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

2025年

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

目录

1. 大数据优化简介（3课时）
   - I. 模型与基本概念
   - II. 优化建模技术
   - III. 应用实例
   - IV. 求解器与大模型
2. 基础知识（7课时）
   - I. 范数与导数
   - II. 凸集与凸函数
   - III. 共轭函数与次梯度
3. 无约束优化理论（2课时）
   - I. 最优性问题解的存在性
   - II. 无约束可微问题的最优性理论
   - III. 无约束不可微问题的最优性理论
4. 无约束优化算法（15课时）
   - I. 线搜索方法
   - II. (次)梯度类算法
   - III. 共轭梯度算法
   - IV. (拟)牛顿类算法
   - V. 信赖域算法
   - VI.非线性最小二乘算法
5. 约束优化理论（6课时）
   - I. 对偶理论
   - II. 一般约束优化问题的最优性理论
   - III. 凸优化问题的最优性理论
6. 约束优化算法（3课时）
   - I. 罚函数法
   - II. 增广拉格朗日函数法
7. 复合优化算法（9课时）
   - I. PPA
   - II. BCD
   - III. PGD
   - IV. SGD
   - V. SDP

南昌大学
NANCHANG UNIVERSITY

无约束优化算法

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

第 四 部分

共轭梯度算法

$$\min_x \ f(x)$$

梯度下降算法收敛慢，牛顿算法需要存储和计算Hesse矩阵并求逆

- **共轭梯度法**最早是由Hestenes和Stiefle提出来的，在这个基础上，Fletcher和Reeves （1964）首先提出了解非线性优化问题的共轭梯度法。它的每一个搜索方向是互相共轭的，而这些搜索方向仅是**负梯度方向**与**上一次迭代搜索方向**的组合

$$d_k = \begin{cases} -\nabla f(x^0), & k = 0, \\ -\nabla f(x^k) + \beta_k d_{k-1}, & k \geq 1, \end{cases}$$

- β_k 的选择使得求解无约束二次规划时 d_k 与 d_{k-1} 关于 Q 相互共轭

共轭梯度算法

$$\min_x \ f(x)$$

$$d_k^T Q d_{k-1} = \left(-\nabla f\left(x^k\right) + \beta_k d_{k-1}\right)^T Q d_{k-1} = 0$$

$$\beta_k = \frac{\nabla f\left(x^k\right)^T Q d_{k-1}}{d_{k-1}^T Q d_{k-1}}$$

$$\nabla f\left(x^k\right) - \nabla f\left(x^{k-1}\right) = Q\left(x^k - x^{k-1}\right) = \alpha_{k-1} Q d_{k-1}$$

$$\beta_k^{HS} = \frac{\nabla f\left(x^k\right)^T \left(\nabla f\left(x^k\right) - \nabla f\left(x^{k-1}\right)\right)}{d_{k-1}^T \left(\nabla f\left(x^k\right) - \nabla f\left(x^{k-1}\right)\right)} \qquad Hestenes - Stiefel(1952)$$

共轭梯度算法

Algorithm 3 共轭梯度算法

1: 给定 x^0, $k = 0$.
2: **while** $\|\nabla f(x^k)\| > \varepsilon$ **do**
3: 计算搜索方向 $p_k = -\nabla f(x^k) + \beta_k p_{k-1}$ $(p_0 = -\nabla f(x^0))$.
4: 计算精确线搜索步长 α_k
5: 令 $x^{k+1} = x^k + \alpha_k p_k$.
6: $k \leftarrow k + 1$.
7: **end while**

- 例题：$\min_{x,y}\ 0.5x^2+y^2$ 初始点 $(2,1)^T$, 精确线搜索
- 练习：$\min_{x,y}\ x^2+4y^2$ 初始点 $(1,1)^T$, 精确线搜索

共轭梯度算法

$$\min_x \ f(x)$$

$$\beta_k^{FR} = \frac{\left\|\nabla f\left(x^k\right)\right\|^2}{\left\|\nabla f\left(x^{k-1}\right)\right\|^2} \qquad Fletcher-\mathrm{Re}eves(1964)$$

