo准则，α=0.5

$$f\left(x-\alpha\nabla f(x)\right)\le f(x)-\frac{\alpha}{2}\left\|\nabla f(x)\right\|^2$$

$$f\left(x^i\right)-f^*\le\frac{1}{2\alpha_i}\left(\left\|x^{i-1}-x^*\right\|^2-\left\|x^i-x^*\right\|^2\right)$$

$$\sum_{i=1}^{k}f\left(x^i\right)-f^*\le\frac{1}{2\alpha_{\min}}\sum_{i=1}^{k}\left(\left\|x^{i-1}-x^*\right\|^2-\left\|x^i-x^*\right\|^2\right)$$

$$=\frac{1}{2\alpha_{\min}}\left(\left\|x^0-x^*\right\|^2-\left\|x^k-x^*\right\|^2\right)$$

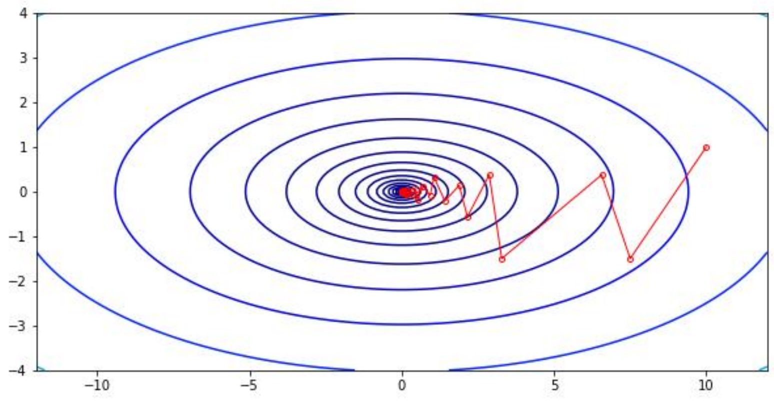
$$\le\frac{1}{2\alpha_{\min}}\left\|x^0-x^*\right\|^2$$

$f(x^i)$ 是非增的

$$f\left(x^k\right)-f^*\le\frac{1}{k}\sum_{i=1}^{k}f\left(x^i\right)-f^*\le\frac{1}{2k\alpha_{\min}}\left\|x^0-x^*\right\|^2$$

梯度下降算法

```
➢ def armijo_line_search(f, grad_f, x, direction, alpha=0.5, beta=0.5, c1=1e-3, max_iter=100):
➢     fx = f(x)
➢     grad = grad_f(x)
➢     slope = np.dot(grad, direction)
➢     for _ in range(max_iter):
➢         candidate = x + alpha * direction
➢         if np.logical_and(f(candidate) <= fx + c1 * alpha * slope,alpha >= 1/40):
➢             return alpha
➢         alpha *= beta
➢         if alpha < 1/40:
➢             alpha = 1/40
➢             break
➢     return alpha
```




```
➢ alpha = armijo_line_search(f, grad_f, x, -grad_f(x))
```

```
最优解为 [ 2.79941447e-07 -3.67867770e-08]
梯度范数值为 9.245407719211133e-07
```

