逻辑回归

Sigmoid函数

```
import numpy as np
import matplotlib.pyplot as plt
def sigmoid(t):
    return 1/(1 + np.exp(-t))
x = np.linspace(-10, 10, 500)
y = sigmoid(x)
plt.plot(x,y)
plt.show()
```

$\hat{p} = \sigma(\theta^T \cdot x_b) = \frac{1}{1+e^{-\theta^T \cdot x_b}}$

$\hat{y} = \begin{cases} 1 & \hat{y} \geq 0.5 \\ 0 & \hat{y} < 0.5 \end{cases}$

损失函数

$cost = \begin{cases} -log(\hat{p}) & y = 1 \\ -log(1-\hat{p}) & y = 0 \end{cases}$

$cost = -ylog(\hat{p}) - (1-y)log(1-\hat{p})$

$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log(\hat{p}^{(i)}) + (1-y^{(i)})log(1-\hat{p}^{(i)})$

$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} log(\sigma(X_b^{(i)}\theta)) + (1-y^{(i)})log(1-\sigma(X_b^{(i)}\theta))$

损失函数的梯度

$\frac{J(\theta)}{\theta_j} = \frac{1}{m}\sum_{i=1}^{m}(\sigma(X_b^{(i)}\theta) - y^{(i)})X_j^{(i)}$

$$\nabla J(\theta) = \frac{1}{m}\begin{pmatrix} \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)}) \\ \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)}) \cdot X_1^{(i)} \\ \cdots \\ \sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)}) \cdot X_n^{(i)} \end{pmatrix} = \frac{1}{m} X_b^T \cdot (\sigma(X_b\theta) - y)$$

决策边界

$\theta^T \cdot x_b = 0$

多项式特征

sklearn中的使用

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(666)
X = np.random.normal(0,1,size=(200,2))
y = np.array(X[:,0]**2 + X[:,1] < 1.5,dtype='int')
for _ in range(20):
    y[np.random.randint(200)] = 1
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=666)
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
def PolynomialLogisticRegression(degree, C, penalty='l2'):
    return Pipeline([
        ('poly', PolynomialFeatures(degree=degree)),
        ('std_scaler', StandardScaler()),
        ('log_reg', LogisticRegression(C=C, penalty=penalty))
    ])
poly_log_reg = PolynomialLogisticRegression(degree=20, C=0.1,penalty='l1')
poly_log_reg.fit(X_train, y_train)
poly_log_reg.score(X_test, y_test)
```

OvR与OvO

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=666)
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(multi_class="multinomial",solver="newton-cg")
log_reg.fit(X_train, y_train)
log_reg.score(X_test, y_test)
```

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.multiclass import OneVsOneClassifier
ovr = OneVsRestClassifier(log_reg)
ovr.fit(X_train, y_train)
ovr.score(X_test, y_test)
```