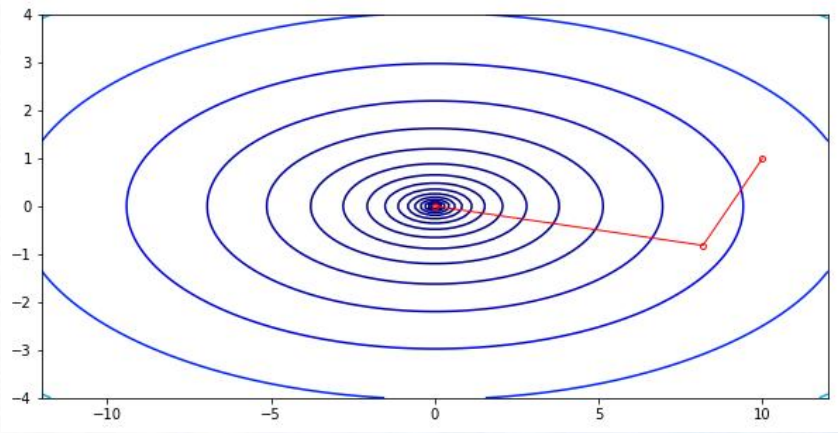
$$\beta_k^{PRP} = \frac{\nabla f\left(x^k\right)^T\left(\nabla f\left(x^k\right)-\nabla f\left(x^{k-1}\right)\right)}{\left\|\nabla f\left(x^{k-1}\right)\right\|^2} \qquad Polak-Ribi\grave{e}re-Polyak(1969)$$

$$\beta_k^{CD} = -\frac{\left\|\nabla f\left(x^k\right)\right\|^2}{d_{k-1}^T\nabla f\left(x^{k-1}\right)} \qquad Fletcher(1987)$$

$$\beta_k^{DY} = \frac{\left\|\nabla f\left(x^k\right)\right\|^2}{d_{k-1}^T\left(\nabla f\left(x^k\right)-\nabla f\left(x^{k-1}\right)\right)} \qquad Dai-Yuan(1995)$$

共轭梯度算法

```
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 exact_line_search(f, grad_f, x, direction):
    grad = grad_f(x)
    slope = np.dot(grad, direction)
    alpha = -
slope/np.dot(np.dot(np.array([[2,0],[0,20]]),direction),direction)
    return alpha
def conjugate_gradient(x, iter = 300, tol = 1e-6):
    x_history = [x]
    k = 0
    grad = grad_f(x)
    direct = -grad_f(x)
    alpha = exact_line_search(f, grad_f, x, direct)
    x = x + alpha * direct
    x_history.append(x)
    k = k + 1
    while k < iter:
        beta = np.dot(grad_f(x),grad_f(x)-grad)/np.dot(direct,grad_f(x)-grad)
        direct = -grad_f(x) + beta * direct
        alpha = exact_line_search(f, grad_f, x, direct)
        x = x + alpha * direct
        grad = 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 = conjugate_gradient(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 -5.55111512e-16]
梯度范数值为 1.1102230246251565e-14
```

经典牛顿算法

梯度下降算法收敛慢，只用了一阶信息

$$f\left(x^k+d^k\right)=f\left(x^k\right)+\nabla f\left(x^k\right)^T d^k+\frac{1}{2}\left(d^k\right)^T\nabla^2 f\left(x^k\right)d^k+o\left(\left\|d^k\right\|\right)$$

$$\min_{d^k} f\left(x^k\right)+\nabla f\left(x^k\right)^T d^k+\frac{1}{2}\left(d^k\right)^T\nabla^2 f\left(x^k\right)d^k+o\left(\left\|d^k\right\|\right)$$

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

$$\nabla^2 f\left(x^k\right)d^k=-\nabla f\left(x^k\right)$$ ← 牛顿方程

↓ 牛顿方向

- **经典牛顿算法：** $x^{k+1}=x^k-\nabla^2 f(x^k)^{-1}\nabla f(x^k)$

- **例题：** $\min_{x,y}\ 0.5x^2+y^2$ **初始点** $(2,1)^T$
- **练习：** $\min_{x,y}\ x^2+4y^2$ **初始点** $(1,1)^T$