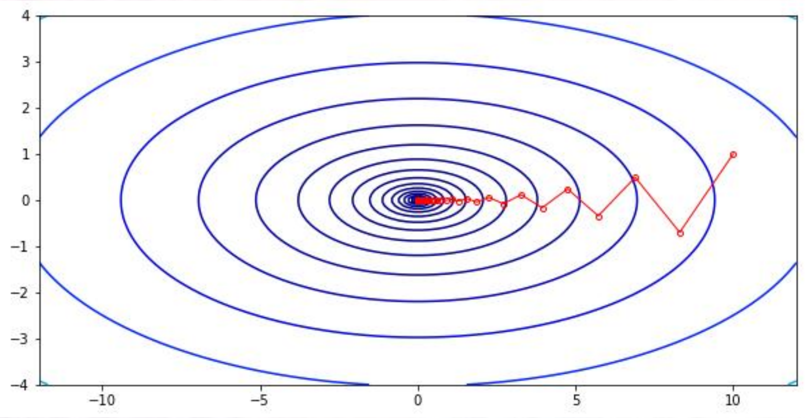
梯度下降算法

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

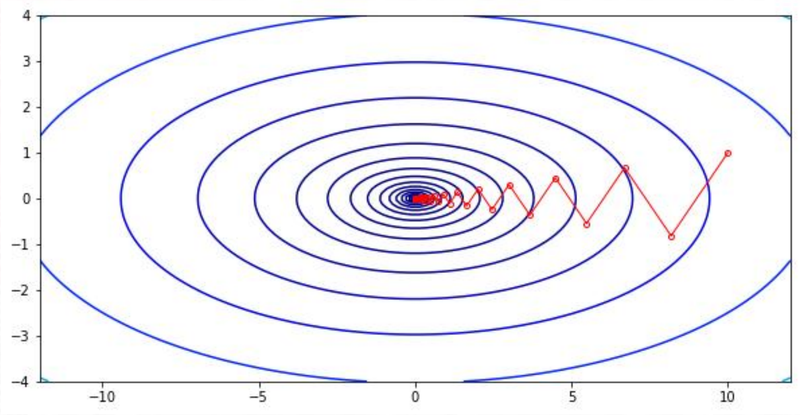
- 设函数 $f(x)$ 为m-强凸的梯度L - 利普希茨连续函数
- 极小值$f^* = f(x^*) = \inf_x f(x)$ 存在且可达.
- 如果步长α_k 取为常数α 且满足$0 < \alpha \leq 2/(m+L)$

结论：点列$\{x^k\}$收敛到x^*, 且收敛速度为Q-线性收敛.

