南昌大学
NANCHANG UNIVERSITY

优 化 建 模 与 深 度 学 习

★★★ 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

最优化简介

- 在给定的区域中寻找最小化或最大化某一函数的最优解

$$\left(\max_{x}\right)\min_{x} \quad f(x) \quad s.t. \quad x \in \chi.$$

决策变量　目标函数　约束条件　可行域

注意：$x=(x_1,x_2,\ldots,x_n)^T$是一个n维向量

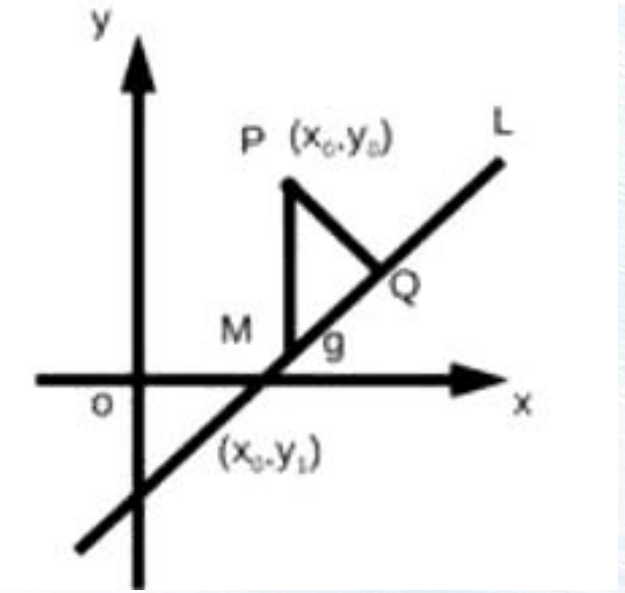
研究流程：

最优化简介

- **点到直线的距离**

$$\min_x \sqrt{(x-x_0)^2+(ax+b-y_0)^2}$$

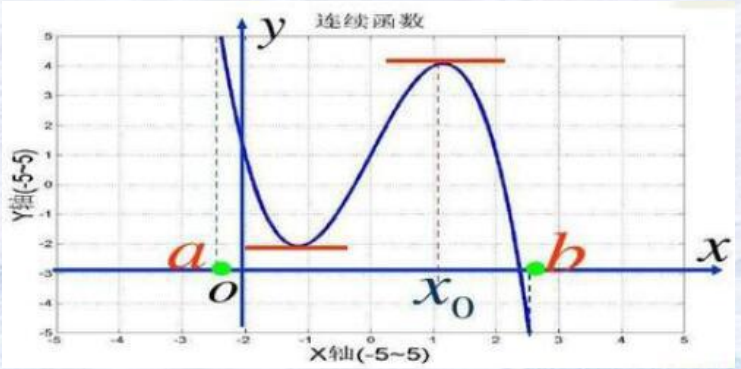
$$x=\frac{x_0+ay_0-ab}{1+a^2},\ d_{\min}=\frac{|ax_0+b-y_0|}{\sqrt{1+a^2}}.$$



- **固定周长面积最大的矩形是正方形**

$$\max_{\ell,d}\ \ell\times d\quad s.t.\ (\ell,d)\in\{(\ell,d):2(\ell+d)=22\}.$$

$$\ell=d=5.5.$$



- **费马引理: 利用导数为0 确定潜在的最优点**

$$\left(\max_x\right)\min_x\ f(x)\ \Rightarrow\ f'(x)=0$$

最优化简介

- **多元线性回归**

$$\min_{a,b} \sum_i \left(a^T x^i + b - y^i \right)^2$$

- **一元多项式回归**

$$\min_{a} \sum_i \left(a_0 + a_1 x^i + a_2 \left(x^i \right)^2 \cdots - y^i \right)^2$$

- 要判断一个模型好不好，仅仅看显著性是不够的。还需用拟合优度，指回归直线对观测值的拟合程度

$$R^2 = \frac{\sum_{i=1}^{N} \left(\hat{y}^i - \bar{y} \right)^2}{\sum_{i=1}^{N} \left(y^i - \bar{y} \right)^2}$$