经典牛顿算法

收敛性定理：假设 f **二阶连续可微**，且存在 x^* 的一个邻域 $N_\delta(x^*)$ 及常数 $L > 0$ 使得

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leqslant L\|x - y\|,\ \forall x, y \in N_\delta(x^*)$$

如果 $f(x)$ 满足 $\nabla f(x^*) = 0$, $\nabla^2 f(x^*)$ **正定**，则对于由经典牛顿算法产生的序列有：

- 如果初始点离 x^* 足够近，则迭代点列 x^k **收敛到** x^*；
- $\{x^k\}$ **Q-二次收敛**到 x^*；
- $\{\nabla f(x^k)\}$ **Q-二次收敛**到 0.

$\nabla f(x^*) = 0$

Proof. $x^{k+1} - x^* = x^k - \nabla^2 f(x^k)^{-1} \nabla f(x^k) - x^*$

$$= \nabla^2 f(x^k)^{-1} \left(\nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*)) \right)$$

$$\nabla f(x^k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x^k + t(x^* - x^k))(x^k - x^*)\, dt$$

$$\left\| \nabla^2 f(x^k)(x^k - x^*) - (\nabla f(x^k) - \nabla f(x^*)) \right\| = \left\| \int_0^1 \left(\nabla^2 f(x^k + t(x^* - x^k)) - \nabla^2 f(x^k) \right)(x^k - x^*)\, dt \right\|$$

$$\leq \int_0^1 \left\| \nabla^2 f(x^k + t(x^* - x^k)) - \nabla^2 f(x^k) \right\| \left\| x^k - x^* \right\| dt$$

$$\leq \left\| x^k - x^* \right\|^2 \int_0^1 Lt\, dt = \frac{L}{2} \left\| x^k - x^* \right\|^2$$

经典牛顿算法

矩阵范数的连续性 $\exists r>0$，对满足$\|x-x^*\|\le r$的所有x均有$\|\nabla^2 f(x)^{-1}\|\le 2\|\nabla^2 f(x^*)^{-1}\|$

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

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

$$\left\|x^{k+1}-x^*\right\|\le\left\|\nabla^2 f\left(x^k\right)^{-1}\right\|\left\|\nabla^2 f\left(x^k\right)\left(x^k-x^*\right)-\left(\nabla f\left(x^k\right)-\nabla f\left(x^*\right)\right)\right\|$$

$$\le L\left\|\nabla^2 f\left(x^*\right)^{-1}\right\|\left\|x^k-x^*\right\|^2$$

只需要对初始点进行限制，可得Q-二次收敛性。

经典牛顿算法

牛顿方程

$$\nabla^2 f\left(x^k\right) d^k = -\nabla f\left(x^k\right)$$

$$\begin{aligned}
\left\|\nabla f\left(x^{k+1}\right)\right\| &= \left\|\nabla f\left(x^{k+1}\right) - \nabla f\left(x^k\right) - \nabla^2 f\left(x^k\right) d^k\right\| \\
&= \left\|\int_0^1 \left(\nabla^2 f\left(x^k + t d^k\right) - \nabla^2 f\left(x^k\right)\right) d^k dt\right\| \\
&\le \int_0^1 \left\|\nabla^2 f\left(x^k + t d^k\right) - \nabla^2 f\left(x^k\right)\right\| \left\|d^k\right\| dt \\
&\le \left\|d^k\right\|^2 \int_0^1 L t \, dt = \frac{L}{2}\left\|d^k\right\|^2 \le 2L \left\|\nabla^2 f\left(x^*\right)^{-1}\right\|^2 \left\|\nabla f\left(x^k\right)\right\|^2
\end{aligned}$$

