Python 实现对数几率回归模型

刘存添

计算机 1601

1611640105

Contents

1 问题描述

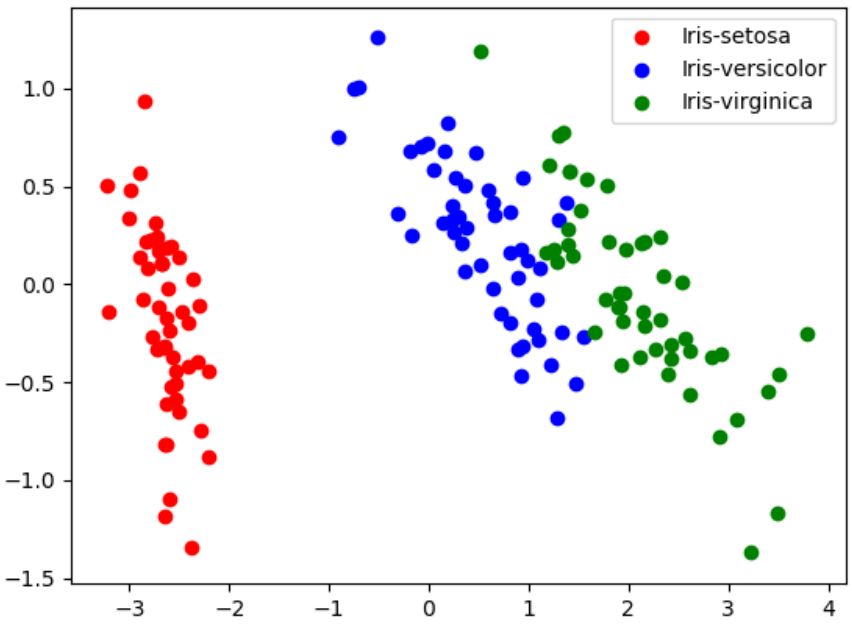
从 Iris 数据集中任选两类，使用对数几率回归模型对其进行分类。

2 数据集描述

Iris 数据集有 150 组数据，分为 3 类。每组数据有 5 种属性，分别为 sepal length、sepql width、petal length、petal width 及 class。前 4 种属性是以 cm 为单位的浮点数值；最后一种属性是固定的字符串，分别是 Iris Setosa、Iris Versicolour 及 Iris Virginica，代表 3 个品种的花。

Attribute	Data Type
sepal length	numeric
sepql width	numeric
petal length	numeric
petal width	numeric
Class	Iris Setosa Iris Versicolour Iris Virginica

对数据做 PCA 降维处理得出下图：

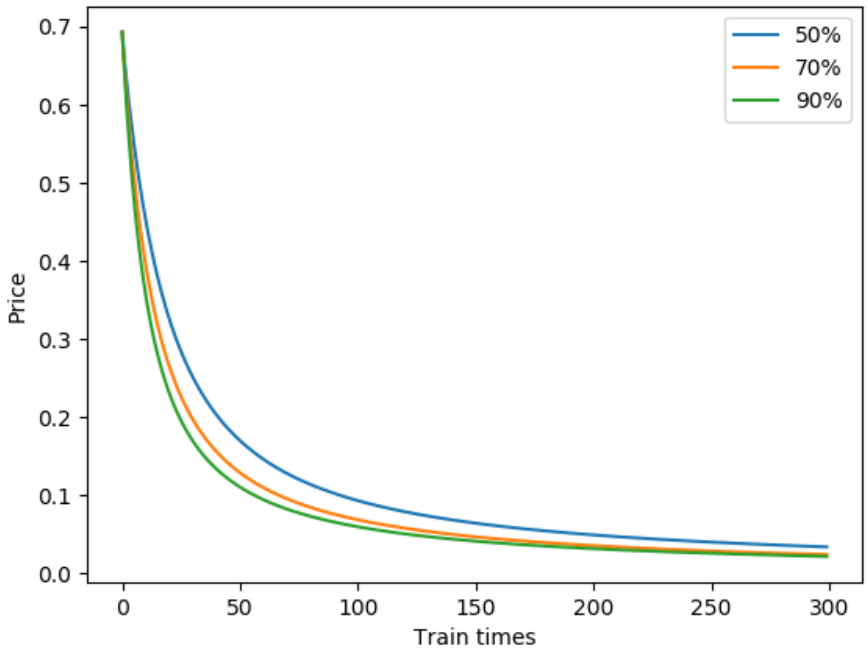


PCA 降维

3 实验结果图

实验选取 Iris Setosa 与 Iris Versicolour 两种类别做二分类。对数据以不同比例（50%, 70%, 90%）划分训练集与测试集在 0.0014 学习率的情况下做 300 次训练，得出下图。