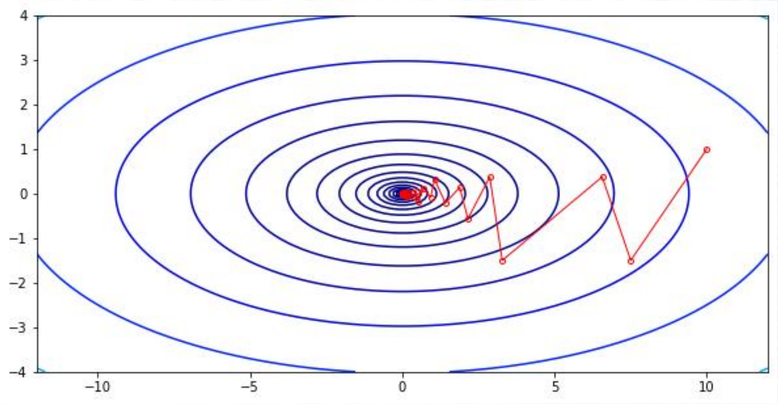
梯度下降算法



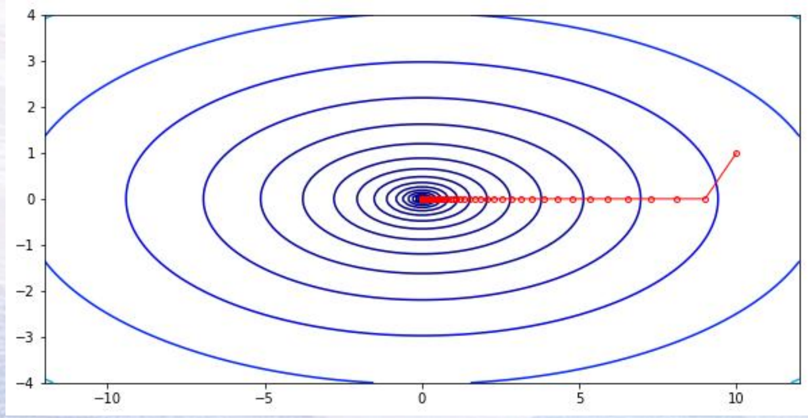
$\alpha_k = 0.085$



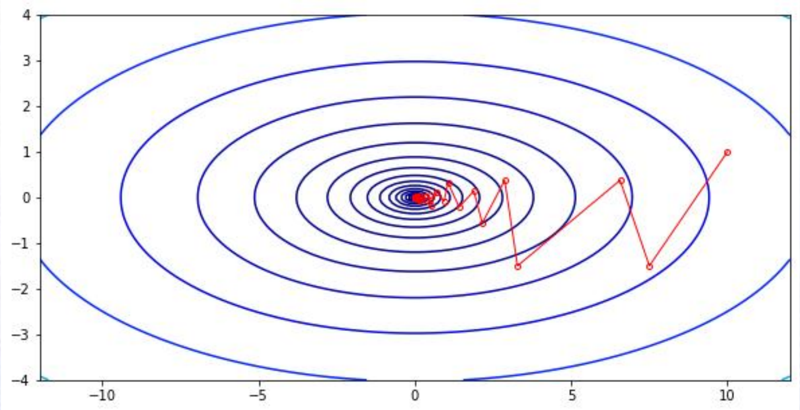
精确线搜索



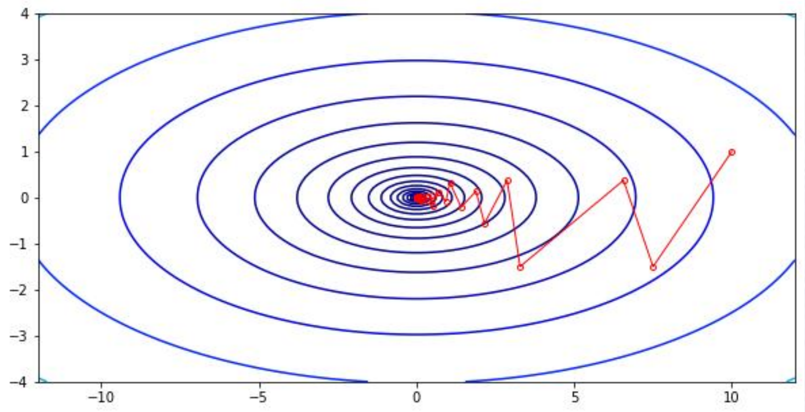
Armijo准则



$\alpha_k = 0.05$



Wolfe准则



非单调线搜索准则(Grippo)

Barzilar-Borwein算法

- **Barzilar-Borwein (BB) 算法**是一种特殊的梯度法，经常比一般的梯度法有着更好的效果. BB 方法的下降方向仍是点x^k 处的负梯度方向$-\nabla f(x^k)$，但**步长**α^k 并不是直接由线搜索算法给出的.
- BB 方法选取的α^k 是如下两个最优问题之一的解:

$$\min_\alpha \|\alpha y^{k-1} - s^{k-1}\|^2,$$

$$\min_\alpha \|y^{k-1} - \alpha^{-1}s^{k-1}\|^2,$$

其中引入记号 $s^{k-1} = x^k - x^{k-1}$ 以及 $y^{k-1} = \nabla f(x^k) - \nabla f(x^{k-1})$.

$$\alpha_{BB1}^k = \frac{\left(s^{k-1}\right)^T y^{k-1}}{\left(y^{k-1}\right)^T y^{k-1}} \qquad \alpha_{BB2}^k = \frac{\left(s^{k-1}\right)^T s^{k-1}}{\left(s^{k-1}\right)^T y^{k-1}}$$

Barzilar-Borwein算法

➢ 计算两种BB 步长的任何一种仅仅需要函数相邻两步的梯度信息和迭代点信息, 不需要任何线搜索算法即可选取算法步长. BB 方法计算出的步长可能过大或过小, 因此我们还需要将步长做上界和下界的截断, 即选取 $0 < \alpha_m < \alpha_M$ 使得$\alpha_m \le \alpha_k \le \alpha_M$.

Algorithm 2 非单调线搜索的 BB 方法

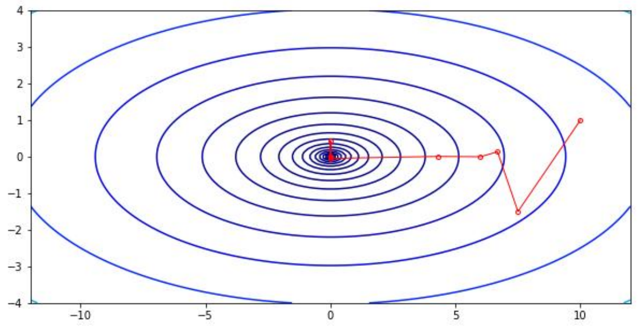
1: 给定 x^0, 选取初值 $\alpha > 0$, 整数 $M \ge 0$, $c_1, \beta, \varepsilon \in (0,1)$, $k = 0$.
2: **while** $\|\nabla f(x^k)\| > \varepsilon$ **do**
3:     **while** $f(x^k - \alpha\nabla f(x^k)) \ge \max_{0\le j\le \min(k,M)} f(x^{k-j}) - c_1\alpha\|\nabla f(x^k)\|^2$ **do**
4:         令 $\alpha \leftarrow \beta\alpha$.
5:     **end while**
6:     令 $x^{k+1} = x^k - \alpha\nabla f(x^k)$.
7:     根据 BB 步长公式之一计算 α, 并做截断使得 $\alpha \in [\alpha_m, \alpha_M]$.
8:     $k \leftarrow k + 1$.
9: **end while**