- R的平方越接近1，回归方程对于样本数据点的拟合优度越高；反之，R的平方越接近0，回归方程对于样本数据点的拟合优度越低。

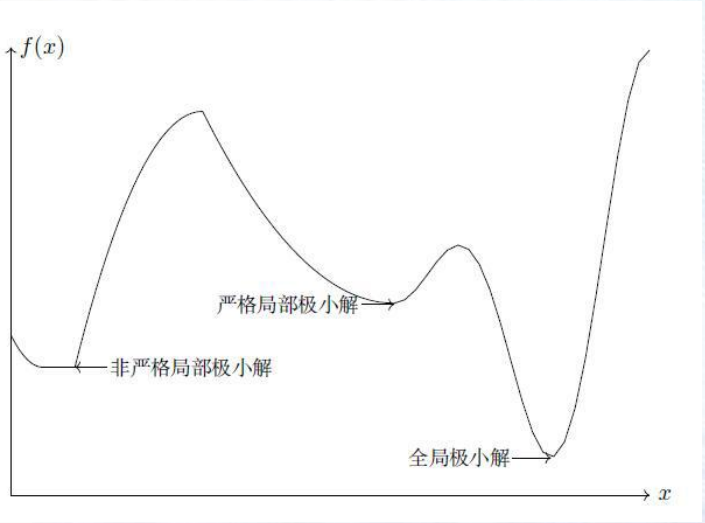
最优化简介

$$\min_{x} \ f(x) \quad s.t. \ x \in \chi.$$

- **可行解：** 可行域内所有的点
- **局部最优解：** 设$z \in X$, 若对所有$x \in N(z, \delta) \cap X$, 有

  $f(z) \leq f(x)$, 则称z是优化问题的局部最优解；
- **全局最优解：** 设$z \in X$, 若对所有$x \in X$, 有$f(z) \leq f(x)$,

  则称z是优化问题的(全局)最优解；
- **注记：若把“≤”换成“<”，则称严格(局部)极小解**



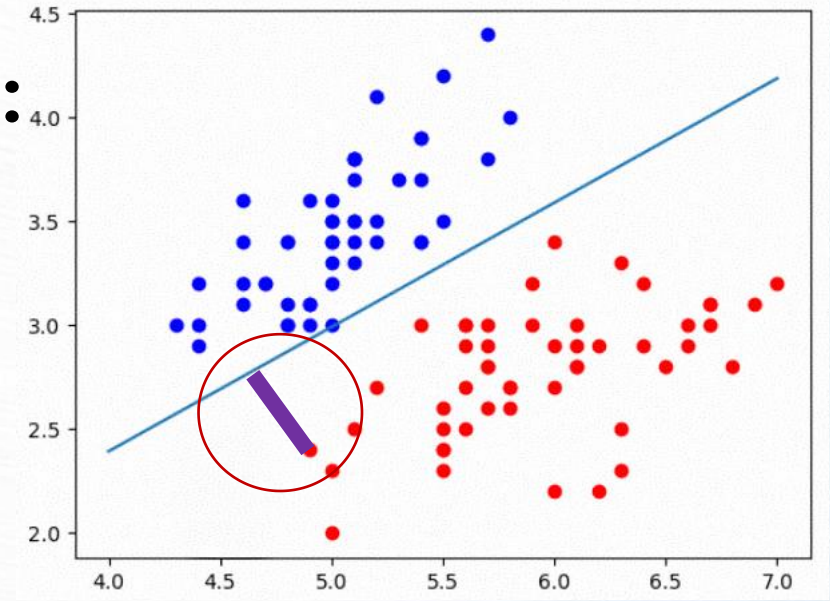
问题：试借助问题II阐释最优解的概念？

*课程思政：只有坚持党的领导，增强“四个意识”、坚定“四个自信”、做到“两个维护”，牢记“国之大者”(可行域X 内)，才有可能能实现自己的理想(最优解 $\bar{x}$)

最优化简介

- 对于正确分类的数据(x_i,y_i)，点**到超平面距离**：

$$d_i = \frac{\left|w^T x_i + b\right|}{\|w\|}$$



注意：绝对值函数在0处是不可导的

- 正确分类点**到超平面距离**可重写为

$$d_i = \frac{\left|w^T x_i + b\right|}{\|w\|} = \frac{y_i\left(w^T x_i + b\right)}{\|w\|}$$

正确分类点解读：

1. $w^T x_i + b > 0, y_i = +1 > 0$
2. $w^T x_i + b < 0, y_i = -1 < 0$

最优化简介

$$\max_{w.b}\min_{i}\frac{y_i\left(w^T x_i+b\right)}{\|w\|}\longrightarrow\max_{w.b,\,\gamma}\gamma\quad s.t.\quad\frac{y_i\left(w^T x_i+b\right)}{\|w\|}\geq\gamma.$$

- 取$\|w\|_2 = 1/\gamma$，则**硬间隔SVM模型**为

$$\min_{w.b}\frac{1}{2}\|w\|^2\quad s.t.\ y_i\left(w^T x_i+b\right)\geq 1.$$

- **例：考虑数据集：正实例点$x_1=(3,3)^T$, $x_2=(4,3)^T$, 负实例点$x_3=(1,1)^T$, 试用硬间隔SVM算法求SVM模型$f(x)=sign(w^Tx+b)$.**

