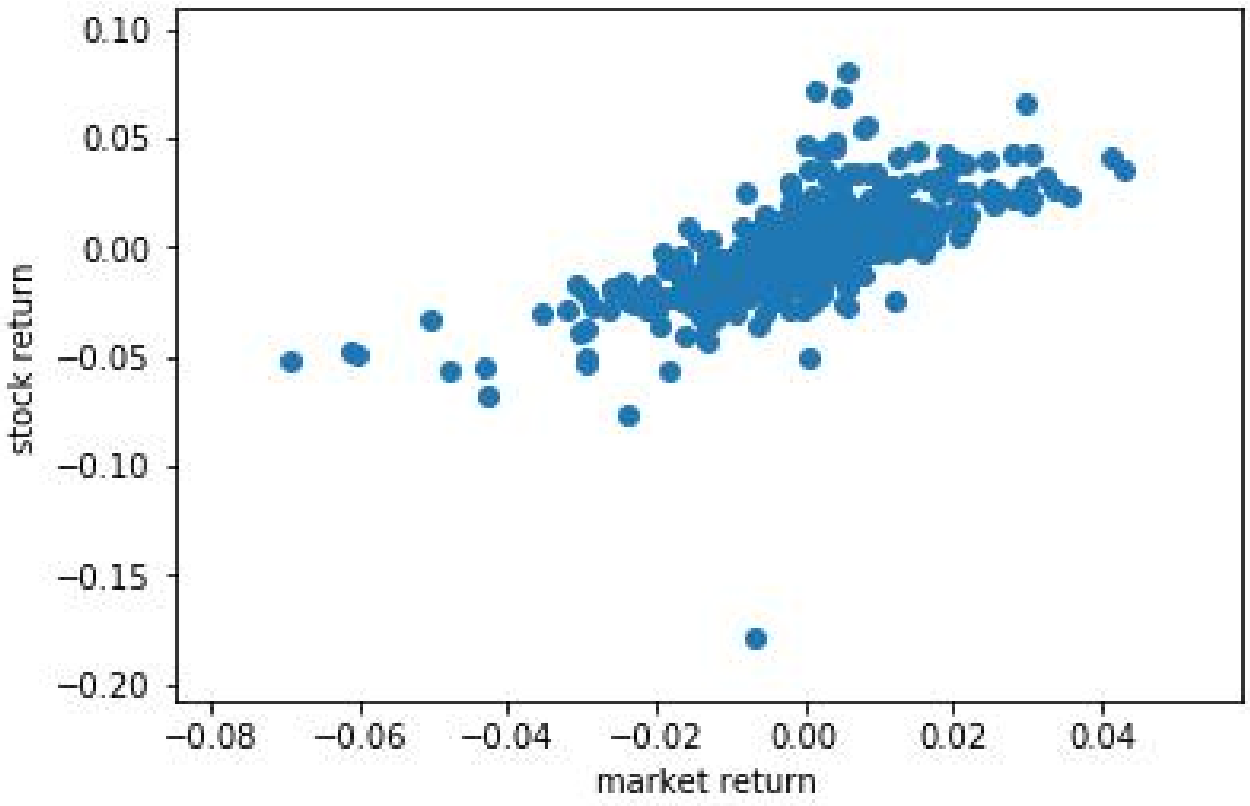
线性模型

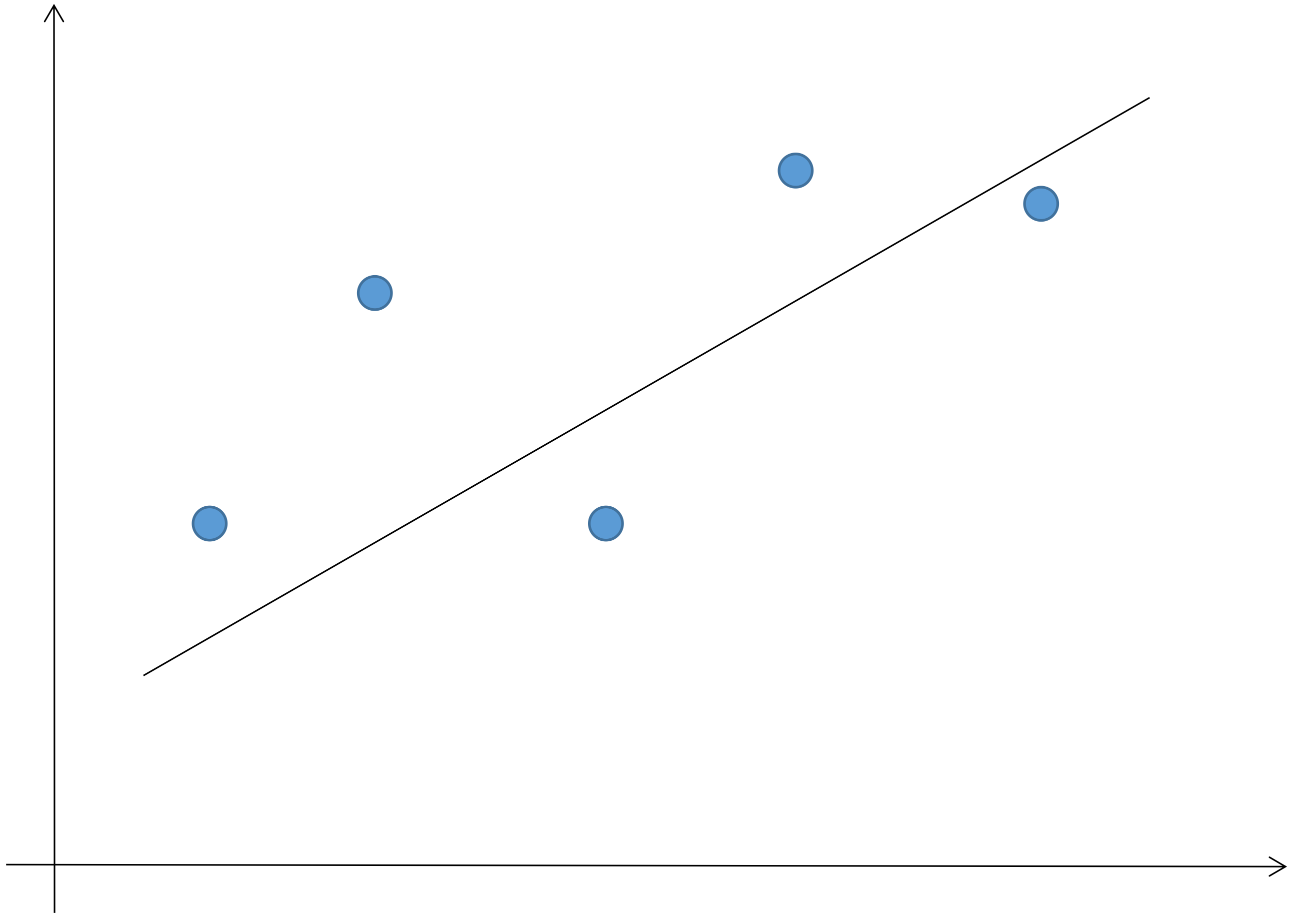
一元线性回归

计算股票的 beta



stock return = a + beta * market return

一般情形



$$y = w_0 + w_1 x$$

loss function

$$= \sum \left[y_i - \left(w_0 + w_1 x_i \right) \right]^2$$

一般情形

$$\min L(w_0, w_1) = \sum\left[y_i - (w_0 + w_1 x_i)\right]^2$$

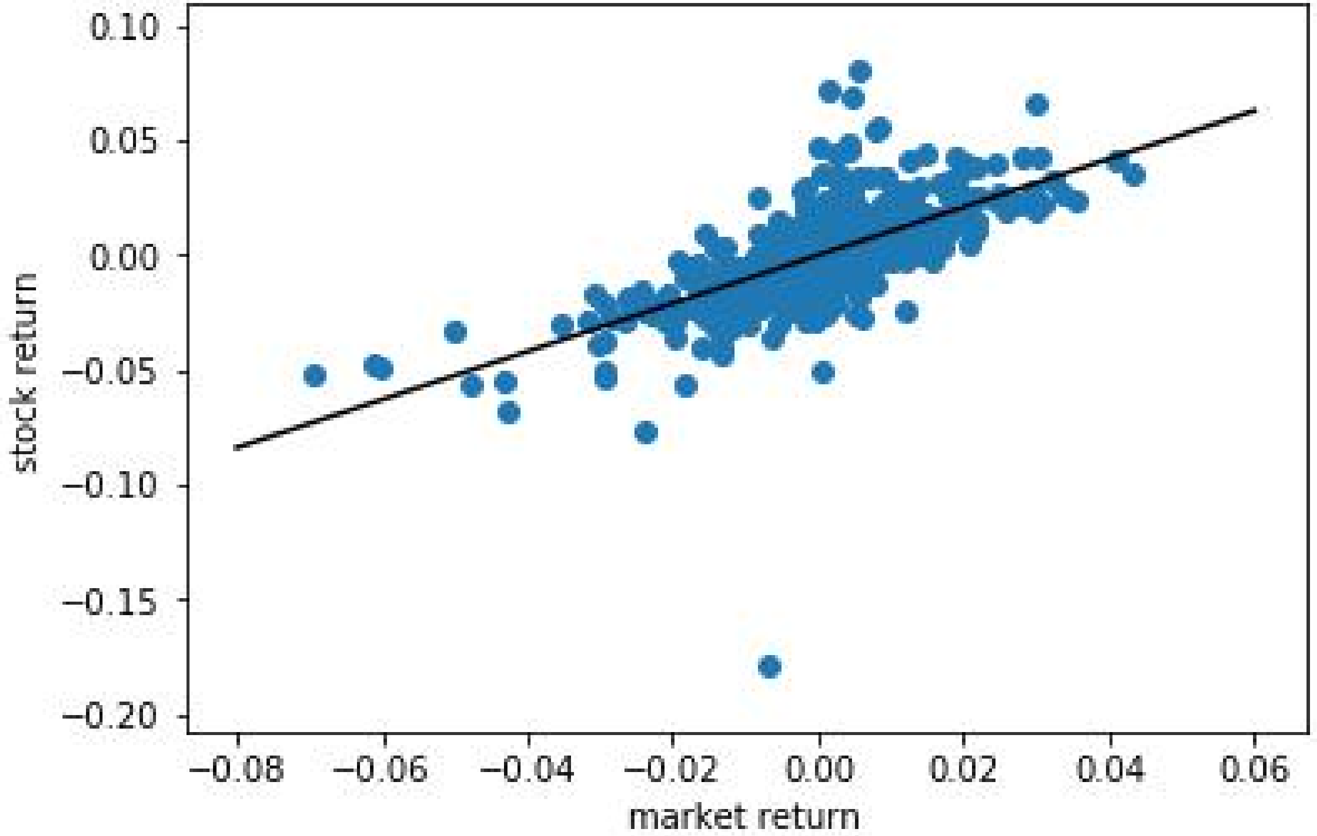
$$\frac{\partial L}{\partial w_0} = -2\sum\left(y_i - w_0 - w_1 x_i\right) = 0$$

$$\Rightarrow \boxed{w_0 = \bar{y} - w_1 \bar{x}}$$

$$\frac{\partial L}{\partial w_1} = -2\sum\left(y_i - w_0 - w_1 x_i\right)x_i = 0$$

$$\Rightarrow w_1 = \frac{\sum x_i y_i - \bar{x}\bar{y}}{\sum x_i^{\,2} - (\bar{x})^2} = \frac{\operatorname{cov}(x, y)}{Var(x)}$$