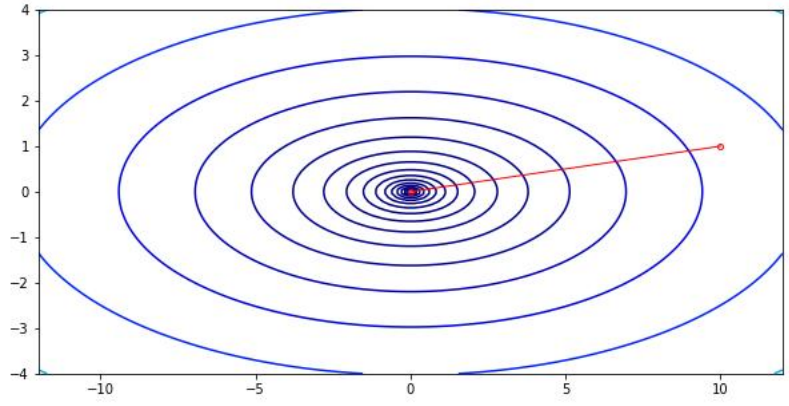
梯度的范数 Q-二次收敛到 0.

- 牛顿法收敛速度非常快, 但初始点需要距离最优解充分近。
- 常以梯度类算法先求得较低精度的解, 后用牛顿法加速.
- $\nabla^2 f(x^*)$需正定, 半正定条件下可能退化到 Q- 线性收敛

经典牛顿算法

```
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 newton(x, alpha = 1, iter = 300, tol = 1e-6):
    x_history = [x]
    k = 0
    while k < iter:
        x = x - alpha
*np.dot(np.linalg.inv(np.array([[2,0],[0,20]])),
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])
alpha = 0.085
gd_history, k = newton(x0)

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()
```



阻尼牛顿算法

- 例题：$\min_{x,y}\ 0.5x^2+y^2$ 初始点 $(2,1)^T$, 精确线搜索
- 练习：$\min_{x,y}\ x^2+4y^2$ 初始点 $(1,1)^T$, 精确线搜索

修正牛顿法

$$\min_x \ f(x)$$

经典牛顿法要求海瑟矩阵正定

- 对海瑟矩阵进行修正, 使其正定(所有特征值大于0);
- 用线搜索确定步长来增加算法的稳定性(Wolfe, Goldstein, Armijo).

Algorithm 4 带线搜索的修正牛顿法

1: 给定初始点 x^0.
2: **for** $k = 0, 1, 2, \cdots$ **do**
3: 确定矩阵 E^k 使得矩阵 $B^k \overset{\text{def}}{=} \nabla^2 f\left(x^k\right) + E^k$ 正定且条件数较小.
4: 求解修正的牛顿方程 $B^k d^k = -\nabla f\left(x^k\right)$ 得方向 d^k.
5: 使用任意一种线搜索准则确定步长 α_k.
6: 更新 $x^{k+1} = x^k + \alpha_k d^k$.
7: **end for**

修正牛顿法

收敛性定理：f在开集D上二阶连续可微, 且初始点x^0满足$\{x\in D: f(x)\le f(x^0)\}$为紧集. 若算法中$B^k$的条件数上界存在, 即$\exists C>0$，使得

$$\|B^k\|\,\|(B^k)^{-1}\|\le C$$

则$\lim_{k\to\infty}\nabla f(x^k)=0$，**即算法具有全局收敛性.**

如何修正矩阵 E^k

$$\nabla^2 f\left(x^k\right)=U\Lambda V^T$$

奇异值分解SVD

$$B^k=U\left(\Lambda+\tau_k I\right)V^T$$

$$\tau_k=\max\left\{0,\delta-\lambda_{\min}\left(\nabla^2 f\left(x^k\right)\right)\right\}$$

当正数τ_k足够大的时候, 可保证B^k正定, 但d^k会接近负梯度方向, 退化为一阶方法.

非精确牛顿法

海瑟矩阵计算存储困难，求逆或修正代价高

牛顿方程

$$\nabla^2 f\left(x^k\right) d^k = -\nabla f\left(x^k\right)$$

使用迭代法(如共轭梯度法) 求解牛顿方程，在一定的精度下提前停机, 以提高求解效率.

$$\nabla^2 f\left(x^k\right) d^k = -\nabla f\left(x^k\right) + r^k$$

残差

终止条件：$\|r^k\| \leq \eta_k \|\nabla f(x^k)\|$

非精确牛顿法

收敛定理：设函数 $f(x)$ 二阶连续可微, 且 $\nabla^2 f(x^*)$ 正定, 则

在非精确牛顿法中,

(1) 若 $\exists t < 1$, 使得 η_k 满足 $0 < \eta_k < t$, $k = 1, 2, \cdots$, 且起始点 x^0 充分靠近 x^*