$$\min_{w.b}\frac{1}{2}\left(w_1^2+w_2^2\right)$$

$$s.t.\ 3w_1+3w_2+b\geq 1, 4w_1+3w_2+b\geq 1, -w_1-w_2-b\geq 1.$$

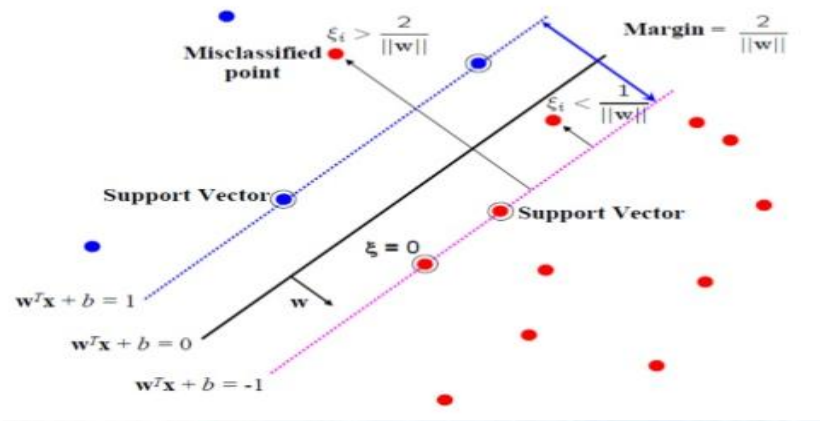
问题：试用硬间隔SVM算法实现逻辑与、逻辑或?

最优化简介

- **软间隔SVM模型为**

$$\min_{w.b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$s.t.\ \ y_i\left(w^T x_i + b\right) \geq 1-\xi_i, \xi_i \geq 0, i = 1,2,\ldots,N.$$



- **例：考虑数据集：正实例点$x_1=(3,3)^T$, $x_2=(4,3)^T$, 负实例点$x_3=(1,1)^T$, 试用软间隔SVM算法求SVM模型$f(x)=sign(w^Tx+b)$.**

$$\min_{w.b,\xi} \frac{1}{2}\left(w_1^2 + w_2^2\right) + C\left(\xi_1 + \xi_2 + \xi_3\right)$$

$$s.t.\ \ 3w_1 + 3w_2 + b \geq 1-\xi_1, 4w_1 + 3w_2 + b \geq 1-\xi_2, -w_1 - w_2 - b \geq 1-\xi_3, \xi_1, \xi_2, \xi_3 \geq 0.$$

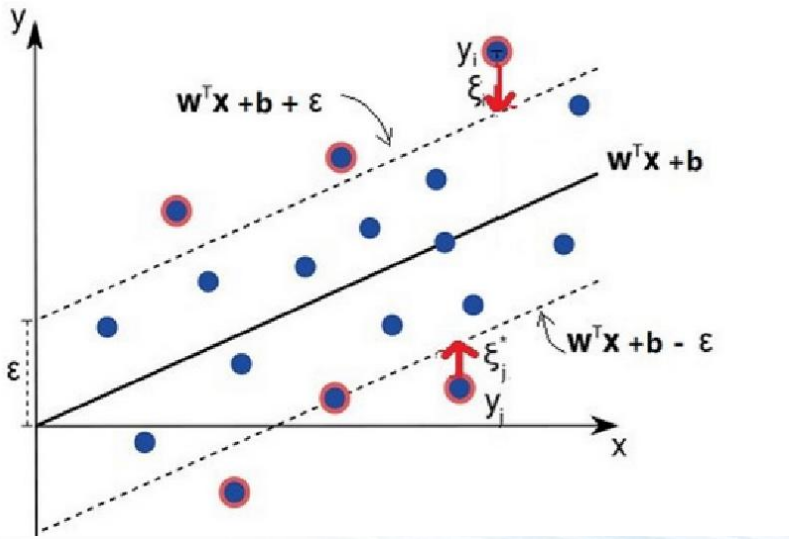
问题：试用软间隔SVM算法实现逻辑与、逻辑或?

*课程思政：软间隔SVM 允许数据结果存在一些偏差，我们做人和做事中也要养成包容的性格

最优化简介

- **SVR模型为**

$$\min_{w.b} \frac{1}{2}\|w\|^2 \quad s.t. \left|w^T x_i + b - y_i\right| \le \varepsilon.$$



- **考虑数据集*{((0, 1), 2), ((1, 1), 3)}*, 线性回归训练模型为**

$$\min_{w.b} \frac{1}{2}\left(w_1^2 + w_2^2\right)$$

$$s.t. \left|w_2 + b - 2\right| \le \varepsilon,$$

$$\left|w_1 + w_2 + b - 3\right| \le \varepsilon.$$

可用于药物筛选中ADMET（吸收、分布、代谢、排泄和毒性）预测

最优化简介

$$\min_x \ f(x)$$

- **情形一：函数 f 是连续可微的**

如何判断x^*是否是函数 f 的一个局部极小解或者全局极小解?