计算股票的 beta



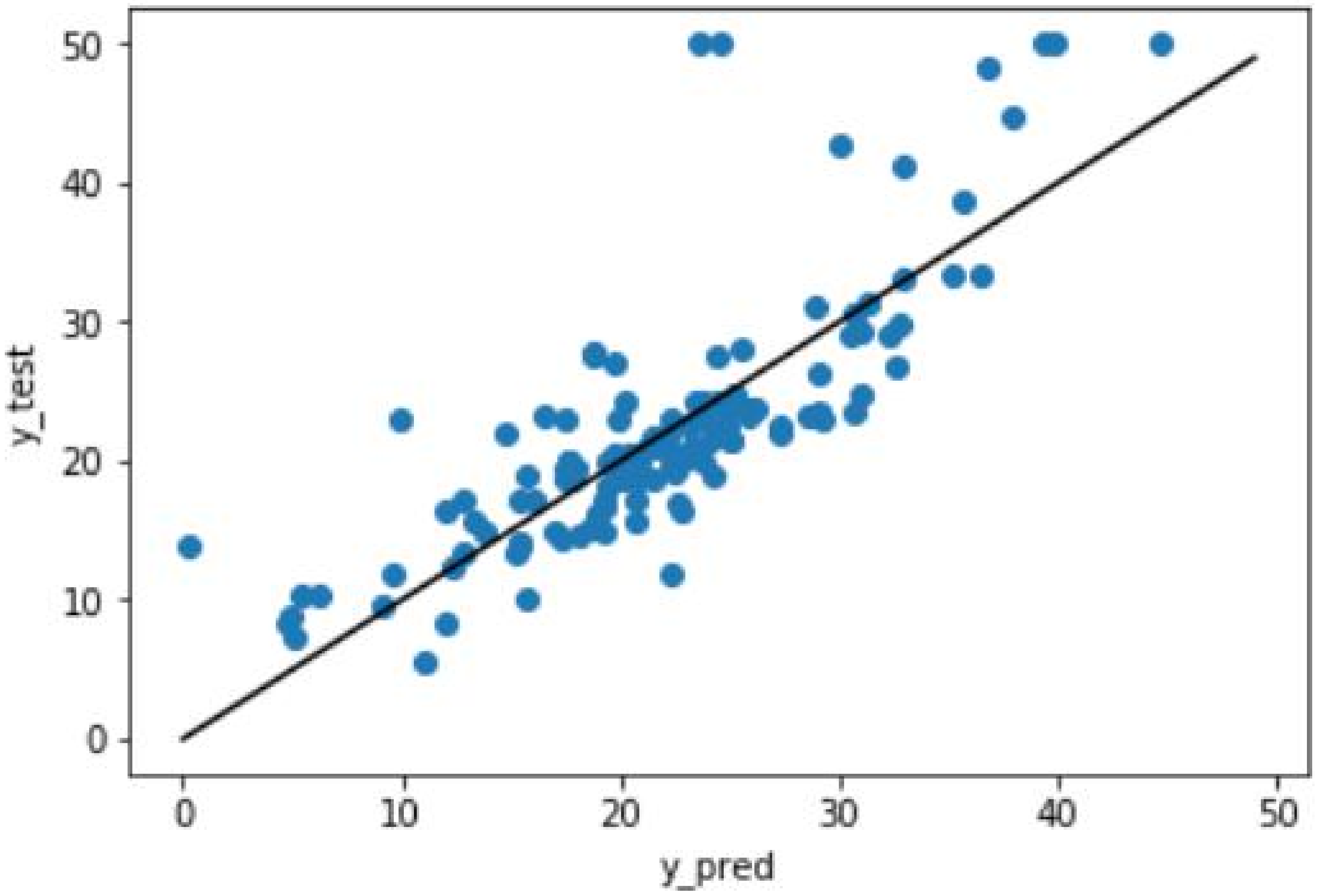
stock return = 4.166e-05 + 1.048 market return

多元线性回归

预测 Boston 房价

```
**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None
```

预测 Boston 房价



coef	feature
36.9805	INTER
-0.11687	CRIM
0.0439939	ZN
-0.00534808	INDUS
2.39455	CHAS
-15.6298	NOX
3.76145	RM
-0.00695007	AGE
-1.4352	DIS
0.239756	RAD
-0.0112937	TAX
-0.986626	PTRATIO
0.00855688	B
-0.500029	LSTAT

一般情形

$$y = w_0 + w_1 x_1 + \ldots + w_d x_d$$

$$\text{loss function } = \sum \left[y_i - \left(w_0 + w_1 x_{i1} + \ldots + w_d x_{id} \right) \right]^2$$

使用矩阵形式

$$y=\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix}, X=\begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1d} \\ 1 & x_{21} & x_{22} & \cdots & x_{2d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{m1} & x_{m2} & \cdots & x_{md} \end{bmatrix}, \quad w=\begin{bmatrix} 1 \\ w_1 \\ w_2 \\ \cdots \\ w_d \end{bmatrix}$$

$$L(w)=\left(y-Xw\right)^T\left(y-Xw\right)$$

求解线性回归

$$\min : L(w) = (y - Xw)^T (y - Xw)$$

- Normal Equation

$$\mathrm{w}^* = \left(X^T X\right)^{-1} X^T y$$

- Gradient Descent

$$w := w - \alpha X^T \left(Xw - y\right)$$

Normal Equation (Optional)

$$\frac{\partial L(w)}{\partial w} = 2\left(Xw - y\right)^T X = 0$$

$$\Downarrow$$

$$\mathrm{w}^* = \left(X^T X\right)^{-1} X^T y$$

$$Note: \frac{\partial^2 L(w)}{\partial w \partial w^{\mathrm{T}}} = 2X^T X$$

$$\frac{\partial Ax}{\partial x} = A$$

$$\frac{\partial x^T Ax}{\partial x} = x^T (A + A^T)$$

不可逆的问题 （Optional）

(X'X) 不可逆的情形：

- **冗余特征：**
  E.g.   x1 = 交易量（股数）
          x2 = 交易量（手数）

- **特征太多：**
  E.g.   特征个数 >= 样本量

解决方法：

- 删除某些特征
- 使用 regularization

Gradient Descent (Optional)

$$\min : L(w) = (y - Xw)^T (y - Xw)$$

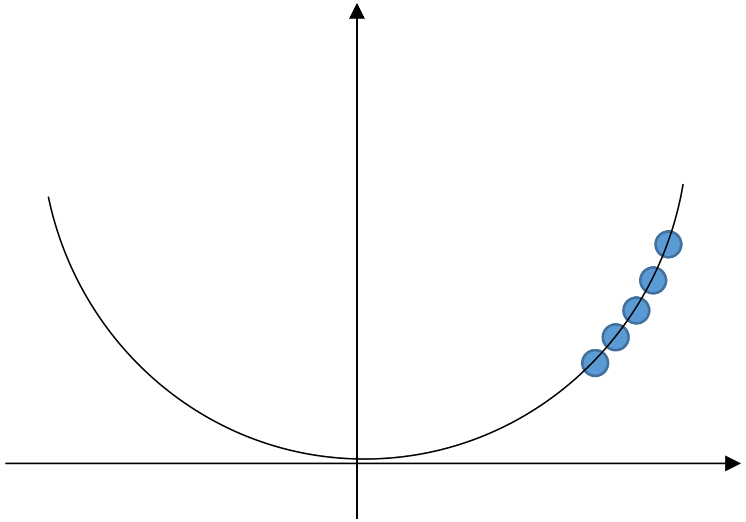
$$\frac{\partial L(w)}{\partial w} = 2(Xw - y)^T X$$

$$w := w - \alpha X^T (Xw - y)$$

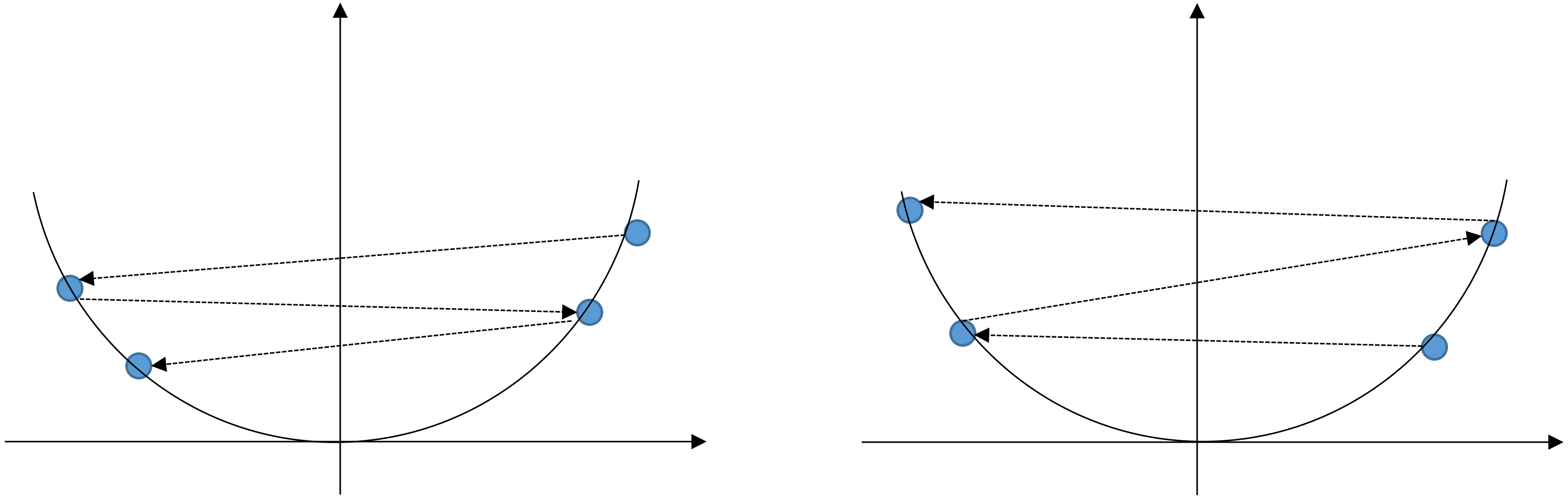
Learning Rate（Optional）

$$w := w - \alpha X^T \left(Xw - y \right)$$

Learning rate too small:
收敛速度慢



Learning rate too large:
Overshoot



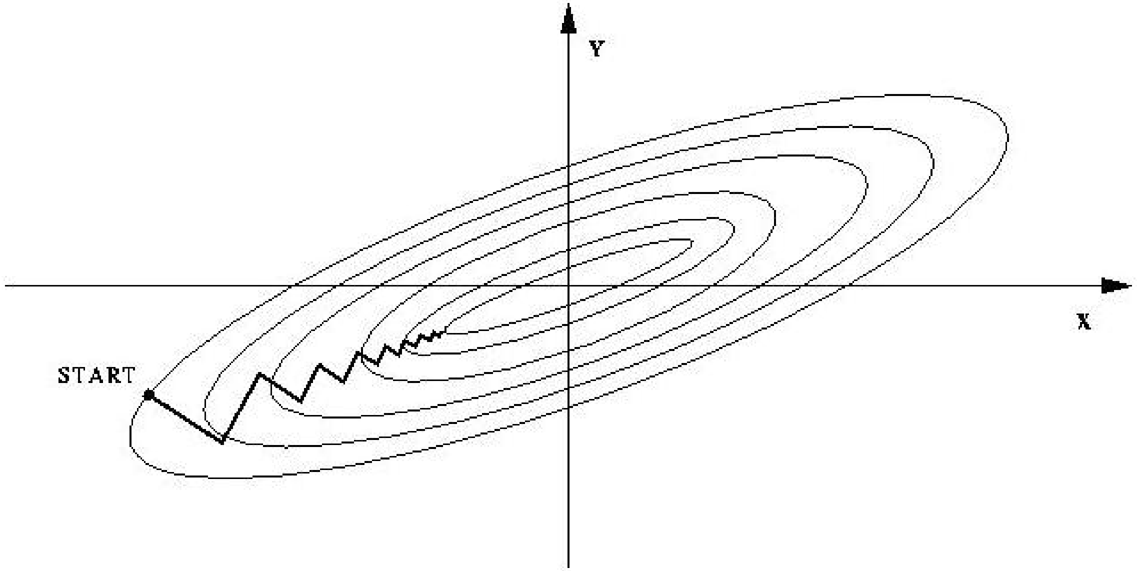
Normal Equation v.s. Gradient Descent （Optional）

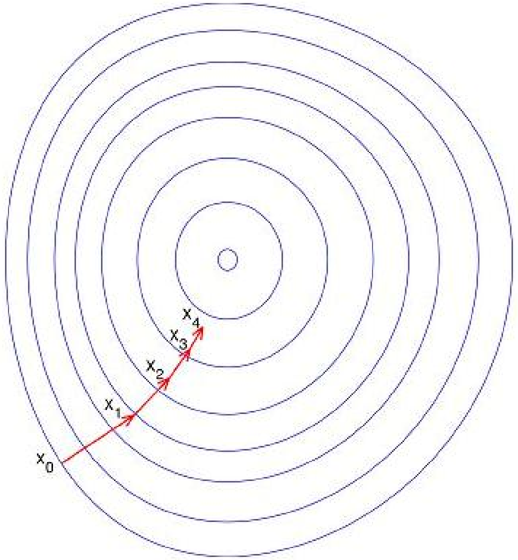
	Normal Equation	Gradient Descent
优点	不需要 alpha 参数 不需要迭代	适合较大的样本量
缺点	需要进行矩阵求逆 当样本量较大时速度较慢	需要 alpha 参数 需要进行迭代