Barzilar-Borwein算法

```python
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x[0]**2 + 10*x[1]**2
def grad_f(x):
    return np.array([2*x[0], 20*x[1]])
def grippo_line_search(f, grad_f, x, direction, m=10, alpha_init=1.0, beta=0.5, c=1e-3, max_iter=100):
    history = [f(x)]  # 函数值历史记录
    grad = grad_f(x)
    slope = np.dot(grad, direction)
    alpha = alpha_init
    for _ in range(max_iter):
        candidate = x + alpha * direction
        f_current = f(candidate)
        f_max = max(history[-min(m, len(history)):])
        if f_current <= f_max + c * alpha * slope:
            history.append(f_current)
            return alpha
        alpha *= beta
    return alpha
```

```python
def Barzilar_Borwein(x, iter = 300, tol = 1e-6):
    x_history = [x]
    k = 0
    alpha = grippo_line_search(f, grad_f, x, -grad_f(x))
    x = x - alpha * grad_f(x)
    x_history.append(x)
    k = k + 1
    while k < iter:
        alpha = np.dot(x_history[-1]-x_history[-2],grad_f(x_history[-1])-grad_f(x_history[-2]))/np.dot(grad_f(x_history[-1])-grad_f(x_history[-2]),grad_f(x_history[-1])-grad_f(x_history[-2]))
        x = x - alpha * grad_f(x)
        x_history.append(x)
        k = k + 1
        if np.linalg.norm(grad_f(x)) < tol:
            break
    return x_history, k
x0 = np.array([10, 1])
gd_history, k = Barzilar_Borwein(x0)
print("最优解为", gd_history[-1])
print("梯度范数值为", np.linalg.norm(grad_f(gd_history[-1])))
x = np.linspace(-12, 12, 100)
y = np.linspace(-4, 4, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + 10*Y**2  # 定义目标函数
# 绘制等高线图
plt.figure(figsize=(10, 5))
plt.contour(X, Y, Z, levels=np.logspace(-2, 3, 20), cmap='jet')

plt.plot([x[0] for x in gd_history], [x[1] for x in gd_history], marker='o',markersize=4, linewidth=1, color = 'red', markerfacecolor='none', label='Gradient Descent')
plt.show()
```




```
最优解为 [0.00000000e+00 6.67909548e-16]
梯度范数值为 1.3358190954231156e-14
```

应用案例

➢ LASSO问题

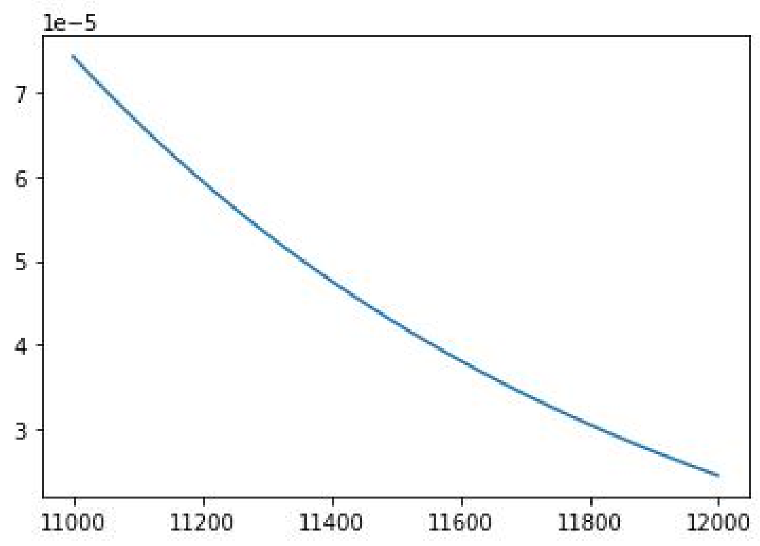
$$\min_x \frac{1}{2}\|Ax-b\|^2 + \mu\boxed{\|x\|_1}$$

$$l_\delta(x_i)=\begin{cases}\frac{1}{2\delta}x_i^2, & |x_i|<\delta,\\ |x_i|-\frac{\delta}{2}, & otherwise.\end{cases}$$

```
import numpy as np
def f(A, b, mu, n, x):
    res = 0.5 * np.linalg.norm(np.dot(A, x) - b) ** 2
    deta = 0.2
    for i in np.arange(n):
        if np.abs(x[i])<deta:
            res += mu * 1/(2 * deta) * x[i] ** 2
        else:
            res += mu * np.abs(x[i]) - deta/2
    return res

def grad_f(A, b, mu, n, x):
    res_grad = np.dot(A.T, np.dot(A, x) - b)
    deta = 0.2
    reg_grad = np.zeros(n)
    for i in np.arange(n):
        if np.abs(x[i])<deta:
            reg_grad[i] = x[i]/deta
        else:
            reg_grad[i] = np.sign(x[i])
    return res_grad + mu * reg_grad
```

```
def gradient_descent(A, b, mu, x, alpha, iter = 100000, tol = 1e-5):
    x_history = [x]
    k = 0
    while k < iter:
        x = x - alpha * grad_f(A, b, mu, n, x)
        x_history.append(x)
        k = k + 1
        if np.linalg.norm(grad_f(A, b, mu, n, x)) < tol:
            break
    return x_history, k
m = 512
n = 1024
A = np.random.randn(m,n)
u = np.zeros(n)
for i in np.arange(10):
    u[10*i] = 1
b = np.dot(A, u) + 1e-5 * np.random.randn(m)
x0 = np.random.randn(n)
alpha = 0.0005
mu = 1
gd_history, k = gradient_descent(A, b, mu, x0, alpha)
print("最优解为", gd_history[-1])
print("梯度范数值为", np.linalg.norm(grad_f(A, b, mu, n, gd_history[-1])))
```