损失函数： $-y*log(h(x))-(1-y)*log(1-h(x))$



损失函数变化曲线图

经过 300 次训练后，分类正确率如下表：

比例	正确率
5:5	96.609703%
7:3	97.592389%
9:1	97.822833%

4 实验结果分析

由实验结果可得，随着训练次数增加，测试正确率提高。

附录 A　主模块

```
IRIS_PATH = 'iris.data'
LEARN_RATE = 0.0014
TIMES = 300

def TEST_SET_PATH():
    return 'archive/train-%d.data'%int(TT_RATE*100)
def TRAIN_SET_PATH():
    return 'archive/test-%d.data'%int(TT_RATE*100)

TT_RATE = 0.5
exec(open('iris.py').read()) # 生成测试集 训练集
exec(open('logit.py').read())

TT_RATE = 0.7
exec(open('iris.py').read())
exec(open('logit.py').read())

TT_RATE = 0.9
exec(open('iris.py').read())
exec(open('logit.py').read())
```

附录 B　样本生成

```
import numpy as np
import random

train_set = []
test_set = []
iris = ([], [], [])
map = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}

data = np.genfromtxt(IRIS_PATH, delimiter=',', usecols=range(5))
target = np.genfromtxt(IRIS_PATH, delimiter=',', usecols=(4), dtype=str)

for i in range(len(data)):
    idx = map[target[i]]
    data[i][4] = idx
    iris[idx].append(data[i])

for i in range(2): # 取前两种属性
    random.shuffle(iris[i])
    si = int(len(iris[i]) * TT_RATE)
    train_set += iris[i][:si]
    test_set += iris[i][si:]


random.shuffle(train_set)
random.shuffle(test_set)
train_set = np.array(train_set)
test_set = np.array(test_set)

np.savetxt(TRAIN_SET_PATH(), train_set, fmt='%.1f', delimiter=',')
```

```
np.savetxt(TEST_SET_PATH(), test_set, fmt='%.1f', delimiter=',')
```

附录 C　对数几率回归模型

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


def sigmoid(z):
    if z >= 0:
        return 1.0/(1+np.exp(-z))
    else:
        return np.exp(z)/(1+np.exp(z))


def coe(B): # 求导
    tmp = np.zeros(5)
    for c in train_set:
        x, y = c[:4], c[-1]
        X = np.append(x, [1])
        w, b = B[:4], B[-1]
        p1 = np.exp(np.dot(w.T, x) + b)
        tmp += np.dot(X, y - p1 / (1 + p1))
    return -tmp


def cost_func(h, y): # 损失函数
    return -y * np.log(h) - (1 - y) * np.log(1-h)


bh = [np.zeros(5)]

train_set = np.genfromtxt(TRAIN_SET_PATH(), delimiter=',')
test_set = np.genfromtxt(TEST_SET_PATH(), delimiter=',')

print('TRAIN : TEST ')
print('%5d : %-5d' % (len(train_set), len(test_set)))

start = time.time()

for i in range(TIMES): # 训练
    b = bh[i]
    bh.append(b - np.dot(LEARN_RATE, coe(b)))

end = time.time()
print('Done. Took %.3f seconds.' % (end - start))

pots = []
for i in range(0, TIMES, 1):
    B = bh[i]
    svm = 0.0
    for c in test_set: # 测试
        x, y = c[:4], c[-1]
        w, b = B[:4], B[-1]
```

```
        h = sigmoid(np.dot(w.T, x) + b)
        svm += cost_func(h, y)
    svm /= len(test_set)
    pots.append(svm)

print('正确率: %f%%'%((1-pots[-1])*100))
plt.plot(pots, label='%d%%' % int(TT_RATE*100))

plt.xlabel('Train times')
plt.ylabel('Price')
plt.legend()
plt.savefig('picture/%d-%s-test.png' % (int(TT_RATE*100), TIMES))
```

附录 D　PCA 降维算法实现

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

iris_path = 'iris.data'
map = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}

x = np.genfromtxt(iris_path, delimiter=',', usecols=range(4))

mean_x = x.mean(axis=0)
x = x - mean_x # 去中心化
cov_x = np.cov(x, rowvar=0) # 协方差矩阵
eval, evec = np.linalg.eig(np.mat(cov_x)) # 特征值与特征向量
idx = np.argsort(eval)
idx = idx[:-3:-1]
x = x * evec[:, idx]


plt.scatter(x[:50][:, 0].tolist(), x[:50][:, 1].tolist(),
           c='red', label='Iris-setosa')
plt.scatter(x[50:100][:, 0].tolist(), x[50:100]
           [:, 1].tolist(), c='blue', label='Iris-versicolor')
plt.scatter(x[100:][:, 0].tolist(), x[100:][:, 1].tolist(),
           c='green', label='Iris-virginica')
plt.legend()
plt.savefig('picture/pca.png')
plt.show()
```