Feature Scaling

统一特征值的数量级，有助于提高运算效率：

- 乘数
- MinMaxScaler
- StandardScaler





LinearRegression

sklearn.linear_model.LinearRegression

关键参数：--

惩罚回归

特征较多的回归

考察一个含有 104 个特征的线性回归

$$y = w_0 + w_1 x_1 + \ldots + w_{104} x_{104}$$

training score: 0.9523526436864238
test score: 0.6057754892935543

test score << training score **极有可能存在 overfitting**

惩罚回归

$$L(w) = (y - Xw)^T (y - Xw) + \lambda \sum_{\mathrm{i}} \left[(1-\alpha)|w_i| + \alpha |w_i|^2 \right]$$

$\lambda = 0,$ 普通线性回归

$\lambda > 0, \quad \alpha = 1,$ 岭回归（ridge regression)

$\lambda > 0, \quad \alpha = 0,$ Lasso (least absolute shrinkage and selection operator)

Ridge Regression --- Normal Equation （Optional）

$$L(w)$$

$$=(y-Xw)^T(y-Xw)+\lambda\sum_{\mathrm{i}} w_i^{\,2}$$

$$=(y-Xw)^T(y-Xw)+\lambda w^{\mathrm{T}}w$$

$$\frac{\partial L(w)}{\partial w}=2(Xw\text{-}y)^T X+2\lambda w^{\mathrm{T}}=0$$

当λ较大时，$X^TX+\lambda I$可保证正定

$$\Rightarrow w^*=(X^TX+\lambda I)^{-1}X^Ty$$

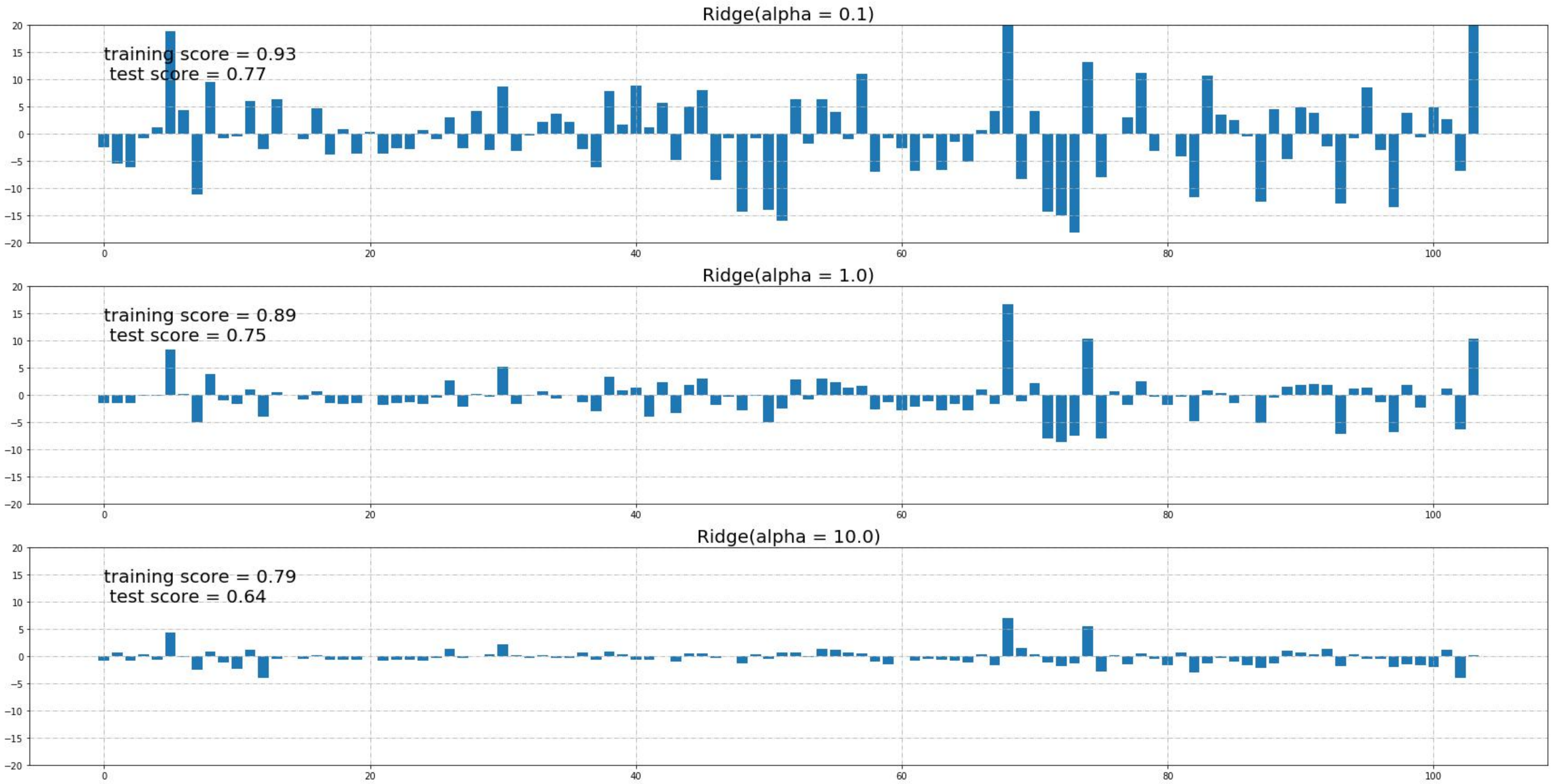
Ridge Regression --- Gradient Descent (Optional)

$$\frac{\partial L(w)}{\partial w} = 2(Xw - y)^T X + 2\lambda w^{\mathrm{T}}$$

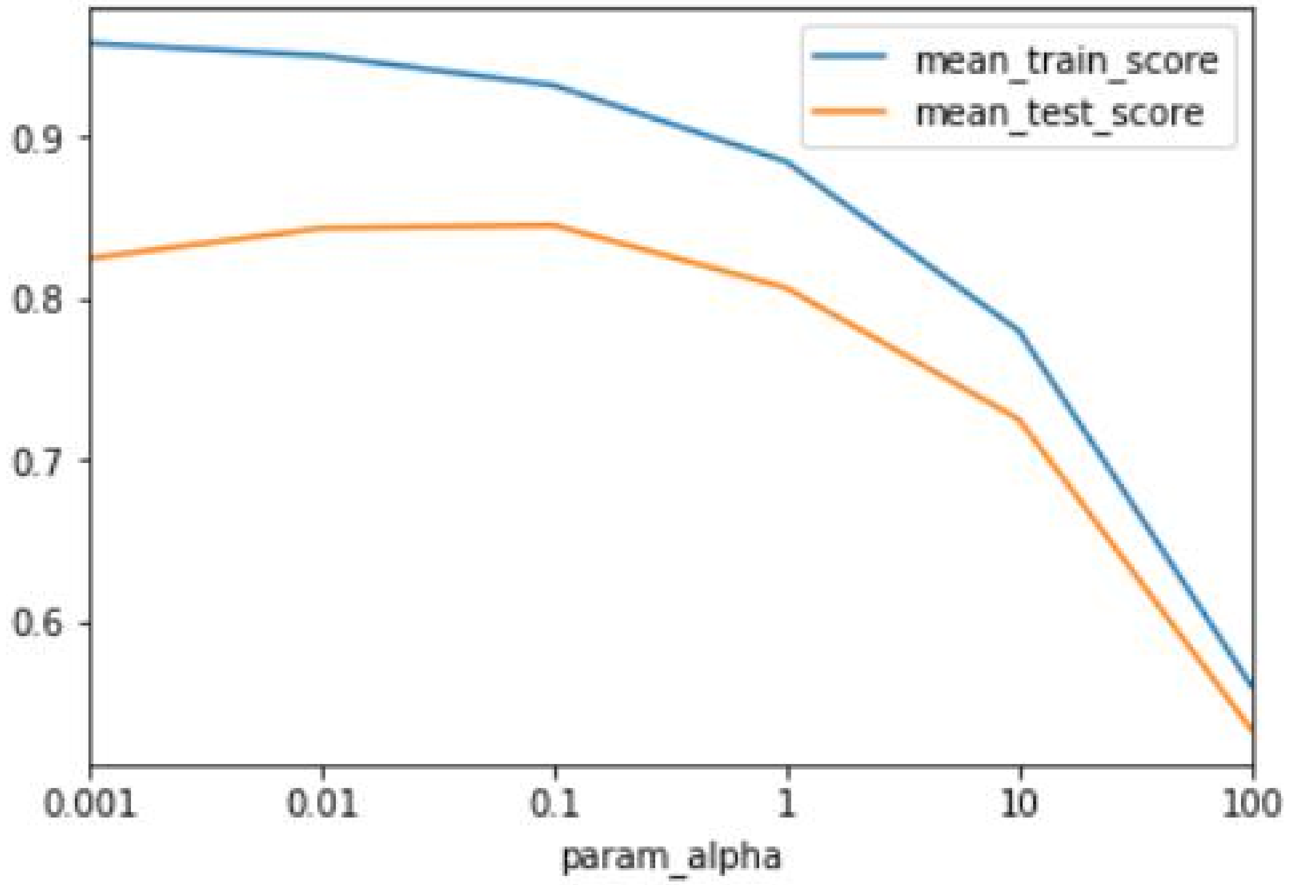
$$w := w - \alpha\left[X^T(Xw - y) + w\right]$$

$$= \underline{(1-\alpha)}w - \alpha X^T(Xw - y)$$

使用 Ridge Regression



Grid Search for Alpha



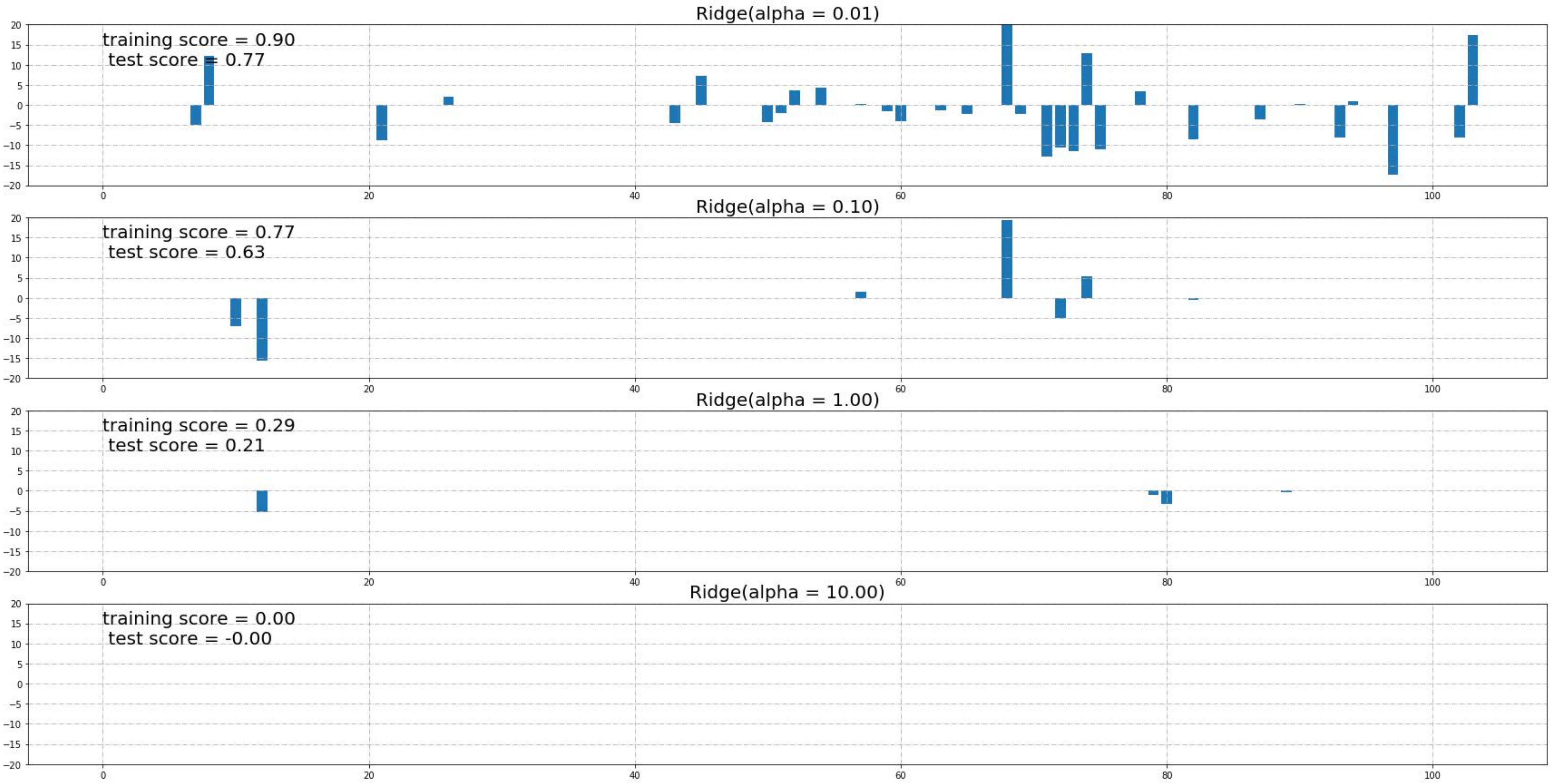
Lasso（Optional)