次梯度算法

如果 $f(x)$不可微，该咋求解？梯度下降算法是否有效？

$$\min_x \max\left\{\frac{1}{2}x_1^2+(x_2-1)^2, \frac{1}{2}x_1^2+(x_2+1)^2\right\}$$

$$x^k=\begin{pmatrix}2(1+|\varepsilon_k|)\\ \varepsilon_k\end{pmatrix} \longrightarrow \nabla f(x^k)=\begin{pmatrix}2(1+|\varepsilon_k|)\\ 2(1+|\varepsilon_k|)sign(\varepsilon_k)\end{pmatrix}$$

精确线搜索

$$x^{k+1}=x^k-\alpha_k\nabla f(x^k)=\begin{pmatrix}2(1+|\varepsilon_k|/3)\\ -\varepsilon_k/3\end{pmatrix}$$

给定一个初始点$x^0=(2+2|\delta|,\delta)^T$，我们有$x^k\rightarrow(2,0)^T$. 然而$(2,0)^T$ 并不是稳定点.

次梯度算法

$$\min_x \ f(x)$$

凸但不可微

- 可类似梯度法构造如下次梯度算法的迭代格式：

$$x^{k+1} = x^k - \alpha_k g^k,\ g^k \in \partial f(x^k)$$

- 步长通常有如下四种选择：

I. 固定步长$\alpha_k = \alpha$

II. 固定$\|x^{k+1} - x^k\|$，即$\alpha_k\|g^k\|$ 为常数

III. 选取α_k 使其满足某种线搜索准则

IV. 消失步长$\alpha_k \to 0$ 且$\sum_k \alpha_k = +\infty$

次梯度算法

- f 为G-利普希茨连续的，当且仅当 $f(x)$ 的次梯度是有界的
- proof. 充分性：假设$\|g\| \le G$, $\forall g \in \partial f(x)$;
- 取$g_y \in \partial f(y)$, $g_x \in \partial f(x)$, 有

$$g_x^T(x-y) \ge f(x) - f(y) \ge g_y^T(x-y)$$

- 由柯西不等式得

$$G\|x-y\| \ge f(x) - f(y) \ge -G\|x-y\|$$

- 必要性：反设存在x 和$g \in \partial f(x)$，使得$\|g\| > G$;
- 取$y = x + g/\|g\|$，有

$$f(y) \ge f(x) + g^T(y-x) = f(x) + \|g\| > f(x) + G$$

- 这与 $f(x)$ 是G- 利普希茨连续的矛盾.

次梯度算法

I. f为凸函数；

II. f至少存在一个有限的极小值点x^*，且$f(x^*) > -\infty$；

III. f为利普希茨连续的

设$\{\boldsymbol{\alpha}_k > 0\}$为任意步长序列，则对任意$k \geq 0$, 有

$$2\left(\sum_{i=0}^{k}\alpha_i\right)\left(\hat{f}^k - f^*\right) \leq \left\|x^0 - x^*\right\|^2 + \sum_{i=0}^{k}\alpha_i^2 G^2$$

其中$\hat{f}^k = \min_{0\leq i\leq k} f\left(x^i\right)$

次梯度算法

$$\left\|x^{i+1}-x^*\right\|^2=\left\|x^i-\alpha_i g^i-x^*\right\|^2$$

$$=\left\|x^i-x^*\right\|^2-2\alpha_i\left\langle g^i,x^i-x^*\right\rangle+\alpha_i^2\left\|g^i\right\|^2$$