并迭代最终收敛到 x^*, 则梯度 $\nabla f(x^k)$ 以 Q-线性收敛速度收敛;

(2) 若 $\lim_{k\to\infty} \eta_k = 0$ 成立, 则梯度 $\nabla f(x^k)$ 以 Q-超线性收敛速度收敛;

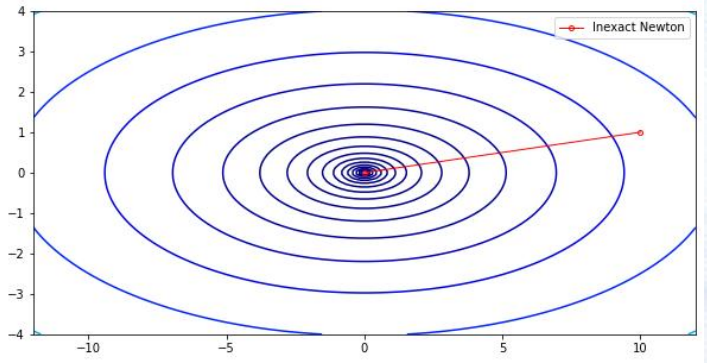
(3) 若(1) 或(2) 成立, 且 $\nabla^2 f$ 在 x^* 附近Lip. 连续, $\eta_k = O(\|\nabla f(x^k)\|)$, 则梯度 $\nabla f(x^k)$ 以 Q-二次收敛速度收敛.

1. 取 $\eta_k = \min\{0.5, \sqrt{\|\nabla f(x^k)\|}\}$, 可成立**局部的超线性收敛**;
2. 取 $\eta_k = \min\{0.5, \|\nabla f(x^k)\|\}$, 可成立**局部的二次收敛**.

非精确牛顿法

```
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 backtracking_line_search(x, p, alpha=1, beta=0.5, c=1e-4):
    """Armijo线搜索条件"""
    while f(x + alpha*p) > f(x) + c*alpha*np.dot(grad_f(x), p):
        alpha *= beta
    return alpha
def conjugate_gradient(A, b, max_iter=100):
    x = np.zeros_like(b)
    r = b - A @ x
    p = r.copy()
    for _ in range(max_iter):
        alpha = np.dot(r, r) / np.dot(p, A @ p)
        x = x + alpha * p
        r_new = r - alpha * (A @ p)
        if np.linalg.norm(r_new) < np.minimum(0.5,np.round(np.sqrt(np.linalg.norm(A @ p)))):
            break
        beta = np.dot(r_new, r_new) / np.dot(r, r)
        p = r_new + beta * p
        r = r_new
    return x
```

```
def inexact_newton(x0, max_iter=300, tol=1e-6):
    x_history = [x0]
    k = 0
    x = x0.copy()
    while k < max_iter:
        grad = grad_f(x)
        if np.linalg.norm(grad) < tol:
            break
        p = conjugate_gradient(np.array([[2, 0], [0, 20]]), -grad, max_iter=100)
        alpha = backtracking_line_search(x, p)
        x_new = x + alpha*p
        x = x_new
        x_history.append(x)
        k += 1
    return x_history, k
def plot_optimization(x_history):
    x = np.linspace(-12, 12, 100)
    y = np.linspace(-4, 4, 100)
    X, Y = np.meshgrid(x, y)
    Z = X**2 + 10*Y**2
    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 x_history],
             [x[1] for x in x_history],
             'r-o', markersize=4, linewidth=1,
             markerfacecolor='none', label='Inexact Newton')
    plt.legend()
    plt.show()
if __name__ == "__main__":
    x0 = np.array([10.0, 1.0])
    history, iterations = inexact_newton(x0)
    print(f"Converged in {iterations} iterations")
    plot_optimization(history)
```