$$L(w)$$

$$=(y-Xw)^T(y-Xw)+\lambda\sum_{\mathrm{i}}|w_i|$$

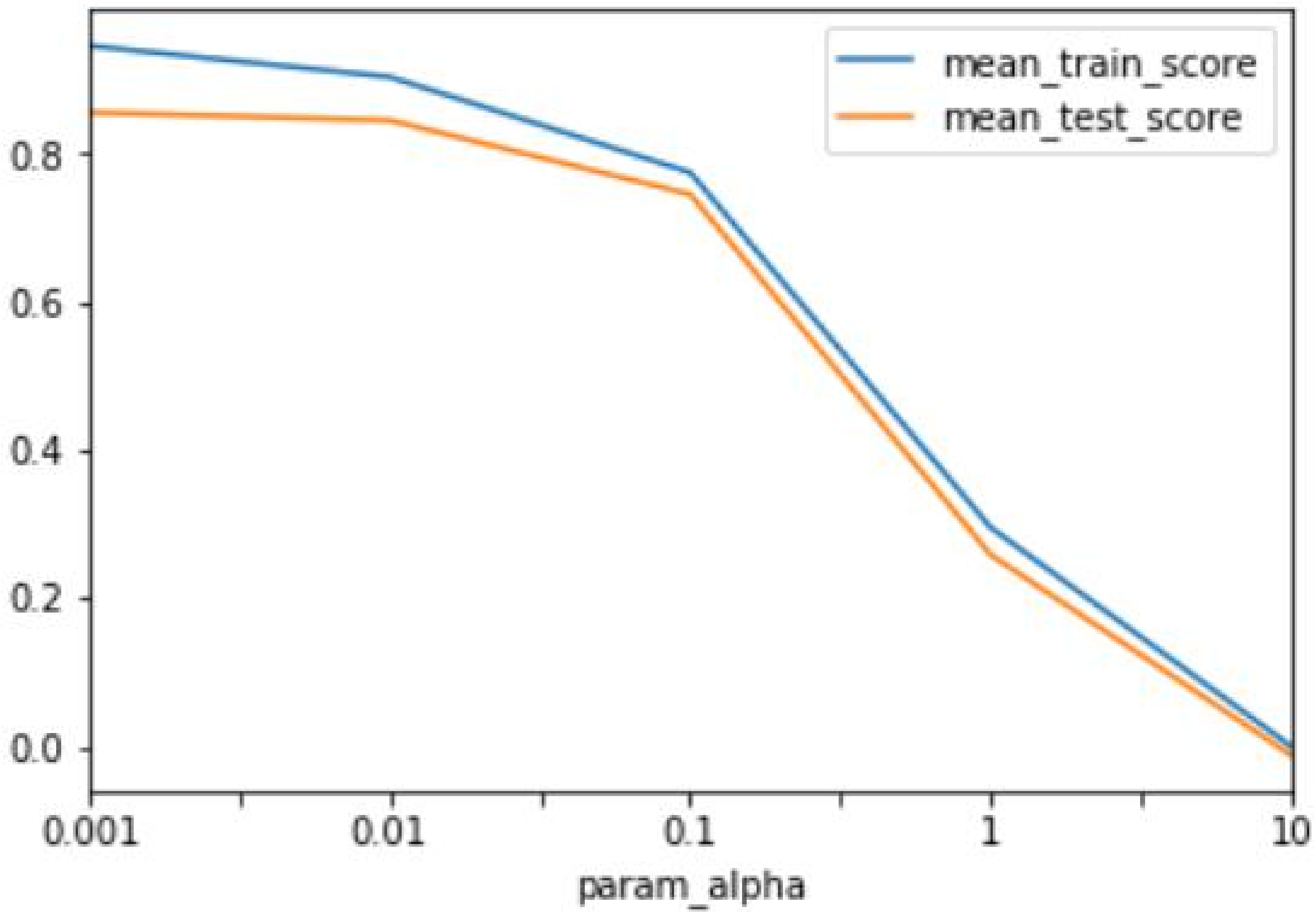
$$\frac{\partial L(w)}{\partial w}=? \longrightarrow \text{无解析表达式}$$

使用 Lasso



Ridge(alpha = 0.01)
training score = 0.90
test score = 0.77
Ridge(alpha = 0.10)
training score = 0.77
test score = 0.63
Ridge(alpha = 1.00)
training score = 0.29
test score = 0.21
Ridge(alpha = 10.00)
training score = 0.00
test score = -0.00

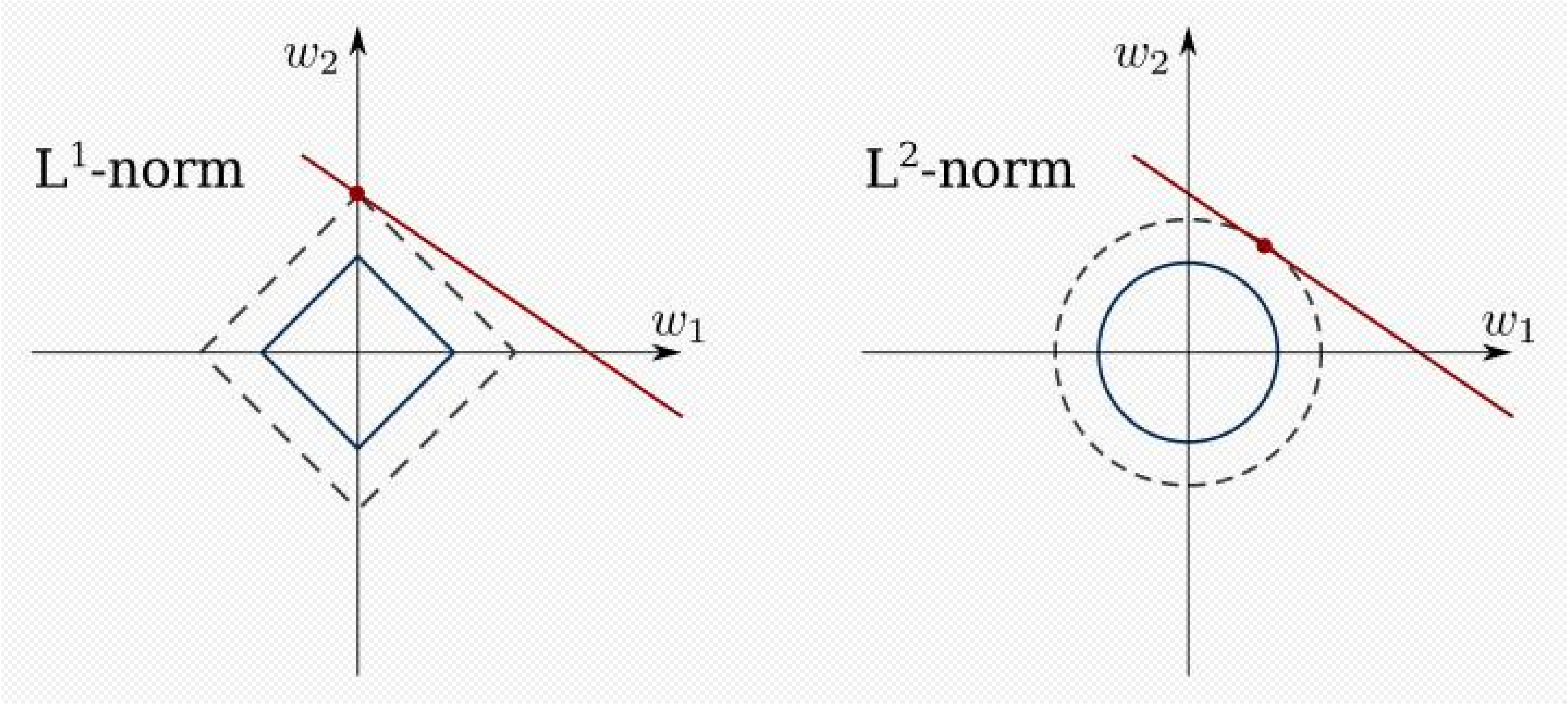
Grid Search for Alpha



Ridge v.s. Lasso

	Ridge	Lasso
算法复杂性	低	高
参数稳定性	高	低
可用于 feature selection	-	√

为什么 Lasso 会使得系数等于零



使用 Ridge 和 Lasso

sklearn.linear_model.Ridge
sklearn.linear_model.Lasso

关键参数：

- alpha，alpha 越大，惩罚力度越大

Logistic Regression

乳腺癌的诊断

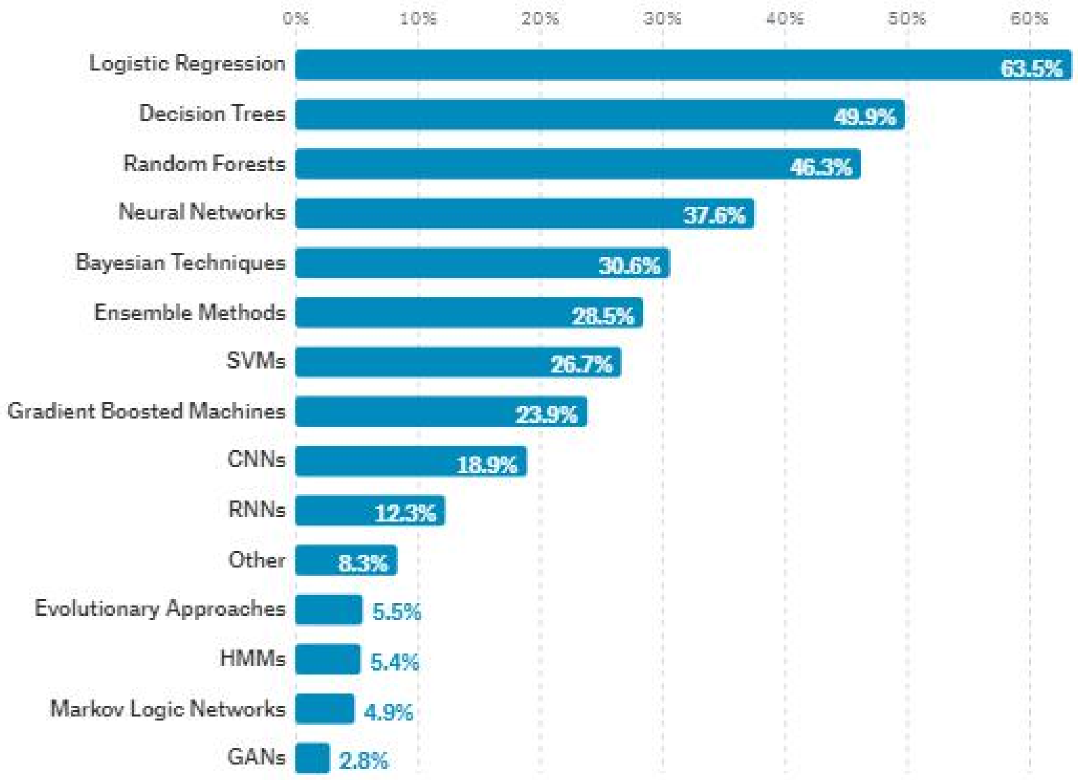
```
Data Set Characteristics:
    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:AA
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)
```