次梯度 + ||g|| ≤ G

$$\le\left\|x^i-x^*\right\|^2-2\alpha_i\left(f\left(x^i\right)-f^*\right)+\alpha_i^2G^2$$

$$2\alpha_i\left(f\left(x^i\right)-f^*\right)\le\left\|x^i-x^*\right\|^2-\left\|x^{i+1}-x^*\right\|^2+\alpha_i^2G^2$$

$$2\sum_{i=0}^{k}\alpha_i\left(f\left(x^i\right)-f^*\right)\le\left\|x^0-x^*\right\|^2-\left\|x^{k+1}-x^*\right\|^2+G^2\sum_{i=0}^{k}\alpha_i^2\le\left\|x^0-x^*\right\|^2+G^2\sum_{i=0}^{k}\alpha_i^2$$

$$\hat{f}^k=\min_{0\le i\le k}f\left(x^i\right)\longrightarrow 2\left(\sum_{i=0}^{k}\alpha_i\right)\left(\hat{f}^k-f^*\right)\le\left\|x^0-x^*\right\|^2+\sum_{i=0}^{k}\alpha_i^2G^2$$

次梯度算法

- 取 $\alpha_i = t$ 为固定步长，则

$$\hat{f}^k - f^* \le \frac{\left\|x^0 - x^*\right\|^2}{2kt} + \frac{G^2 t}{2}$$

无法保证收敛

- 取α_i使得$\|x^{i+1} - x^i\|$固定，即$\alpha_i\|g^i\| = s$为常数，则

$$\hat{f}^k - f^* \le \frac{G\left\|x^0 - x^*\right\|^2}{2ks} + \frac{Gs}{2}$$

无法保证收敛

- 消失步长$\alpha_k \to 0$ 且$\sum_k \alpha_k = +\infty$

$$\hat{f}^k - f^* \le \frac{\left\|x^0 - x^*\right\|^2 + G^2 \sum_{i=0}^{k} \alpha_i^2}{2\sum_{i=0}^{k} \alpha_i}$$

常用步长$\alpha_k = 1/k$

收敛

次梯度算法

➢ 取 $\alpha_i = t$ 为固定步长，则

$$\hat{f}^k - f^* \le \frac{\left\|x^0 - x^*\right\|^2}{2kt} + \frac{G^2 t}{2}$$

假设$\|x^0 - x^*\| \leqslant R$，并且总迭代步数 k 是给定的，则

$t = R / (G\sqrt{k})$时，右端达到最小

$$\hat{f}^k - f^* \le \frac{GR}{\sqrt{k}}$$

在$k = O(1/\epsilon^2)$ 步迭代后可以得到 ϵ 的精度

次梯度算法

- **取α_i使得$\|x^{i+1}-x^i\|$固定，即$\alpha_i\|g^i\|=s$为常数，则**

$$\hat{f}^k - f^* \le \frac{G\left\|x^0 - x^*\right\|^2}{2ks} + \frac{Gs}{2}$$

假设$\|x^0-x^*\|\leqslant R$，并且总迭代步数 k 是给定的，取$s=R/\sqrt{k}$

$$\hat{f}^k - f^* \le \frac{GR}{\sqrt{k}}$$

在$k=O(1/\epsilon^2)$步迭代后可以得到 ϵ 的精度

次梯度算法

➢ 一般情形

$$\left\|x^{i+1}-x^*\right\|^2 \le \left\|x^i-x^*\right\|^2 - 2\alpha_i\left(f\left(x^i\right)-f^*\right)+\alpha_i^2\|g^i\|^2$$

➢ **当$\alpha_i=(f(x^i)-f^*)/\|g^i\|^2$时，不等式右端达到最小值。**

$$\frac{\left(f\left(x^i\right)-f^*\right)^2}{\|g^i\|^2} \le \left\|x^i-x^*\right\|^2 - \left\|x^{i+1}-x^*\right\|^2$$

