东北财经大学

DONGBEI UNIVERSITY OF FINANCE & ECONOMICS

第三章

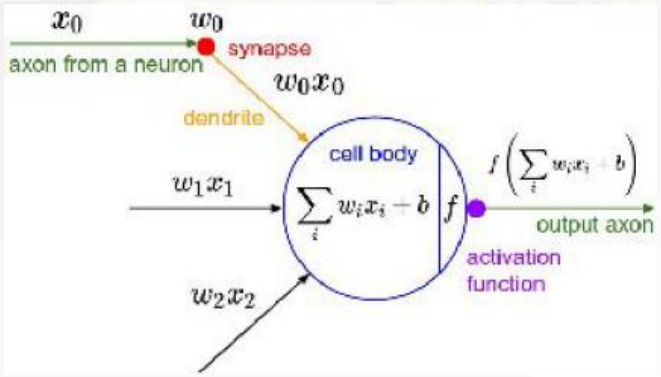
线性模型

管理科学与工程学院

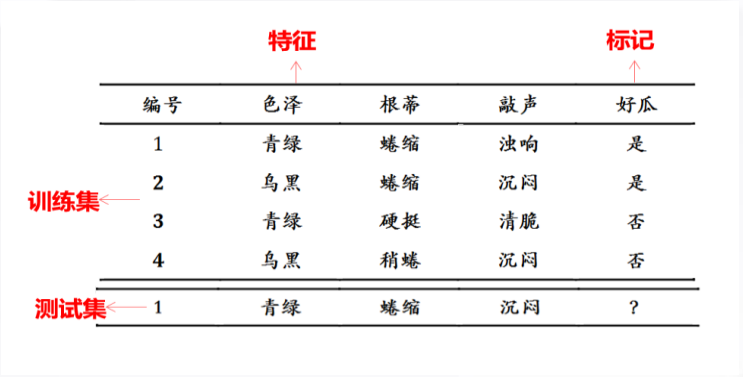
谢琛

1. 线性模型

1. 一般形式：



$$f(\boldsymbol{x}) = w_1x_1 + w_2x_2 + \ldots + w_dx_d + b$$



	编号	色泽	根蒂	敲声	好瓜
		特征			标记
训练集	1	青绿	蜷缩	浊响	是
	2	乌黑	蜷缩	沉闷	是
	3	青绿	硬挺	清脆	否
	4	乌黑	稍蜷	沉闷	否
测试集	1	青绿	蜷缩	沉闷	?

2. 向量形式：

$$\boldsymbol{x} = (x_1; x_2; \ldots; x_d)$$
$$\boldsymbol{w} = (w_1; w_2; \ldots; w_d)$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

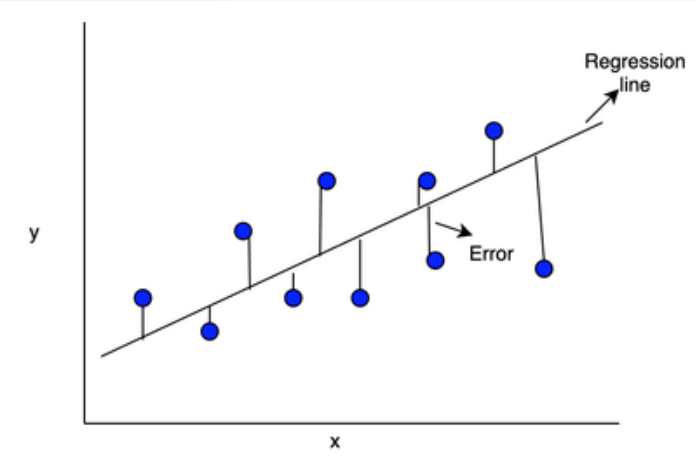
$$f_{好瓜}(\boldsymbol{x}) = 0.2 \cdot x_{色泽} + 0.5 \cdot x_{根蒂} + 0.3 \cdot x_{敲声} + 1$$