Logistic Regression

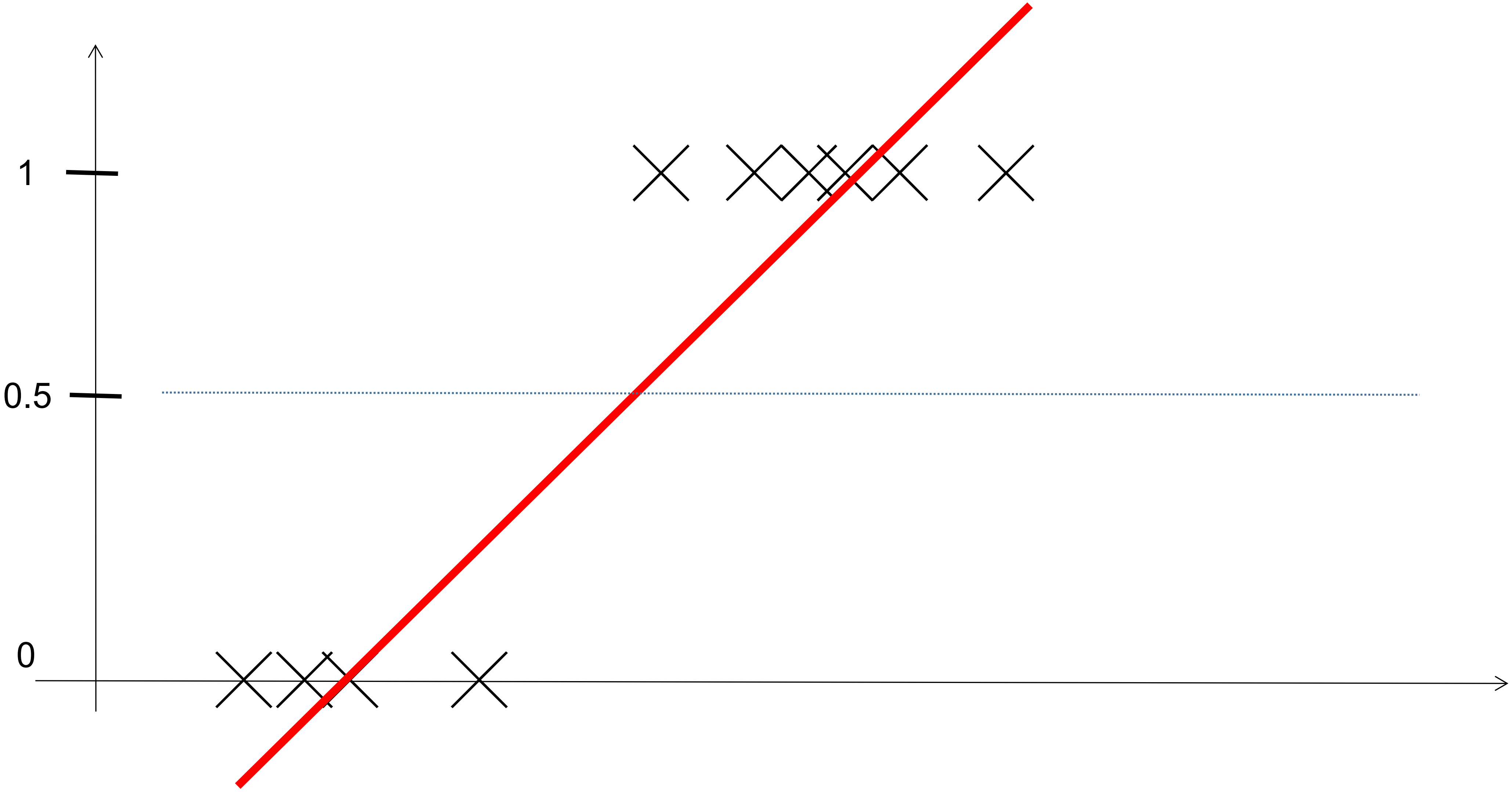
What data science methods are used at work?

Logistic regression is the most commonly reported data science method used at work for all industries *except* Military and Security where Neural Networks are used slightly more frequently.