二分类的逻辑回归模型

$$\min_x \ell(x) = \frac{1}{m}\sum_{i=1}^{m} \ln\left(1+\exp\left(-b_i a_i^T x\right)\right) + \lambda \|x\|_2^2$$

$$\nabla \ell(x) = \frac{1}{m}\sum_{i=1}^{m} \frac{\exp\left(-b_i a_i^T x\right)}{1+\exp\left(-b_i a_i^T x\right)} \cdot \left(-b_i a_i\right) + 2\lambda x$$

$$= -\frac{1}{m}\sum_{i=1}^{m} \left(1-p_i(x)\right) b_i a_i + 2\lambda x \left(p_i(x) = \frac{\exp\left(-b_i a_i^T x\right)}{1+\exp\left(-b_i a_i^T x\right)} \right)$$

$$\nabla^2 \ell(x) = \frac{1}{m}\sum_{i=1}^{m} \left(1-p_i(x)\right) p_i(x) a_i a_i^T + 2\lambda I \left(b_i^2 = 1\right)$$

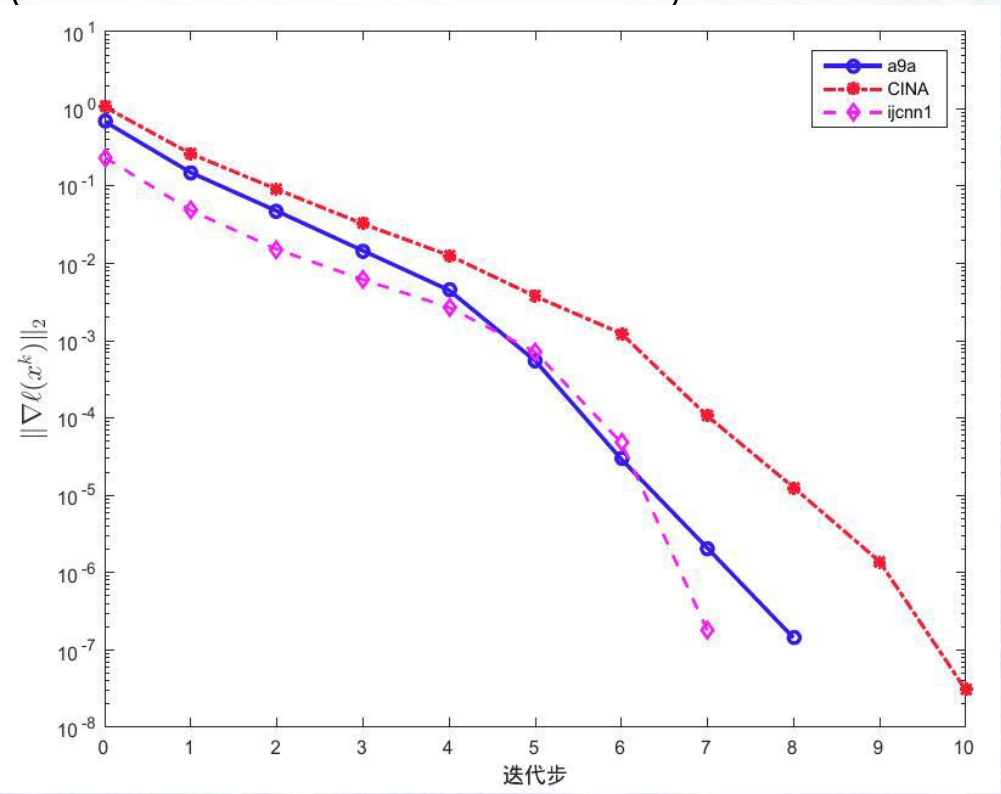
逻辑回归模型

使用LIBSVM 网站的数据集(具体数据集见下表), 对不同的数据集均调用非精确CG-牛顿法求解, 设置精度条件为

$$\left\|\nabla^2 \ell\left(x^k\right) d^k+\nabla \ell\left(x^k\right)\right\| \leq \min \left\{\left\|\nabla \ell\left(x^k\right)\right\|^2, 0.1\left\|\nabla \ell\left(x^k\right)\right\|\right\}$$

表: LIBSVM 数据集 (部分)

名称 \ 维数	m	n
$a9a$	16281	122
$ijcnn1$	91701	22
$CINA$	3206	132



南昌大学
NANCHANG UNIVERSITY

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

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