综合考虑色泽、根蒂和敲声来判断西瓜好不好:

- 根蒂的系数最大，表明根蒂最要紧
- 敲声的系数比色泽大，说明敲声比色泽更重要

线性回归

1. 单一属性的线性回归：



$$f(x) = wx_i + b$$

$$\text{Error} = \hat{y} - y$$

目标: 找到一条线使所有点对应的Error之和最小

$$\arg\min \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

- 最小二乘法（Least Squared Method）

Error:

$$\sum_{i=1}^{m} (f(x_i) - y_i)^2 = \sum_{i=1}^{m} (wx_i + b - y_i)^2$$

- 分别对w和b求导:

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m} x_i^2 - \sum_{i=1}^{m}(y_i - b)\,x_i\right)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m}(y_i - wx_i)\right)$$

$$w = \frac{\sum_{i=1}^{m} y_i (x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

2. 实现：sklearn.linear_model.LinearRegression()

- 属性: coef_ 和 intercept_

① regressor = LinearRegression()

② regressor.fit(x_train,y_train)

③ y_pred = regressor.predict(x_test)

（1）导入包和数据

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.linear_model import LinearRegression
%matplotlib inline
iris = load_iris()   #导入数据集iris
data=pd.DataFrame(iris.data)
data.columns=['sepal-length', 'sepal-width', 'petal-length', 'petal-width']
data.head()   #显示前5行
```

（2）对数据集中的'petal-length'和'petal-width'两列数据进行回归分析

```
# 使用sklearn完成一元线性回归
x = data['petal-length'].values
y = data['petal-width'].values
x = x.reshape(len(x),1)
y = y.reshape(len(y),1)
clf = LinearRegression()
clf.fit(x,y)
pre = clf.predict(x)
plt.scatter(x,y,s=50)
plt.plot(x,pre,'r-',linewidth=2)
plt.xlabel('petal-length')
plt.ylabel('petal-width')
for idx, m in enumerate(x):
    plt.plot([m,m],[y[idx],pre[idx]], 'g-')
plt.show()
```

（3）显示回归线的参数

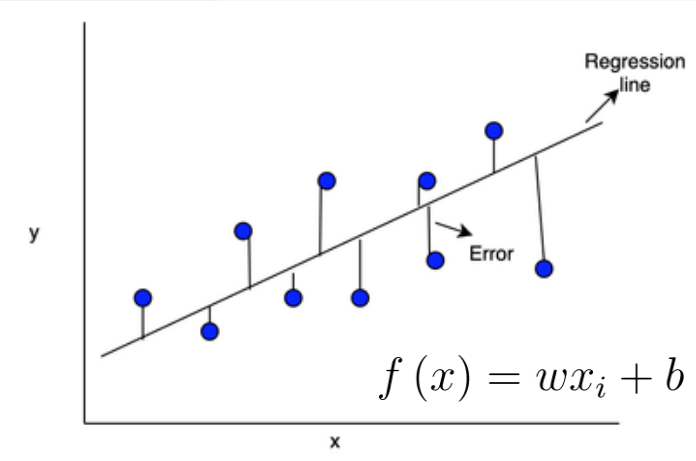
```
print(u"系数", clf.coef_  )
print(u"截距", clf.intercept_  )
print(np.mean(y-pre)**2  )
```

（4）对花萼长度为3.9的花，预测其花萼宽度。