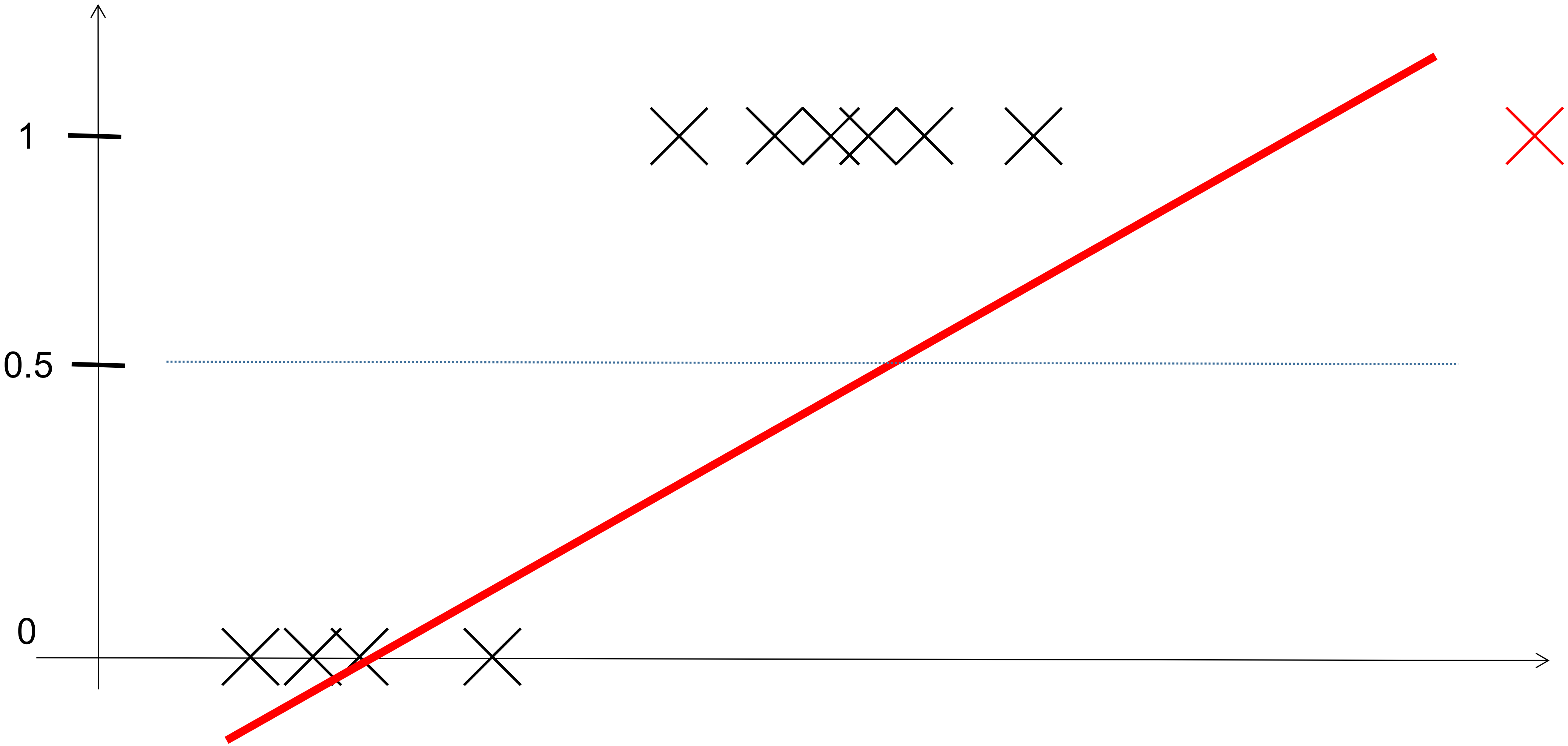
Company ▼ Industry ▼ Job Title ▼



为什么线性回归不能用于分类问题



为什么线性回归不能用于分类问题



Logistic Regression

$$z = w_0 + w_1 x$$

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = P(\text{正类})$$

sigmoid function or **logistic function**

确定损失函数（Optional）

Loss function of linear regression

$$\sum\left[y_i - \left(w_0 + w_1 x_i\right)\right]^2$$

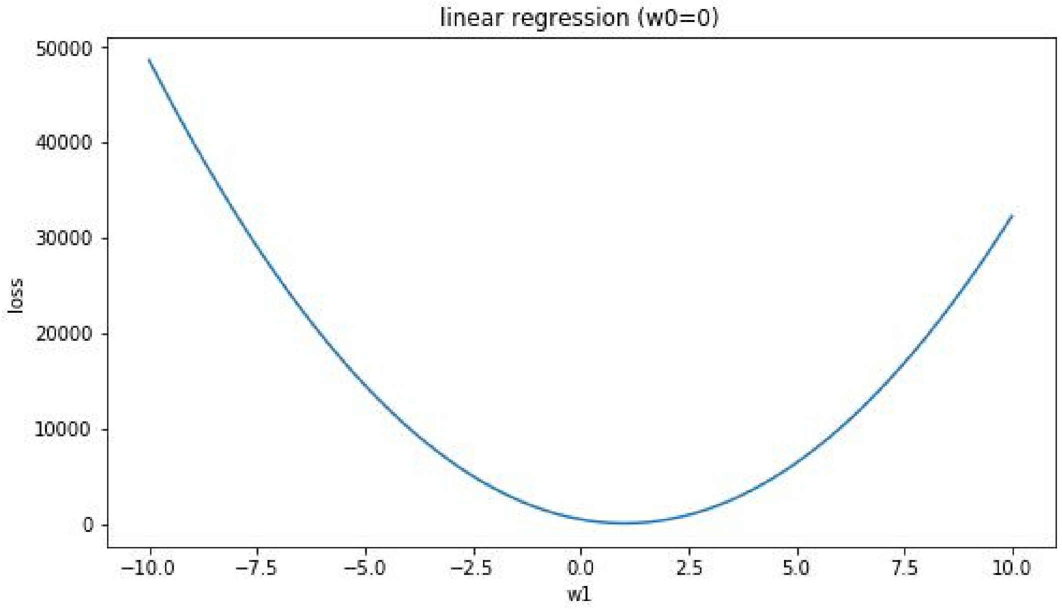
Loss function of logistic regression

$$\sum\left[y_i - \frac{1}{1+e^{-\left(w_0+w_1 x_i\right)}}\right]^2 \quad ?$$

确定损失函数（Optional）

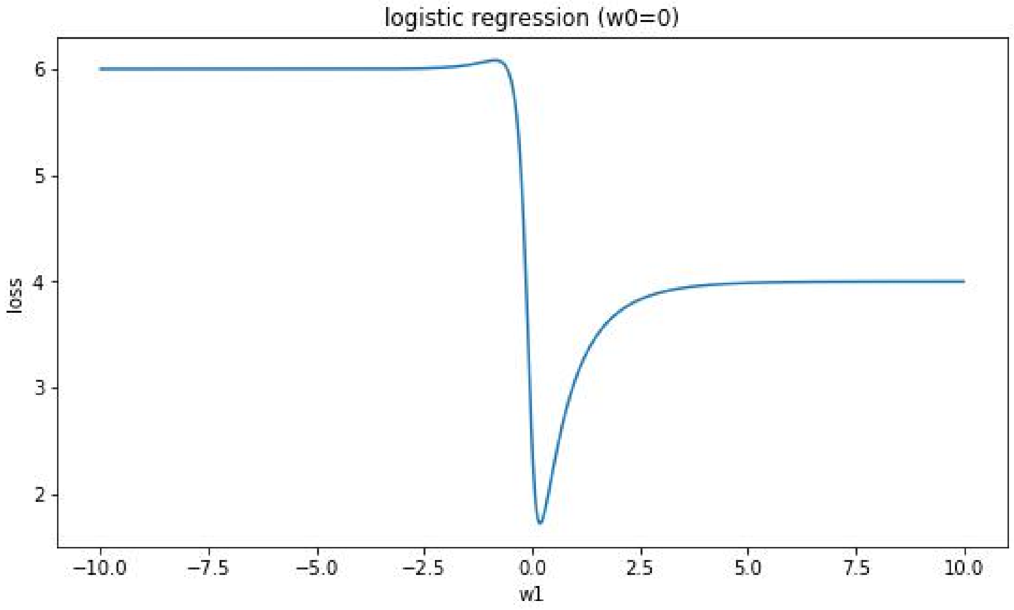
Loss function of linear regression

$$L = \sum \left[y_i - \left(w_0 + w_1 x_i \right) \right]^2$$



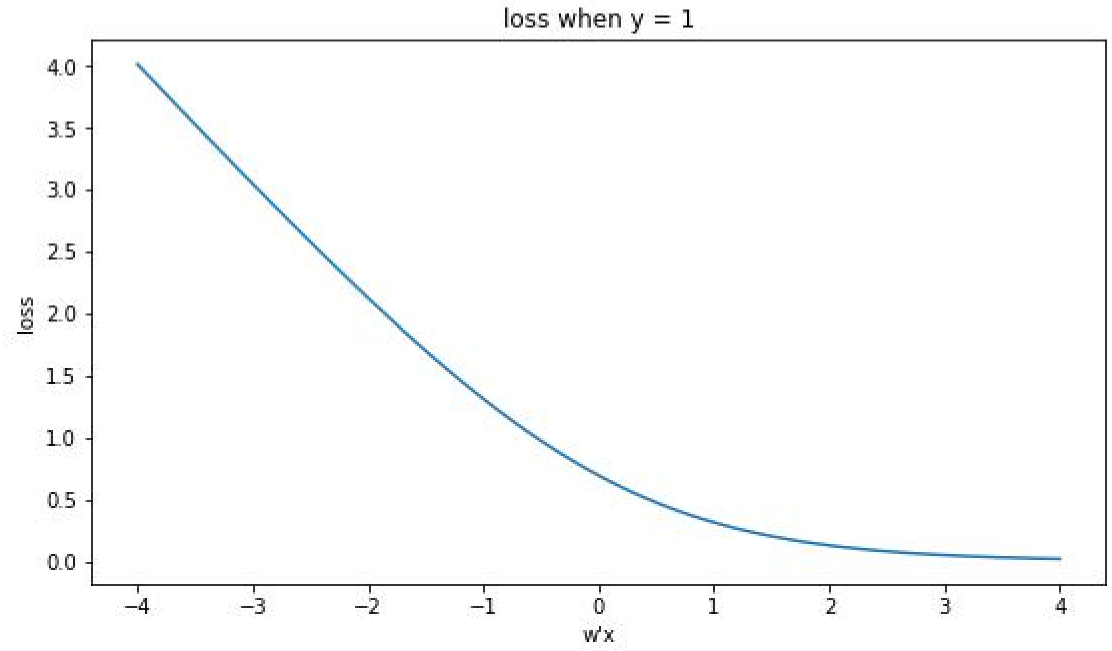
Loss function of logistic regression ?

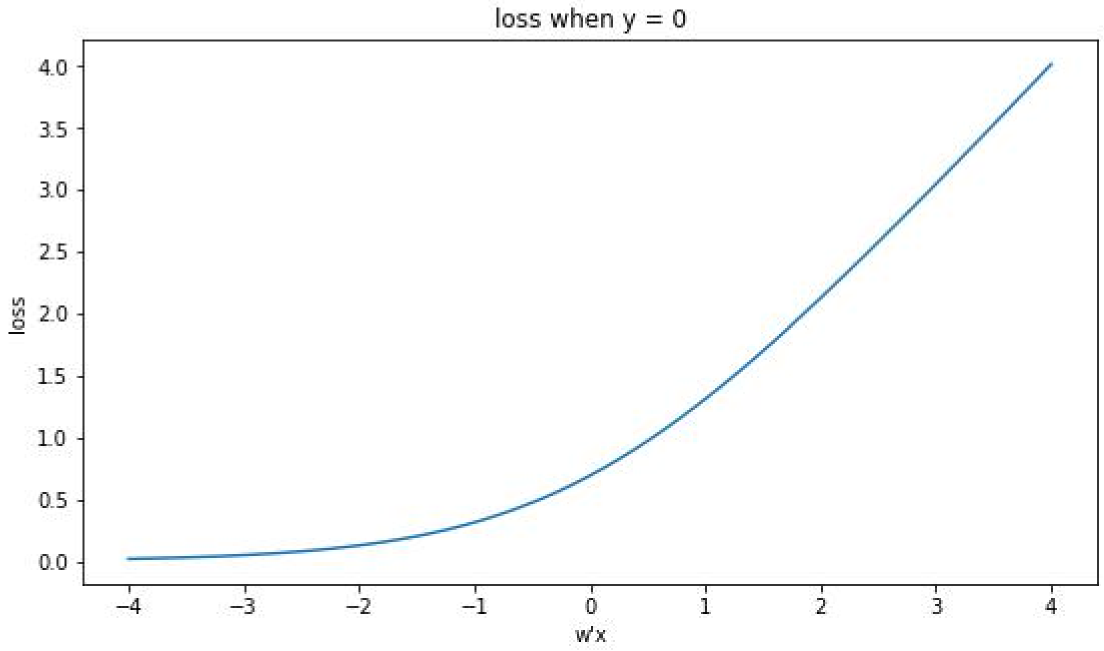
$$L = \sum \left[y_i - \frac{1}{1 + e^{-\left(w_0 + w_1 x_i \right)}} \right]^2$$



确定损失函数（Optional）

$$Loss_i = \begin{cases} -\log\left[\sigma\left(w^T x_i\right)\right], & y_i = 1 \\ -\log\left(1 - \left[\sigma\left(w^T x_i\right)\right]\right), & y_i = 0 \end{cases}$$





$y_i = 1$，want $w^T x_i \to +\infty \Leftrightarrow \sigma(x_i) \to 1$

$y_i = 0$，want $w^T x_i \to -\infty \Leftrightarrow \sigma(x_i) \to 0$

确定损失函数

Loss function of linear regression

$$\sum \left[y_i - \left(w_0 + w_1 x_i\right)\right]^2$$

Loss function of logistic regression

$$\sum \left[- y_i \log\left(\sigma\left(w^T x_i\right)\right) - \left(1 - y_i\right)\log\left(1 - \sigma\left(w^T x_i\right)\right)\right]$$

Gradient Descent (Optional)

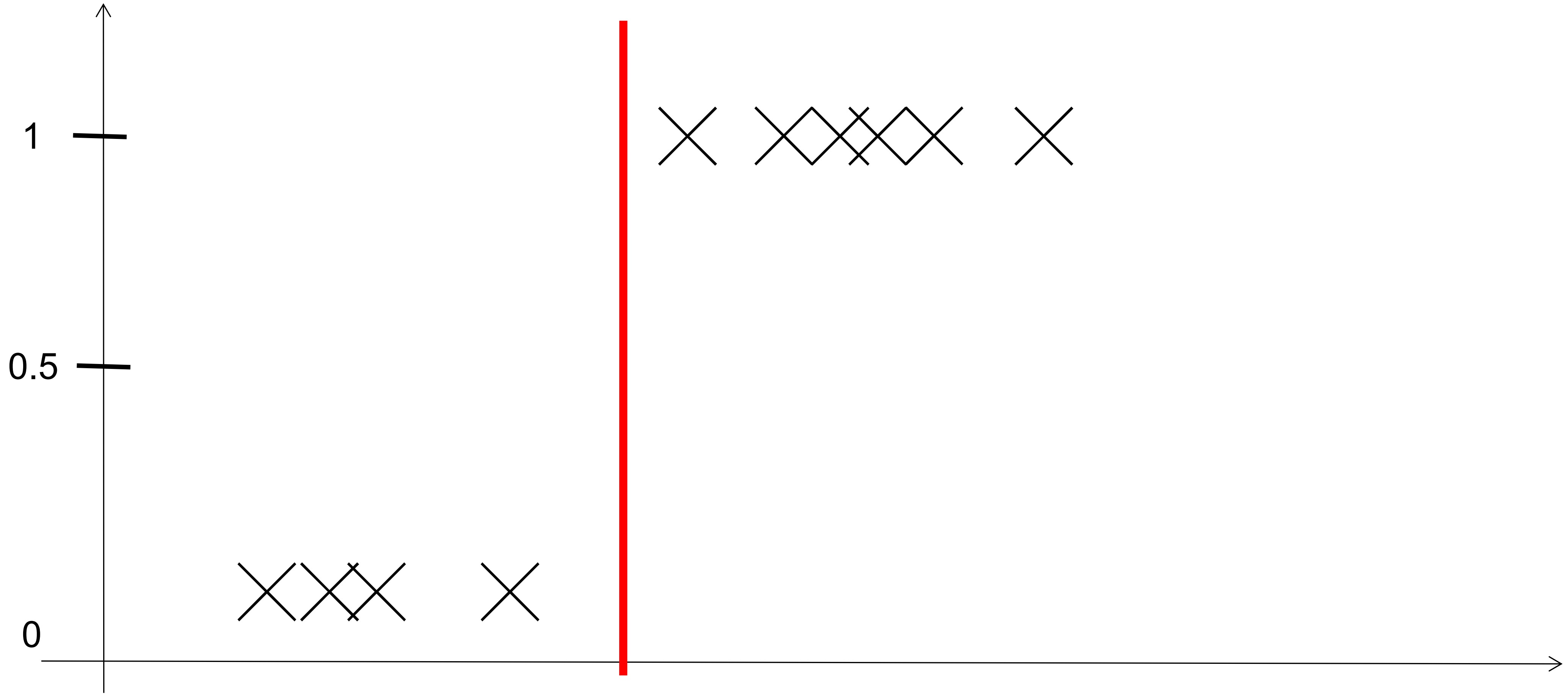
$$L(w) = \sum \left[-y_i \log\left(\sigma\left(w^T x_i\right)\right) - \left(1 - y_i\right) \log\left(1 - \sigma\left(w^T x_i\right)\right) \right]$$

$$\frac{\partial L(w)}{\partial w} = -\sum \left(y_i - \sigma\left(w^T x_i\right)\right) x_i^{\ T}$$

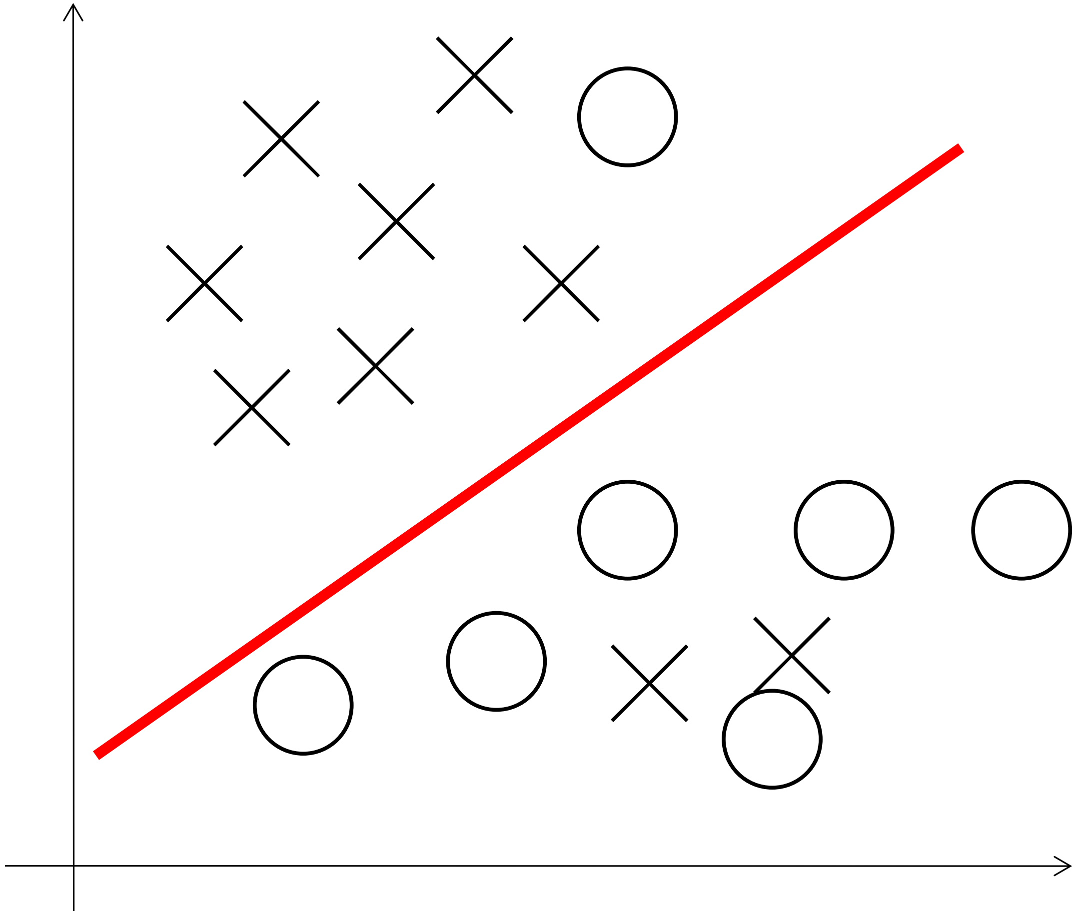
$$w := w + \alpha \sum \left(y_i - \sigma\left(w^T x_i\right)\right) x_i$$