$$\hat{f}^k - f^* \le \frac{GR}{\sqrt{k}}$$

在$k=O(1/\epsilon^2)$步迭代后可以得到ϵ的精度

表明：步长的选取与最大迭代数无关

应用案例

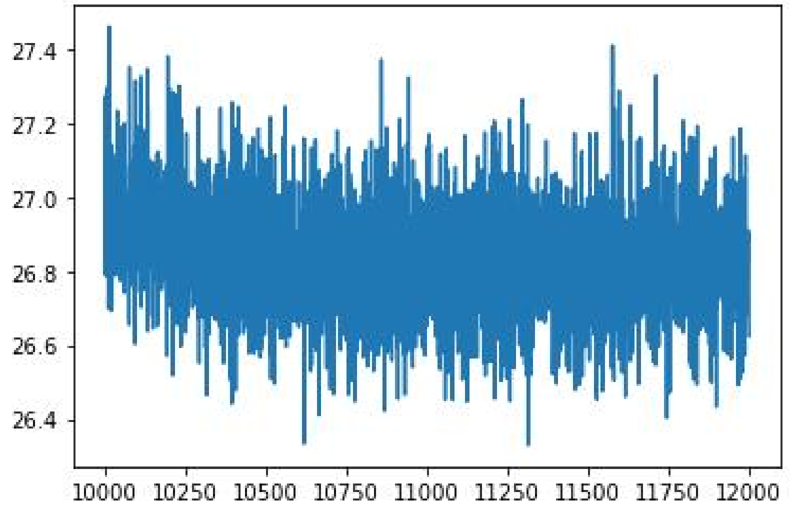
➢ LASSO问题

$$\min_{x} \frac{1}{2}\|Ax-b\|^2 + \mu \boxed{\|x\|_1}$$

```
import numpy as np
import matplotlib.pyplot as plt
def f(A, b, mu, x):
    res = 0.5 * np.linalg.norm(np.dot(A, x) - b) ** 2 + mu * np.linalg.norm(x,ord=1)
    return res

def grad_f(A, b, mu, x):
    res_grad = np.dot(A.T, np.dot(A, x) - b)
    reg_grad = np.zeros(n)
    for i in np.arange(n):
        reg_grad[i] = np.sign(x[i])
    return res_grad + mu * reg_grad
def sub_gradient(A, b, mu, x, iter = 12000, tol = 1e-5):
    x_history = [x]
    k = 1
    while k < iter:
        alpha = 0.002/np.sqrt(k)
        x = x - alpha * grad_f(A, b, mu, x)
        x_history.append(x)
        k = k + 1
        if np.linalg.norm(grad_f(A, b, mu, x)) < tol:
            break
    return x_history, k
```

```
m = 512
n = 1024
A = np.random.randn(m,n)
u = np.zeros(n)
for i in np.arange(10):
    u[10*i] = 1
b = np.dot(A, u) + 1e-5 * np.random.randn(m)
x0 = np.random.randn(n)
mu = 1
gd_history, k = sub_gradient(A, b, mu, x0)
print("最优解为", gd_history[-1])
print("梯度范数值为", np.linalg.norm(grad_f(A, b, mu, gd_history[-1])))
grad_history = []
for i in np.arange(10000,12000):
    grad_history.append(np.linalg.norm(grad_f(A, b, mu, gd_history[i])))
plt.plot(np.arange(10000,12000),grad_history)
```



