线性回归实战

线性回归实战

使用正规方程进行求解

简单的线性回归

$y = wx + b$

一元一次方程，在机器学习中一元表示一个特征，b表示截距，y表示目标值。

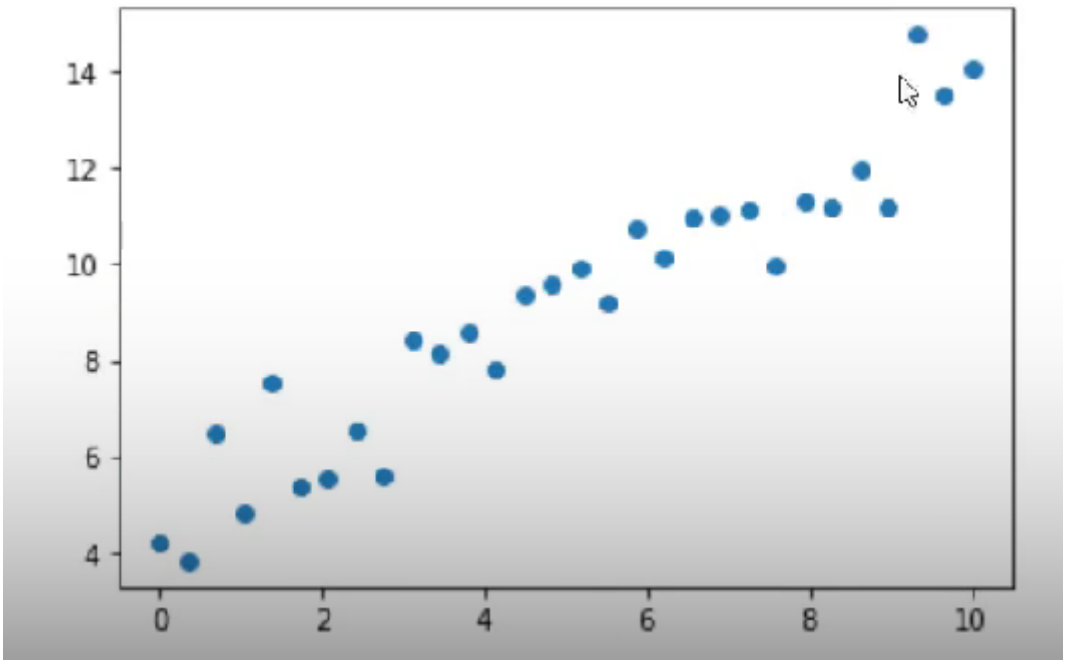
代码块

```
import numpy as np
import matplotlib.pyplot as plt

# 转化成矩阵
X = np.linspace(0,10,num=30).reshape(-1,1)

# 斜率和截距，随机生成
w = np.random.randint(1,5,size=1)
b = np.random.randint(1,10,size =1)

# 根据一元一次方程计算目标值y，并加上“噪声”，数据有上下波动~
y = X * w + b + np.random.randn(30,1)
plt.scatter(X,y)

# 重新构造x，b截距，相当于系数w0，前面统一乘以1
X = np.concatenate([X,np.full(shape = (30,1),fill_value= 1)],axis = 1)

#正规方程求解
theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y).round(2)
print('一元一次方程真实的斜率和截距是:',w,b)
print('通过正规方程求解的斜率和截距是:',theta)

#根据求解的斜率和截距绘制线性回归线型图
plt.plot(X[:,0],X.dot(0),color ='green')
plt.show()
```

效果如下（random.randn是随机生成正态分布数据，所以每次执行图形会有所不同）：

代码块

```
一元一次方程真实的斜率和截距是: [3] [5]
通过正规方程求解的斜率和截距是: [[2.92]
 [5.54]]
```



多元线性回归

$$y = w_1 x_1 + w_2 x_2 + b$$

二元一次方程，x_1、x_2相当于两个特征值，b是方程截距

代码块

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 设置中文字体支持
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False    # 用来正常显示负号

# 设置随机种子保证结果可复现
np.random.seed(42)

# 生成两个特征（30个样本，每个样本2个特征）
X1 = np.linspace(0, 10, num=100)
X2 = np.linspace(5, 15, num=100)
X = np.column_stack((X1, X2))

# 真实参数：两个权重和一个截距
true_w = np.array([2.5, 1.8]).reshape(-1, 1)
true_b = np.array([3.0])

```

```python
# 生成目标值 y = X1*w1 + X2*w2 + b + 噪声
y = X.dot(true_w) + true_b + np.random.randn(100, 1) * 2

# 添加截距项（全1列）
X_with_bias = np.concatenate([X, np.ones((100, 1))], axis=1)

# 正规方程求解参数
theta = np.linalg.inv(X_with_bias.T.dot(X_with_bias)).dot(X_with_bias.T).dot(y)

# 提取求解的参数
w1_pred, w2_pred, b_pred = theta.flatten().round(2)

print('真实参数:')
print(f'权重 w1 = {true_w[0][0]}, w2 = {true_w[1][0]}, 截距 b = {true_b[0]}')
print('\n通过正规方程求解的参数:')
print(f'权重 w1 = {w1_pred}, w2 = {w2_pred}, 截距 b = {b_pred}')

# 创建3D图形
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

# 绘制原始数据点
ax.scatter(X[:, 0], X[:, 1], y, c='r', marker='o', label='原始数据')

# 创建回归平面
x1_range = np.linspace(0, 10, 10)
x2_range = np.linspace(5, 15, 10)
X1_plane, X2_plane = np.meshgrid(x1_range, x2_range)
Y_plane = w1_pred * X1_plane + w2_pred * X2_plane + b_pred

# 绘制回归平面
ax.plot_surface(X1_plane, X2_plane, Y_plane, alpha=0.5, color='blue', label='回归平面')

# 设置坐标轴标签
ax.set_xlabel('特征 X1')
ax.set_ylabel('特征 X2')
ax.set_zlabel('目标值 y')
ax.set_title('二元线性回归可视化')

# 添加图例
scatter_proxy = plt.Line2D([0], [0], linestyle='none', c='red', marker='o')
surface_proxy = plt.Rectangle((0,0), 1, 1, fc='blue', alpha=0.5)
ax.legend([scatter_proxy, surface_proxy], ['原始数据', '回归平面'], loc='upper left')

plt.tight_layout()
```

```
66   plt.show()
```

机器学习库

https://scikit-learn.org/stable/index.html

1. 分类问题
2. 回归问题
3. 聚类问题
4. 降维问题
5. 模型选择问题
6. 数据预处理问题

举例线性回归参考2.1文章