决策边界（Decision Boundary)

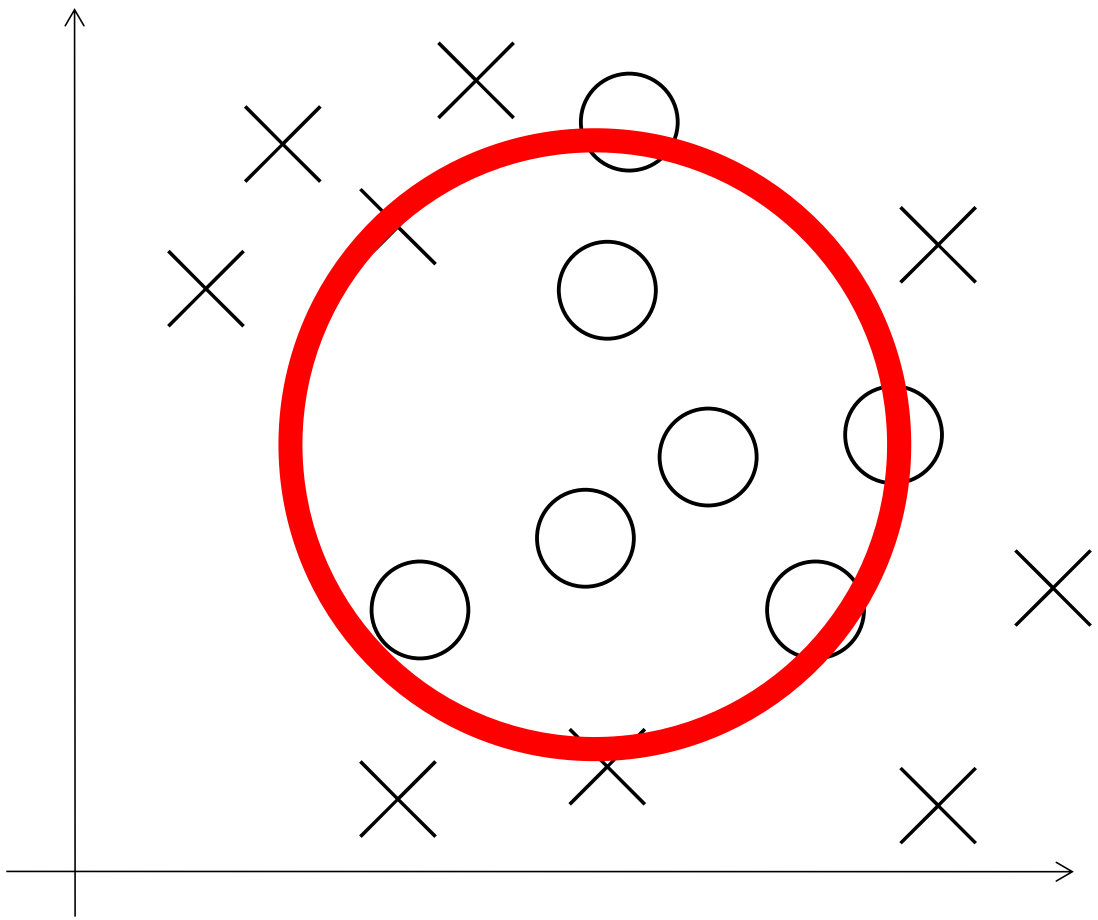
$$\sigma(z) \geq 0.5, \quad z \geq 0$$
$$\sigma(z) < 0.5, \quad z < 0$$



决策边界（Decision Boundary)



决策边界（Decision Boundary)



交叉熵（Cross Entropy）与损失函数

交叉熵

p, q 是两个概率分布，定义**交叉熵**

$$H(p,q) = -\sum p_i log(q_i)$$

交叉熵衡量了两个分布之间的 **“距离”**

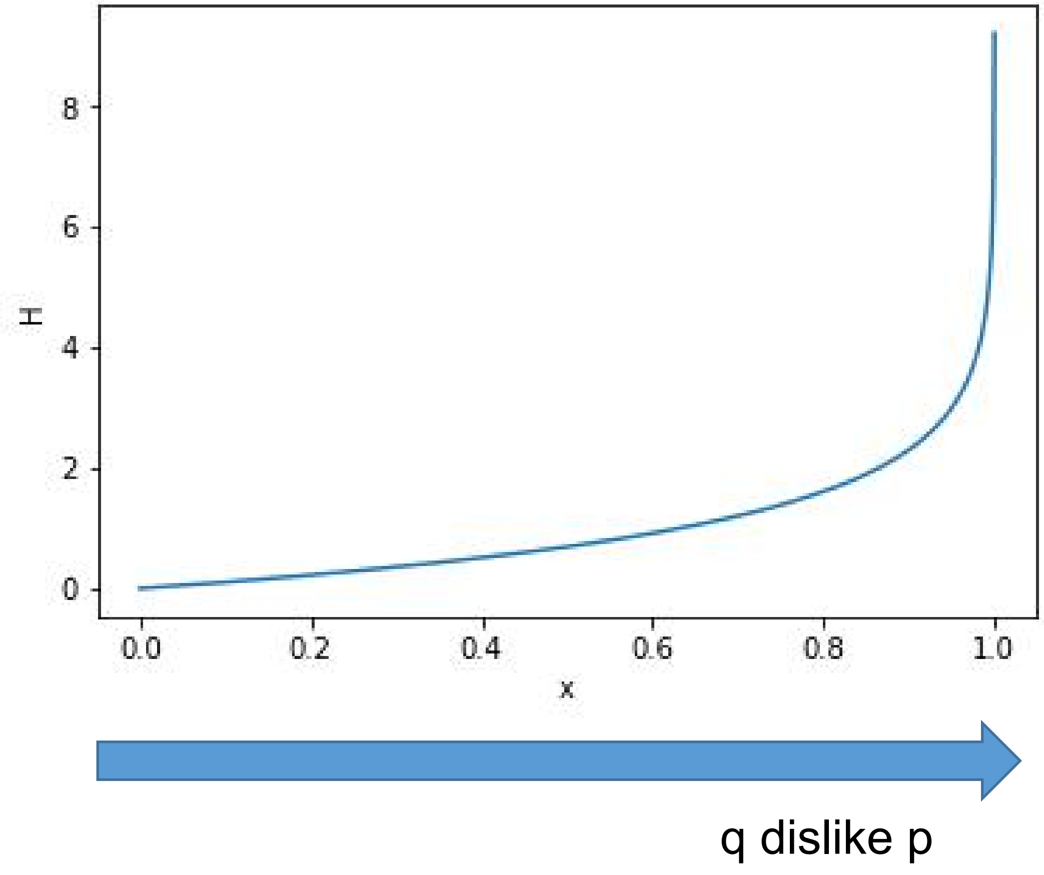
交叉熵

$$\min H(p,q) = -\sum p_i log(q_i)$$

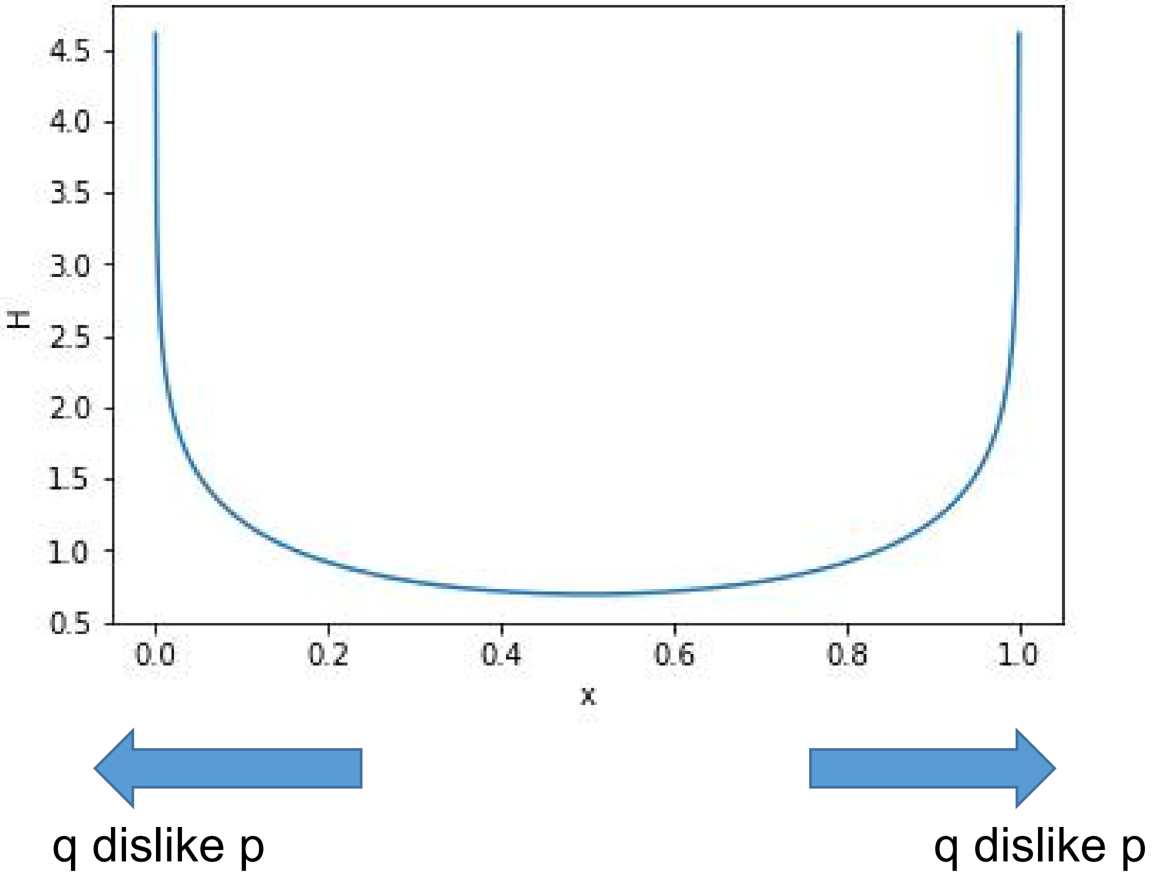
$\rightarrow p_i = q_i$

交叉熵

p: [0,1]
q: [x,1-x]



p: [0.5,0.5]
q: [x,1-x]



交叉熵

真实分布	预测分布	
[1, 0]	$\left[1-\sigma\left(w^T x_i\right), \sigma\left(w^T x_i\right)\right]$	$H_i = -\log\left[1-\sigma\left(w^T x_i\right)\right]$
[0, 1]	$\left[1-\sigma\left(w^T x_i\right), \sigma\left(w^T x_i\right)\right]$	$H_i = -\log\left[\sigma\left(w^T x_i\right)\right]$
...	...	...

yi

$$\sum H_i = -\sum\left(y_i \log\left[\sigma\left(w^T x_i\right)\right] + (1-y_i)\log\left[1-\sigma\left(w^T x_i\right)\right]\right)$$