- **(一阶必要条件) 假设 f 在全空间R^n可微. 如果x^* 是一个局部极小点, 那么 $\nabla f(x^*) = 0$.（若函数 f 是凸的，则 $\nabla f(x^*) = 0 \rightarrow x^*$ 是 f 的全局极小点）**

- **情形二：函数 f 是不可微的**

- **(一阶充要条件) 假设 f 是适当且凸的函数.. 如果x^* 是一个全局极小点, 当且仅当 $0 \in \partial f(x^*)$.**

假设 f 是可微的, h 是凸函数(可能非光滑). 如果x^* 是$f + h$ 的一个局部极小点, 当且仅当$-\nabla f(x^*) \in \partial h(x^*)$.

最优化简介

$$\min_x \ f(x)$$
$$s.t. \ \ g_i(x) \le 0, i \in I,$$
$$h_j(x) = 0, j \in E.$$

- **必要性条件**：假设 x^* 是一般优化问题的一局部最优解，如果$T_X(x^*) = F(x^*)$成立，那么存在拉格朗日乘子 λ_j和 μ_i使得

稳定性条件： $\nabla f(x^*) + \sum_{j \in E} \lambda_j \nabla h_j(x^*) + \sum_{i \in I} \mu_i \nabla g_i(x^*) = 0$

原始可行性： $g_i(x^*) \le 0, \forall i \in I, h_j(x^*) = 0, \forall j \in E$

对偶可行性： $\mu_i \ge 0, \forall i \in I$

互补条件： $\mu_i g_i(x^*) = 0, \forall i \in I$

凸规划 ✚ Slater CQ

↓

充要条件

线性约束品性

↓

线性无关约束品性 → Mangasarian Fromovitz约束品性 → $T_X(x) = F(x)$

最优化简介

- 逻辑非：$y \quad 1-y \quad y \in \{0, 1\}$
- 逻辑与：$y \le x_1 \quad y \le x_2 \quad y \ge x_1 + x_2 - 1 \quad x_1, x_2, y \in \{0, 1\}$

  $y \le x_i \quad y \ge \Sigma x_i - (N-1) \quad x_i, y \in \{0, 1\}$
- 逻辑或：$y \ge x_1 \quad y \ge x_2 \quad y \le x_1 + x_2 \quad x_1, x_2, y \in \{0, 1\}$

  $y \ge x_i \quad y \le \Sigma x_i \quad x_i, y \in \{0, 1\}$
- 逻辑异或：$y \ge x_1 - x_2 \quad y \ge x_2 - x_1 \quad y \le x_1 + x_2 \quad y \le 2 - x_1 - x_2 \quad x_1, x_2, y \in \{0, 1\}$

  $y \ge x_i - \Sigma_{i \ne j} x_j \quad y \le \Sigma x_i \quad \Sigma x_i \le N + (1-N)y \quad x_i, y \in \{0, 1\}$

- 当 $y = 0$ 时，$x \leq 0 : x - My \leq 0$

约束描述	约束
如果 $y = 1$，则 $z = 1$	$z \geqslant y,\ z, y \in \{0, 1\}$
如果 $y = 1$，则 $f(x) \leqslant 0$	$f(x) - M(1 - y) \leqslant 0,\ y \in \{0, 1\}$
如果 $y = 1$，则 $f(x) < 0$	$f(x) + \epsilon - M(1 - y) \leqslant 0,\ y \in \{0, 1\}$
如果 $y = 1$，则 $f(x) \leqslant 0$；否则 $f(x) > 0$	**方法 1**: $f(x) - M(1 - y) \leqslant 0$ $f(x) - \epsilon + My \geqslant 0$ $y \in \{0, 1\}$ **方法 2**: $f(x) - M(1 - z_1) \leqslant 0$ $-(1 - z_1) \leqslant y - 1 \leqslant (1 - z_1)$ $-(f(x) - \epsilon) - M(1 - z_2) \leqslant 0$ $-(1 - z_2) \leqslant y \leqslant (1 - z_2)$ $z_1 + z_2 = 1,\ z_1, z_2, y \in \{0, 1\}$

最优化简介

如果 $y=1$，则 $f(x)\leqslant 0$；否则 $g(x)\leqslant 0$	**方法 1**: $f(x)-M(1-y)\leqslant 0$ $g(x)-My\leqslant 0$ $y\in\{0,1\}$ **方法 2**: $f(x)-M(1-z_1)\leqslant 0$ $-(1-z_1)\leqslant y-1\leqslant(1-z_1)$ $g(x)-M(1-z_2)\leqslant 0$ $-(1-z_2)\leqslant y\leqslant(1-z_2)$ $z_1+z_2=1,\ z_1,z_2,y\in\{0,1\}$
如果 $y=1$，则 $f(x)=0$	$f(x)-M(1-y)\leqslant 0$ $-f(x)-M(1-y)\leqslant 0$ $y\in\{0,1\}$
如果 $f(x)\leqslant 0$，则 $y=1$	$f(x)-\epsilon+My\geqslant 0,\ y\in\{0,1\}$
如果 $f(x)\leqslant 0$，则 $y=1$；否则 $y=0$	**方法 1**: $f(x)-\epsilon+My\geqslant 0$ $f(x)-M(1-y)\leqslant 0$ $y\in\{0,1\}$ **方法 2**: $f(x)-M(1-z_1)\leqslant 0$ $-(1-z_1)\leqslant y-1\leqslant(1-z_1)$ $f(x)-\epsilon+M(1-z_2)\geqslant 0$ $-(1-z_2)\leqslant y\leqslant(1-z_2)$ $z_1+z_2=1,\ z_1,z_2,y\in\{0,1\}$