```
print(clf.predict([[3.9]]) )
```

线性回归

1. 单一属性的线性回归：



$$\text{Error} = \hat{y} - y$$

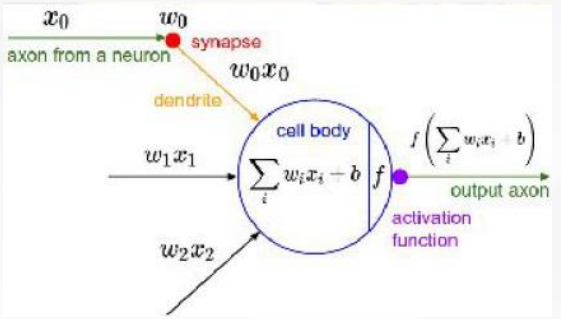
目标: 找到一条线使所有点对应的Error之和最小

$$\arg\min \sum_{i=1}^{m} (f(x_i) - y_i)^2$$

- 最小二乘法（Least Squared Method）

3. 多元线性回归：

1. 一般形式：



$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

2. 向量形式：

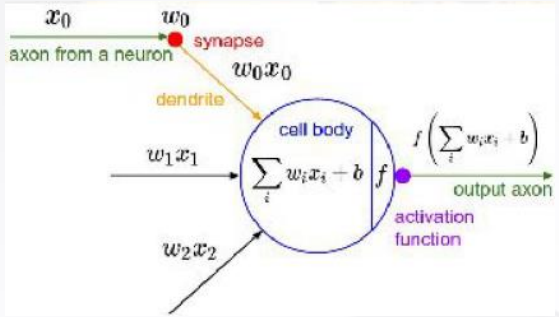
$$\boldsymbol{x} = (x_1; x_2; \ldots; x_d)$$
$$\boldsymbol{w} = (w_1; w_2; \ldots; w_d)$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$$

线性回归

3. 多元线性回归：

1. 一般形式：



$$f(\boldsymbol{x}) = w_1x_1 + w_2x_2 + \ldots + w_dx_d + b$$

2. 向量形式：

$$\boldsymbol{x} = (x_1; x_2; \ldots; x_d)$$
$$\boldsymbol{w} = (w_1; w_2; \ldots; w_d)$$

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$$

把$\boldsymbol{w}$和 b 吸收入向量形式 $\hat{\boldsymbol{w}} = (\boldsymbol{w}; b)$，数据集表示为

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1^{\mathrm{T}} & 1 \\ \boldsymbol{x}_2^{\mathrm{T}} & 1 \\ \vdots & \vdots \\ \boldsymbol{x}_m^{\mathrm{T}} & 1 \end{pmatrix}$$