= Loss Function

Logistic Regression 调参

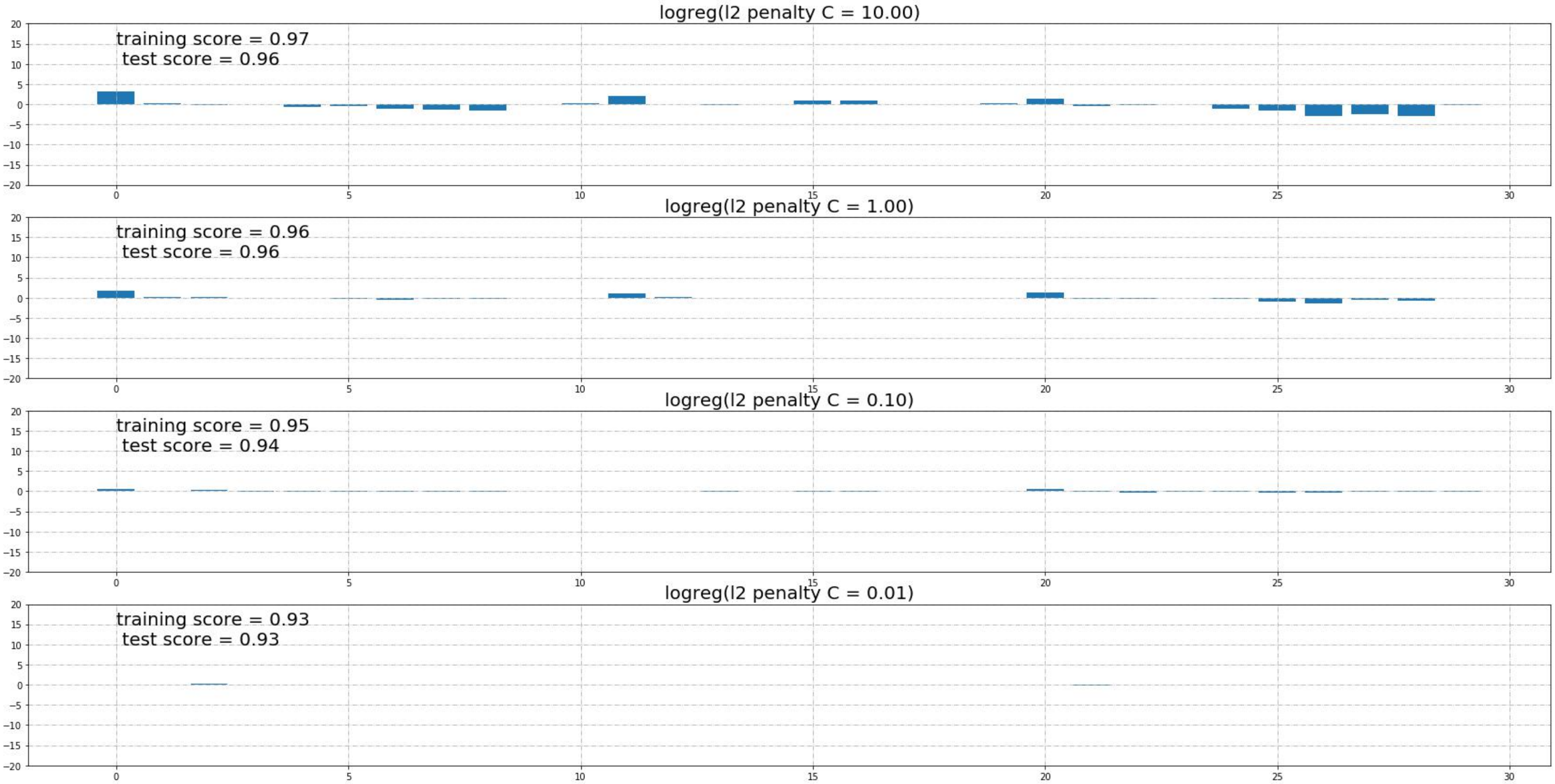
使用 Logistic Regression

sklearn.linear_model.LogisticRegression

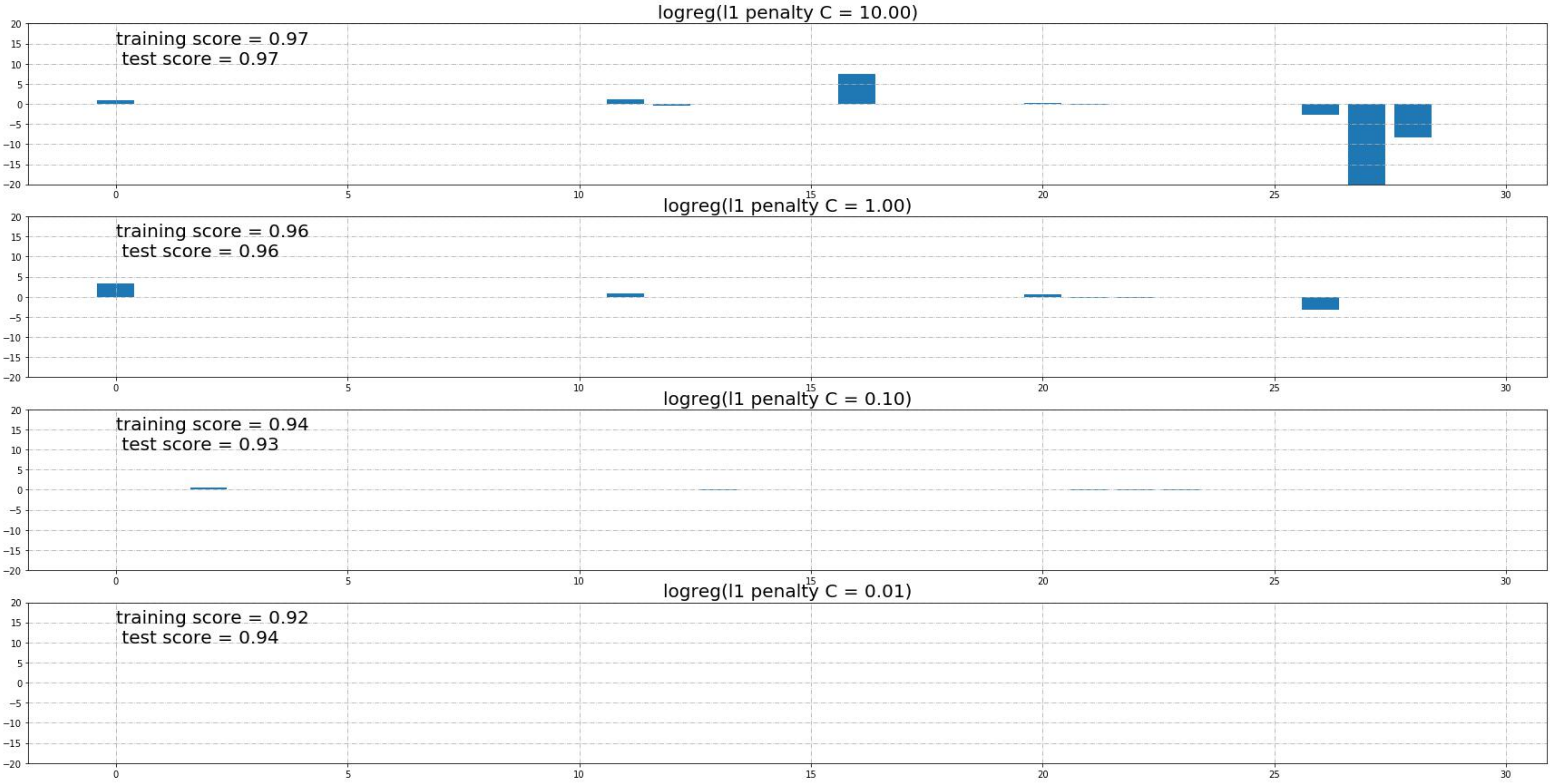
关键参数:

- penalty = 'l1' / 'l2', default: ‘l2’
- C : float, default: 1.0， **C 越小，惩罚越大**

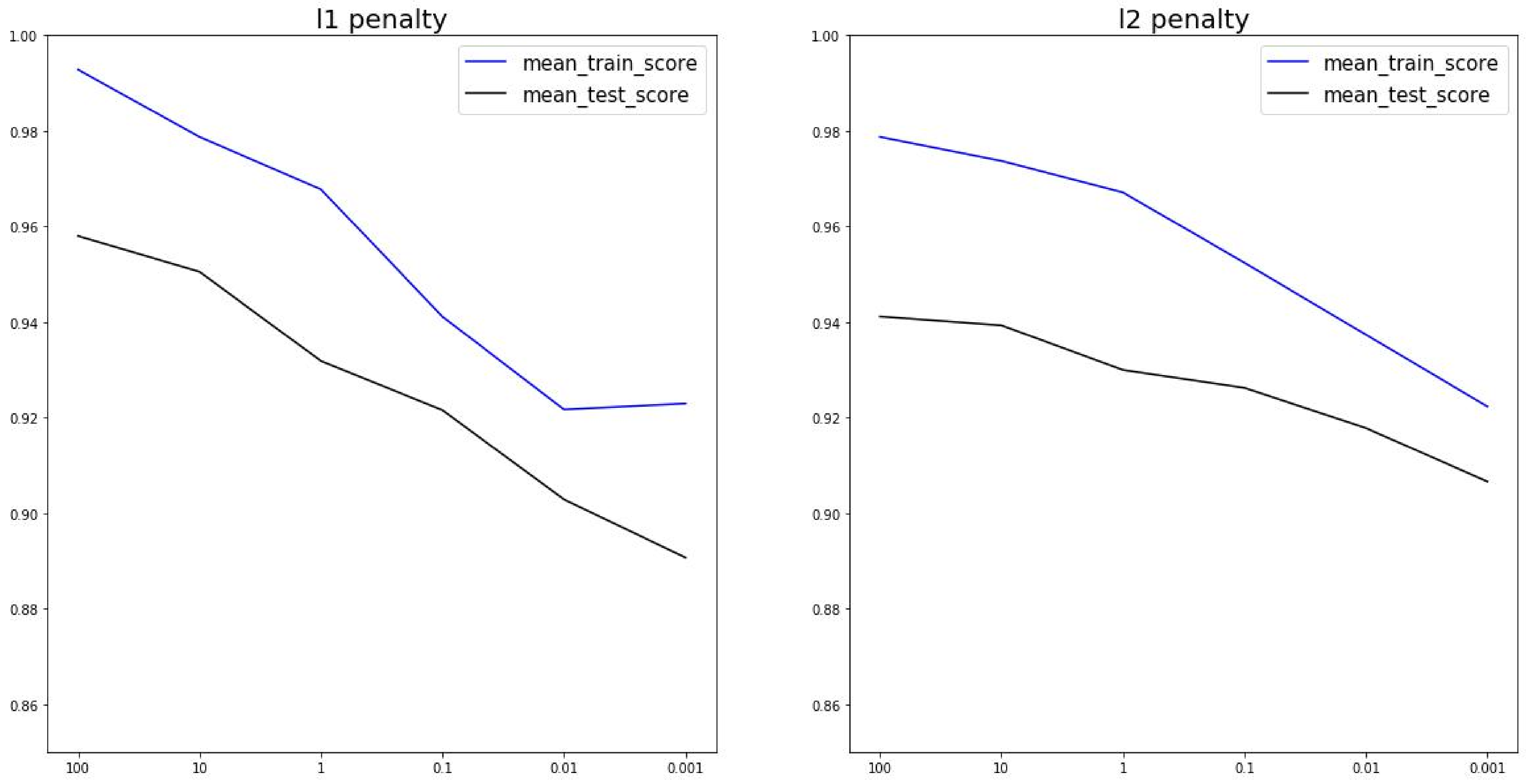
使用 Logistic Regression （l1 penalty)



使用 Logistic Regression (l2 penalty)



Grid Search for Penalty and C



多分类问题

OvR

对于一个新样本
判断是否属于 Ci 类

四分类问题 → 4 个分类器 →

C1 v.s. Rest	No
C2 v.s. Rest	Yes
C3 v.s. Rest	No
C4 v.s. Rest	No

→ C2

每次训练使用全样本数据

OvO

	对于一个新样本 判断属于哪一类
C1 v.s. C2	C1
C1 v.s. C3	C1
C1 v.s. C4	C4
C2 v.s. C3	C3
C2 v.s. C4	C4
C3 v.s. C4	C4

四分类问题 → 6 个分类器 → (6 classifiers above) → C4

每次训练仅使用两类数据

OvR v.s. OvO

	OvR	OvO
子分类器数量	N	C(N,2)
子分类器需要训练的样本量	全样本	仅相关的两类样本
适用情形	类型较多	类型较少

使用 OneVsRestClassifier 和 OneVsRestClassifier

```
from sklearn.multiclass import OneVsRestClassifier, OneVsRestClassifier

ovr = OneVsRestClassifier(LogisticRegression())

ovr.fit(X_train, y_train)
ovr.score(X_test, y_test)
ovr.pred(X_test)
```

可传入任意二分类器