最优化简介

约束描述	约　束
如果 $f(x)=0$，则 $y=1$	$-(1-z_1)\leqslant y\leqslant(1-z_1)$ $f(x)\leqslant-\epsilon+M(1-z_1)$ $-(1-z_2)\leqslant y-1\leqslant(1-z_2)$ $-M(1-z_2)-\epsilon\leqslant f(x)\leqslant\epsilon+M(1-z_2)$ $-(1-z_3)\leqslant y\leqslant(1-z_3)$ $f(x)\geqslant\epsilon-M(1-z_3)$ $z_1+z_2+z_3=1,\ z_1,z_2,z_3,y\in\{0,1\}$
如果 $f(x)\leqslant0$，则 $g(x)\leqslant0$	$f(x)-\epsilon+My\geqslant0$ $g(x)-M(1-y)\leqslant0$ $y\in\{0,1\}$
如果 $f(x)\leqslant0$ 且 $h(x)\leqslant0$，则 $g(x)\leqslant0$	$f(x)-\epsilon+Mz_1\geqslant0$ $h(x)-\epsilon+Mz_2\geqslant0$ $g(x)-M(1-z_3)\leqslant0$ $z_3\geqslant z_1+z_2-1$ $z_1,z_2,z_3\in\{0,1\}$

至少有m个不等式成立

$$\sum_{i=1}^{N} a_{ki}x_i \leqslant b_k + M\left(1-y_k\right), \qquad \forall k=1,\cdots,K$$

$$\sum_{i=1}^{K} y_i \geqslant m$$

$$y_k \in \{0,1\}, \qquad \forall k=1,\cdots,K$$

至少有m个等式成立

$$\sum_{i=1}^{N} a_{ki}x_i + w_k = b_k, \qquad \forall k=1,\cdots,K$$

$$w_k \leqslant M\left(1-y_k\right), \qquad \forall k=1,\cdots,K$$

$$w_k \geqslant -M\left(1-y_k\right), \qquad \forall k=1,\cdots,K$$

$$\sum_{i=1}^{K} y_i \geqslant m,$$

$$y_k \in \{0,1\}, w_k\text{无约束}, \qquad \forall k=1,\cdots,K$$

- 乘积形式 x_1x_2
- 情形一：均为0-1变量

$$y \le x_1, y \le x_2, y \ge x_1 + x_2 - 1, x_1, x_2, y \in \{0, 1\}$$

- 情形二：一 0-1变量，一[0,u]连续变量

$$y \le ux_1, y \le x_2, y \ge x_2 - u(1 - x_1), x_1 \in \{0, 1\}, x_2 \in [0, u], y \in [0, u]$$

- 情形三：一 0-1变量，一[l,u]连续变量

$$y \le x_2, y \ge x_2 - u(1 - x_1), lx_1 \le y \le ux_1, x_1 \in \{0, 1\}, x_2 \in [l, u], y \in [l, u]$$

- 情形四：两连续变量$l_i \le x_i \le u_i$

$$(x_i - l_i)(x_j - l_j) \ge 0, (u_i - x_i)(u_j - x_j) \ge 0,$$

$$(u_i - x_i)(x_j - l_j) \ge 0, (x_i - l_i)(u_j - x_j) \ge 0, l_il_j \le w_{ij} \le u_iu_j$$

最优化简介

- 取整

$$\min y \ \text{ s.t. } \ y \geq x, \ \ y \leq x + 1, y \in \mathbf{Z}$$

- 绝对值 $|x| \leq M$

$$z = x_p + x_n, x = x_p - x_n,$$

$$x_p \leq My, x_n \leq M(1 - y), y \in \{0, 1\}, z, x_p, x_n \geq 0$$

- min/max函数

$$\max\{x_1, x_2, \ldots, x_N\} \qquad \min \ y \quad \text{s.t. } \ y \geq x_i, i = 1, 2, \ldots, N.$$

$$y \geq x_i, y \leq x_i + (U_{max} - L_i)(1 - v_i), i = 1, 2, \ldots, N,$$

$$\Sigma v_i = 1, v_i \in \{0, 1\}.$$

最优化简介

min/max函数

$$\min\{x_1, x_2, \ldots, x_N\} \qquad \max\ y \quad \text{s.t.}\ y \le x_i,\ i = 1, 2, \ldots, N.$$