$$\boldsymbol{y} = (y_1; y_2; \ldots; y_m)$$

最小二乘法（least square method）

$$\hat{\boldsymbol{w}}^* = \arg\min_{\hat{w}} \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}^{\mathrm{T}}\right)\left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)$$

令 $E_{\hat{\boldsymbol{w}}} = \left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)^{\mathrm{T}}\left(\boldsymbol{y} - \mathbf{X}\hat{\boldsymbol{w}}\right)$ ，对$\hat{\boldsymbol{w}}$求导得到

$$\frac{\partial E_{\hat{\boldsymbol{w}}}}{\partial \hat{\boldsymbol{w}}} = 2\mathbf{X}^{\mathrm{T}}\left(\mathbf{X}\hat{\boldsymbol{w}} - \boldsymbol{y}\right)$$

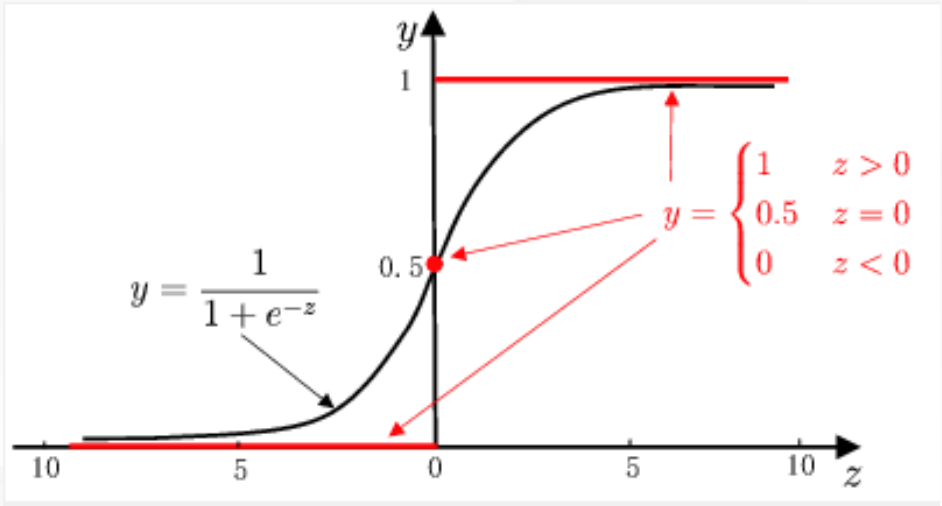
$\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 是满秩矩阵或正定矩阵，则

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

其中$\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}$是 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 的逆矩阵，线性回归模型为

$$f(\hat{\boldsymbol{x}}_i) = \hat{\boldsymbol{x}}_i^{\mathrm{T}}\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$$