大模型在大数据优化研究中的应用

基础智能层
能力训练层
诊断优化层
系统整合层

➢ AI+教育四维框架体系

☐ 基础智能层

I. 跨学科知识图谱

① 基础支撑：哲学与伦理、历史演进、数学基础、计算工具

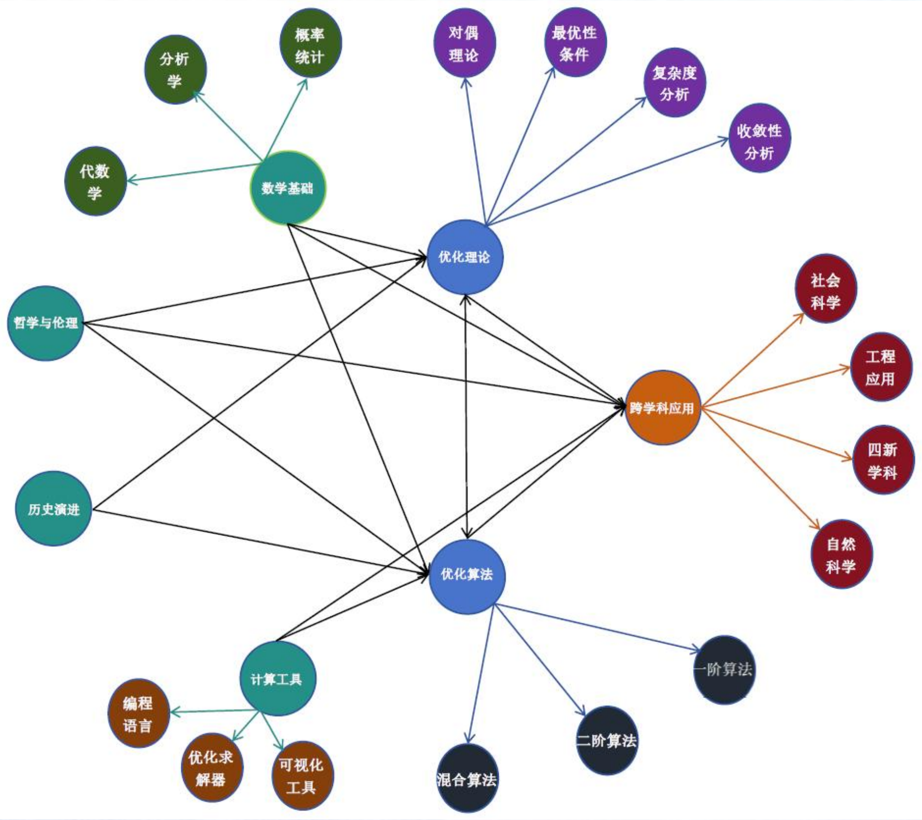
② 课程知识：优化理论、优化算法

③ 跨学科应用：社学科学、自然科学、工程应用、四新学科……

II. 个性化学习计划

① 知识架构

② 学习风格



注：大模型可以辅助提取核心知识点、阐释专业知识等，助力《大数据优化》知识梳理与跨学科知识理解

大模型在大数据优化研究中的应用

◻ 能力训练层

I. 理论推导

例: 帮我给出f(x)=sum(x_{i} ln(x_{i}))的凸性的论述，要求过程严谨和详细

函数 $f(x) = \sum_{i=1}^{n} x_i \ln x_i$ 的凸性分析

1. 定义域与二阶导数

- **定义域**：由于 $\ln x_i$ 要求 $x_i > 0$，故 $f(x)$ 的定义域为 $\mathrm{R}_+^n = \{(x_1, \ldots, x_n) \mid x_i > 0, \forall i\}$。
- **二阶导数**（Hessian矩阵）：
  对任意 x_i, x_j（$i \equiv j$），有：

$$\frac{\partial^2 f}{\partial x_i^2} = \frac{1}{x_i}, \quad \frac{\partial^2 f}{\partial x_i \partial x_j} = 0.$$

  因此，Hessian矩阵 H 为对角阵，其对角元素为 $\frac{1}{x_i}$，非对角元素均为0。

2. 凸性判定

- **正定性**：对于任意 $x \in \mathrm{R}_+^n$，Hessian矩阵 H 的对角元素 $\frac{1}{x_i} > 0$，故 H 正定。
- **凸性结论**：$f(x)$ 在 R_+^n 上是严格凸函数 [1] [2]。

3. 应用：Jensen不等式

由于 $f(x)$ 是凸函数，对任意权重 $\lambda_i \geq 0$（$\sum \lambda_i = 1$），满足：

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i).$$

II. 建模训练

例：帮我建立适合数据集((0,0),0),((0,1),0),((1,0),0),((1,1),1)的硬间隔支持向量机模型

III. 算法调试

例：帮我修正如下代码，要求框架不变

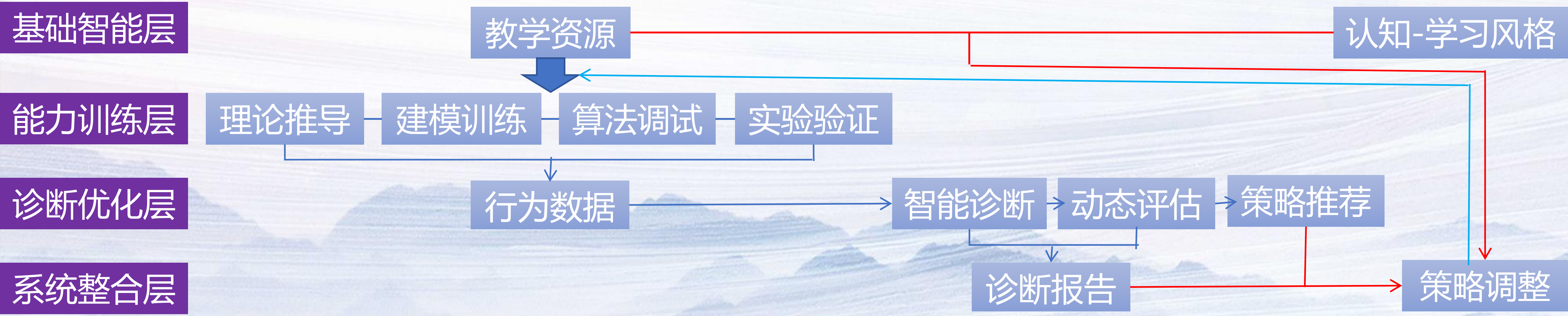
IV. 实验验证

例：对实验数据和结果的可视化

大模型在大数据优化研究中的应用

诊断优化层

系统整合层



课件资源：https://zhenhuapeng.github.io//coursematerials/

南昌大学
NANCHANG UNIVERSITY

感 谢 观 看

大数据优化：理论、算法及其应用

——来源于《最优化计算方法》（高教社）

★★★ 主讲人：彭振华 ★★★

联系方式：zhenhuapeng@ncu.edu.cn
zhenhuapeng@whu.edu.cn
15870605317（微信同号）

研究兴趣：1. 非凸非光滑优化算法与理论
2. 智能决策
3. 智能计算与机器学习

数学与计算机学院