$$y \le x_i,\ y \ge x_i + (U_i - L_{min})(1 - v_i),\ i = 1, 2, \ldots, N,$$

$$\Sigma v_i = 1,\ v_i \in \{0, 1\}.$$

分式函数

$$\frac{\sum_i (c_i x_i + \alpha)}{\sum_i (d_i x_i + \beta)} + y = \frac{1}{\sum_i (d_i x_i + \beta)} > 0, z_i = x_i y \Rightarrow \sum_i (c_i z_i + \alpha y)$$

$$\sum_i (d_i z_i + \beta y) = 1, y > 0, z_i \ge 0$$

优化求解器的介绍与使用

- Gurobi：大规模混合整数规划，普通线性/二次规划，备选Cplex
- MOSEK：二阶锥规划（SOCP）、半定规划（SDP）
- scipy.optimize.minimize：常规非线性凸优化（开源）
- COPT：线性规划

scipy.optimize.minimize(**fun, x0, args=(), method=None, jac=None, hess=None,**

bounds=None, constraints=(), tol=None, callback=None, options=None)

线性规划　非线性规划　混合整数规划　组合优化

优化求解器的介绍与使用

```
from scipy.optimize import minimize
from scipy.optimize import LinearConstraint
import numpy as np
def obj(x):
    return -x[0] - 2*x[1]
def grad(x):
    return np.array([-1.0, -2.0])
A = [[1, 1],
     [2, 1],
     [1, 2],
     [1, 0]]
lb = [-np.inf, -np.inf, 8, 0]
ub = [4, 5, 8, 0]
linear_constraint = LinearConstraint(A, lb, ub)
bounds = [(0, None), (0, None)]
x0 = [0, 0]
res = minimize(obj, x0, method='trust-constr', jac=grad, bounds=bounds, constraints=linear_constraint)
print("最优解:", res.x)
print("最优值:", res.fun)
```

```
最优解: [-2.6990608e-09  4.0000000e+00]
最优值: -7.999999991531572
```

优化求解器的介绍与使用

```
from scipy.optimize import linprog
c = [-1, -2]
A_ub = [[1, 1],
        [2, 1]]
b_ub = [4, 5]
A_eq = [[1, 2],
        [1, 0]]
b_eq = [8, 0]
bounds = [(0, None), (0, None)]
res = linprog(c, A_ub=A_ub, b_ub=b_ub, A_eq=A_eq, b_eq=b_eq, bounds=bounds,
method='highs')
print("最优解:", res.x)
print("最优值:", res.fun)
```

```
最优解: [0. 4.]
最优值: -8.0
```

优化求解器的介绍与使用

```
from scipy.optimize import minimize
import numpy as np
def obj(x):
    return -x[0] - 2*x[1]
def grad(x):
    return np.array([-1.0, -2.0])
cons = [
    {'type': 'ineq', 'fun': lambda x: 4 - x[0] - x[1]},
    {'type': 'ineq', 'fun': lambda x: 5 - 2*x[0] - x[1]},
    {'type': 'eq',   'fun': lambda x: x[0] + 2*x[1] - 8},
    {'type': 'eq',   'fun': lambda x: x[0]}
]
bounds = [(0, None), (0, None)]
x0 = [0, 0]
res = minimize(obj, x0, method='SLSQP', jac=grad, bounds=bounds, constraints=cons)
print("最优解:", res.x)
print("最优值:", res.fun)
```

```
最优解: [0. 4.]
最优值: -8.0
```

优化求解器的介绍与使用

```
import numpy as np
from scipy.optimize import minimize
def objective(x):
    return (1 - x[0])**2 + 100 * (x[1] - x[0]**2)**2
def gradient(x):
    df_dx1 = -2 * (1 - x[0]) - 400 * x[0] * (x[1] - x[0]**2)
    df_dx2 = 200 * (x[1] - x[0]**2)
    return np.array([df_dx1, df_dx2])
def constraint_ineq(x):
    return 2 - x[0]**2 - x[1]**2
def constraint_eq(x):
    return x[0] + x[1] - 1
cons = [
    {'type': 'ineq', 'fun': constraint_ineq},
    {'type': 'eq', 'fun': constraint_eq}
]
bounds = [(0, None), (0, None)]
x0 = np.array([0.5, 0.5])
res = minimize(objective, x0, method='SLSQP', jac=gradient,
               bounds=bounds, constraints=cons)
print(f"最优解: {res.x}")
print(f"最优值: {res.fun:.4f}")
```

```
最优解: [0.61879367 0.38120633]
最优值: 0.1456
```

优化求解器的介绍与使用

```
import numpy as np
from scipy.optimize import minimize
X = np.array([[1, 5], [2, 10], [3, 15], [4, 20], [5, 25]])
y = np.array([55, 65, 75, 85, 95])
def objective(wb):
    w = wb[:-1]
    b = wb[-1]
    error = 0
    for i in range(len(X)):
        error += (y[i] - (np.dot(w, X[i]) + b)) ** 2
    return error
initial_guess = np.array([0, 0, 0])
result = minimize(objective, initial_guess)
print(f"权重: {result.x[:-1]}")
print(f"偏置: {result.x[-1]}")
```

```
权重: [0.38516702 1.92296656]
偏置: 45.00000053250538
```