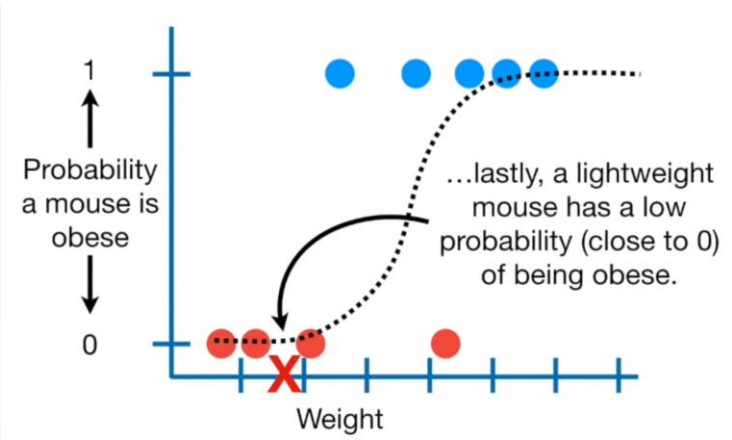
1. 对数几率函数（Logistic Function）：

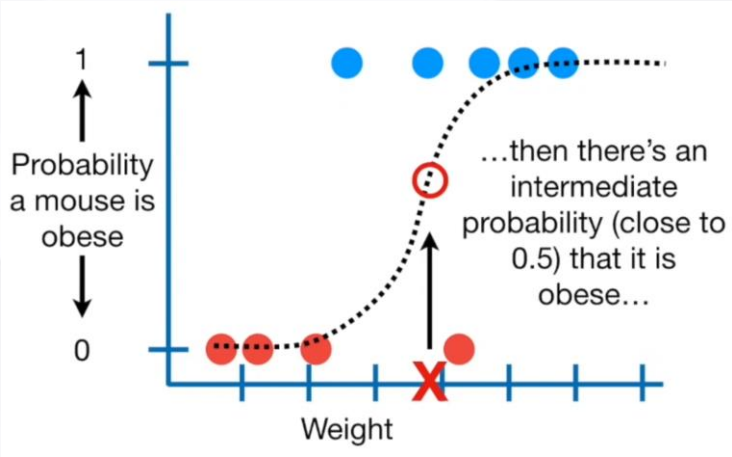


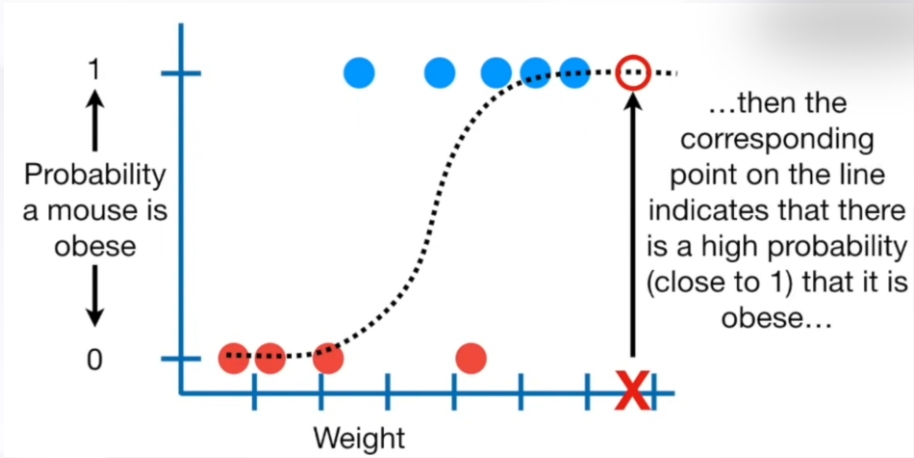
$$y = \frac{1}{1 + e^{-z}}$$

2. Logistic Regression：

- 线性模型
- 线性回归
- Logistics Regression
- LDA







3. Logistic Regression

3. 实现：sklearn.linear_model.LogisticRegression()

- 属性: coef_ 和 intercept_

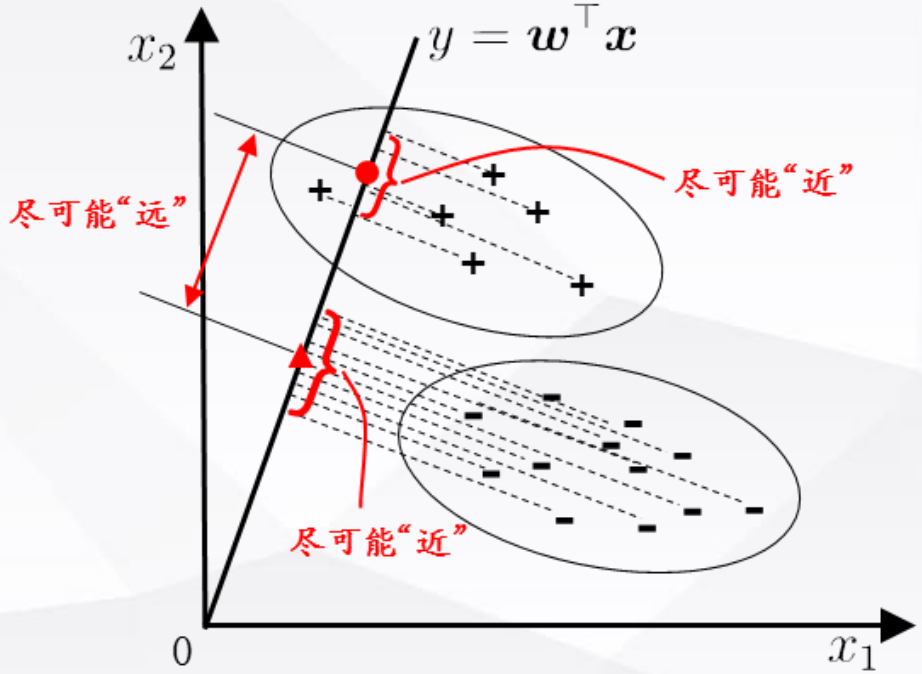
① solver = liblinear -> 坐标轴下降优化法

② solver = lbfgs / newton-cg -> 两种拟牛顿优化方法

③ solver = sag -> 随机梯度下降优化法

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

clf=LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial').fit(X_train, y_train)

print('coef:\n',clf.coef_)
print('intercept:\n',clf.intercept_ )

print('predict first two:\n',clf.predict(X_train[:2, :]))
print('classification score:\n',clf.score(X_train, y_train))

predict_y = clf.predict(X_test)
print('classfication report:\n ',metrics.classification_report (y_test,predict_y))
```

1. LDA：线性判别分析（Linear Discriminant Analysis）



- 欲使同类样例的投影点尽可能接近
- 欲使异类样例的投影点尽可能远离

LDA也可被视为一种监督降维技术