优化求解器的介绍与使用

```
import numpy as np
from scipy.optimize import minimize
X = np.array([[1, 2], [2, 3], [3, 3], [2, 1], [3, 1], [4, 2]])
y = np.array([1, 1, 1, -1, -1, -1])
def objective(wb):
    w = wb[:-1]
    return 0.5 * np.linalg.norm(w) ** 2
def constraint(wb, i):
    w = wb[:-1]
    b = wb[-1]
    return y[i] * (np.dot(w, X[i]) + b) - 1
constraints = [{'type': 'ineq', 'fun': lambda wb, i=i: constraint(wb, i)} for i in range(len(X))]
x0 = np.array([0, 0, 0])
result = minimize(objective, x0, constraints=constraints)
print(f'权重 w: {result.x[:-1]}')
print(f'偏置 b: {result.x[-1]}')
```

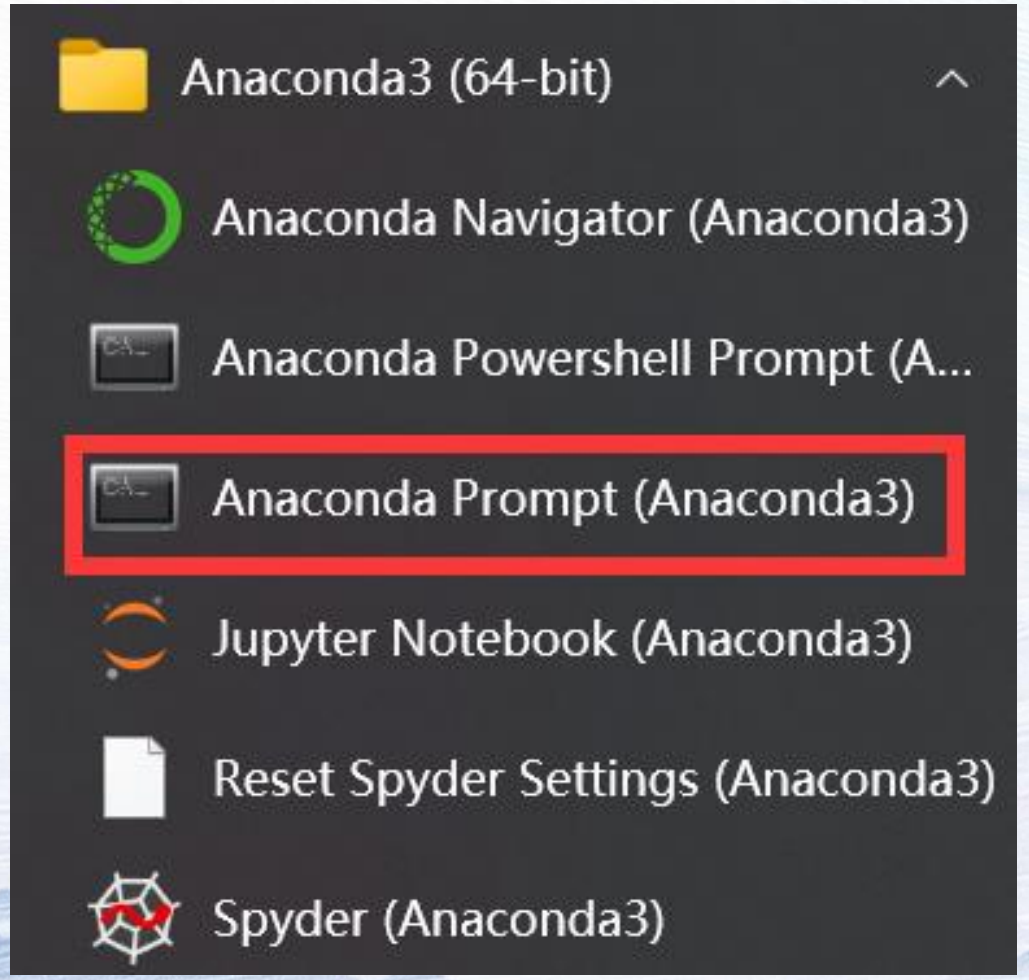
```
权重 w: [-0.66666667  1.33333333]
偏置 b: -0.9999999999999996
```

优化求解器的介绍与使用

Gurobipy加载

1. 打开Prompt

2. pip install gurobipy



优化求解器的介绍与使用

Step 1：前往获得学术许可，获取grbgetkey

https://support.gurobi.com/hc/en-us/articles/13210193318033-What-is-an-Academic-WLS-license

Step 2：运行Anaconda Prompt，执行conda install -c gurobi gurobi=13.0.0

Step 3：执行grbgetkey xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx，输入gurobi.lic生成文件

Step 4：将该文件复制到C:\gurobi

优化求解器的介绍与使用

```
import gurobipy as gp
from gurobipy import GRB
model = gp.Model()
x = model.addVars(range(3), vtype=GRB.CONTINUOUS, name="x")
model.addConstr(x[0] + 2 * x[1] + 3 * x[2] <= 1, name='c1')
model.addConstr(x[0] + x[1] >= 1, name='c2')
model.setObjective(x[0] ** 2 + x[1] ** 2 + 2 * x[2] ** 2, GRB.MINIMIZE)
model.optimize()
if model.status == GRB.OPTIMAL:
    print(f"最优值 (Objective Value): {model.ObjVal}")
    print("最优解 (Variable Values):")
    for i in range(3):
        print(f"x[{i}] = {x[i].X}")
elif model.status == GRB.INFEASIBLE:
    print("模型无可行解 (Infeasible)")
else:
    print(f"求解未成功，状态码: {model.status}")
```

```
最优值 (Objective Value): 1.000000000000143
最优解 (Variable Values):
x[0] = 1.0000000000000715
x[1] = 8.571800034080469e-15
x[2] = 0.0
```

优化求解器的介绍与使用

```
import gurobipy as gp
from gurobipy import GRB
model = gp.Model()
x = model.addVar(vtype=GRB.INTEGER, name="x")
y = model.addVars(range(2), vtype=GRB.CONTINUOUS, name="y")
model.addConstr(x + 2 * y[0] + 3 * y[1] <= 1, name='c1')
model.addConstr(x + y[0] >= 1, name='c2')
model.setObjective(x ** 2 + y[0] ** 2 + 2 * y[1] ** 2, GRB.MINIMIZE)
model.optimize()
if model.status == GRB.OPTIMAL:
    print(f"最优值 (Objective Value): {model.ObjVal}")
    print("最优解 (Variable Values):")
    print(f"x = {x.X}")
    for i in range(2):
        print(f"y[{i}] = {y[i].X}")
elif model.status == GRB.INFEASIBLE:
    print("模型无可行解 (Infeasible)")
else:
    print(f"求解未成功，状态码: {model.status}")
```

```
最优值 (Objective Value): 1.0
最优解 (Variable Values):
x = 1.0
y[0] = 0.0
y[1] = 0.0
```

优化求解器的介绍与使用

```
import numpy as np
import gurobipy as gp
from gurobipy import GRB
X = np.array([[1, 5], [2, 10], [3, 15], [4, 20], [5, 25]])
y = np.array([55, 65, 75, 85, 95])
n_samples, n_features = X.shape
model = gp.Model("LinearRegression_QP")
w = model.addVars(n_features, lb=-GRB.INFINITY, ub=GRB.INFINITY, name="w")
b = model.addVar(lb=-GRB.INFINITY, ub=GRB.INFINITY, name="b")
obj_expr = 0.0
for i in range(n_samples):
    pred_i = gp.quicksum(w[j] * X[i, j] for j in range(n_features)) + b
    residual_i = y[i] - pred_i
    obj_expr += residual_i * residual_i
model.setObjective(obj_expr, GRB.MINIMIZE)
model.optimize()
w_vals = [w[j].X for j in range(n_features)]
b_val = b.X
print(f"权重 (w): {w_vals}")
print(f"偏置 (b): {b_val}")
```

```
权重 (w): [7.191010878693699, 0.5617978242610709]
偏置 (b): 44.99999999999703
```

优化求解器的介绍与使用

```
import numpy as np
import gurobipy as gp
from gurobipy import GRB
X = np.array([[1, 2], [2, 3], [3, 3], [2, 1], [3, 1], [4, 2]])
y = np.array([1, 1, 1, -1, -1, -1])
n_samples, n_features = X.shape
model = gp.Model("SVM_Primal")
w = model.addVars(n_features, lb=-GRB.INFINITY, name="w")
b = model.addVar(lb=-GRB.INFINITY, name="b")
obj_expr = 0.5 * gp.quicksum(w[j] * w[j] for j in range(n_features))
model.setObjective(obj_expr, GRB.MINIMIZE)
for i in range(n_samples):
    dot_product = gp.quicksum(w[j] * X[i, j] for j in range(n_features))
    model.addConstr(y[i] * (dot_product + b) >= 1, name=f"constraint_{i}")
model.optimize()
w_vals = np.array([w[j].X for j in range(n_features)])
b_val = b.X
print(f'权重 w: {w_vals}')
print(f'偏置 b: {b_val}')
```

```
权重 w: [-0.66666667  1.33333334]
偏置 b: -1.0000000002539198
```

TSP

旅行商问题（TSP）求解 - 知乎

```
最优路径顺序: [0, 2, 3, 1, 4]
总距离: 6.5345
完整回路: [0, 2, 3, 1, 4, [0, 2, 3, 1, 4]]
```

南昌大学
NANCHANG UNIVERSITY

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

感 谢 观 看

优 化 建 模 与 深 度 学 习

★★★ 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