南開大學
Nankai University

南　开　大　学

计 算 机 学 院

机器学习实验报告

实验三 参数估计与非参数估计

姓名：王泳鑫

学号：1911479

年级：2019级

专业：计算机科学与技术

指导教师：卫金茂

2021 年 11 月 9 日

摘要

关键字：参数估计，**Machine Learning ， Deep Learning**

目录

一、 实验描述

（一） 实验内容

（二） 实验要求

基本要求：

在两个数据集应用“最大后验概率规则”进行分类实验，计算分类错误率，分析实验结果。

中级要求：

在两个数据集合上使用高斯核函数估计方法，应用“似然率测试规则”分类，在[0.1,0.5,1,1.5,2]范围内交叉验证找到最优h值，分析实验结果。

二、 代码实现

（一） 基本要求

首先，这次实验没有给出数据集，而是需要我们利用np.random.multivariate_normal(mean, cov, temp_num)函数来生成随机数据，这些数据符合正态分布，生成两个各包含 N = 1000 个二维随机矢量的数据集合 X_1 和 X_2, 数据集合中随机矢量来自于三个分布模型, 分别满足均值矢量 $\mathbf{m}_1 = [1,1]^T, \mathbf{m}_2 = [4,4]^T, \mathbf{m}_3 = [8,1]^T$ T和协方差矩阵 $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{S}_3 = 2\mathbf{I}$, 其中 $\mathbf{I}$ 是 2×2 的单位矩阵。在生成数据集合 X 时, 假设来自三个分布模型的先验概率相同 $p(w_1) = p(w_2) = p(w_3) = 1/3$; 而在生成数据集合 X ’ 时, 先验概率分别为 $p(w_1) = 0.6, p(w_2) = 0.3, p(w_3) = 0.1$ 。代码如下：

```
def generate_dataset():
    m1 = (1, 1)
    m2 = (4, 4)
    m3 = (8, 1)

    s1 = s2 = s3 = [[2, 0], [0, 2]]
    x1 = []
    x2 = []

    for i in range(333):
        x1.append(np.random.multivariate_normal(m1, s1))
        x1.append(np.random.multivariate_normal(m2, s2))
        x1.append(np.random.multivariate_normal(m3, s3))
    x1.append(np.random.multivariate_normal(m1, s1))
    for i in range(600):
        x2.append(np.random.multivariate_normal(m1, s1))

    for i in range(300):
        x2.append(np.random.multivariate_normal(m1, s1))

    for i in range(100):
        x2.append(np.random.multivariate_normal(m1, s1))

    x1 = np.array(x1)
    x2 = np.array(x2)
    return x1, x2
```

然后我们要对行训练集与验证集的划分，与前两次实验一样，我们仍旧采用多折交叉验证的方式。需注意的是，为保证验证集中各类数据出现概率均等，我们需要先把 3 类数据分开后

再分别划分训练集和测试集：

```
def train_valid_split(data, label, fold, idx):
 sample1 = data[:333]
 label1 = label[:333]
 sample2 = data[333:666]
 label2 = label[333:666]
 sample3 = data[666:]
 label3 = label[666:]
 train_idx = []
 valid_sample = []
 valid_label = []
 for i in range(sample1.shape[0]):
     if i%fold == idx:
         valid_sample.append(sample1[i, :])
         valid_sample.append(sample2[i, :])
         valid_sample.append(sample3[i, :])
         valid_label.append(label1[i])
         valid_label.append(label2[i])
         valid_label.append(label3[i])
     else:
         train_idx.append(i)
 train_sample1 = sample1[train_idx, :]
 train_sample2 = sample2[train_idx, :]
 train_sample3 = sample3[train_idx, :]
 train_label1 = label1[train_idx]
 train_label2 = label2[train_idx]
 train_label3 = label3[train_idx]
 valid_sample = np.array(valid_sample)
 valid_label = np.array(valid_label)
 return train_sample1, train_sample2, train_sample3, \
        train_label1, train_label2, train_label3, \
        valid_sample, valid_label
```

然后我们便可开始参数估计的核心内容了，首先我们进行正态分布条件下的参数估计。根据前文分析得到的参数计算公式，定义参数计算函数为：

```
def cal_para(sample):
 dim = sample.shape[1]
 mean = np.mean(sample, 0)
 sigma = np.cov(sample.T)
 det_sigma = np.linalg.det(sigma)
 return dim, mean, sigma, det_sigma
```

得到概率密度函数中的各参数后，我们构造概率密度函数公式，定义概率计算函数为：

```
def cal_prob(vec, sample):
 dim, mean, sigma, det_sigma = cal_para(sample)
 # 正态分布概率密度函数
 prob = (1 / (((2*np.pi) ** (dim/2)) * np.sqrt(det_sigma))) * \
        np.exp((-(1/2) * (vec - mean)).dot(np.mat(sigma).I).dot(vec - mean))
 return prob
```

得到概率值后，我们基于实验要求中的似然概率测试规则，比较验证集中各条样本对应的 3 类分布的概率大小，将概率值最大的 1 类作为验证样本的预测标签：

```
def pred_label(valid, train_sample1, train_sample2, train_sample3):
```

```
    label = []
    for i in range(valid.shape[0]):
        prob1 = cal_prob(valid[i, :], train_sample1)
        prob2 = cal_prob(valid[i, :], train_sample2)
        prob3 = cal_prob(valid[i, :], train_sample3)
        if max([prob1, prob2, prob3]) == prob1:  # 选择概率值最大的作为预测标签
            label.append(1)
        elif max([prob1, prob2, prob3]) == prob2:
            label.append(2)
        else:
            label.append(3)
    return np.array(label)
```

最后再通过交叉检验得到正确率：

```
def cross_valid(data, label, fold):
    acc = []
    for i in range(fold):
        train_sample1, train_sample2, train_sample3, \
        train_label1, train_label2, train_label3, \
        valid_sample, valid_label = train_valid_split(data, label, fold, i)
        pred = pred_label(valid_sample, train_sample1, train_sample2,train_sample3)
        correct = 0
        for j in range(valid_label.shape[0]):
            if pred[j] == valid_label[j]:
                correct += 1
        print(i, 'acc', correct/valid_label.shape[0])
        acc.append(correct/valid_label.shape[0])
    return np.mean(acc)
```

（二） 提高要求

对于选取最合适h的大小，我们可以通过sklearn库中kde.KernelDensity来改变，再通过最大似然概率，选取标签，代码如下（大致与要求1相同）：

```
def cal_prob_smooth(vec, sample,h):
    model = kde.KernelDensity(kernel='gaussian', bandwidth=h).fit(sample)
    prob = np.exp(model.score_samples(vec.reshape(1, -1)))
    return prob
```

最后仍是通过交叉检验得到平均的准确率，在不同的h值下，平均的准确率也不同：

```
def cross_valid(data, label, fold,h):
    acc = []
    for i in range(fold):
        train_sample1, train_sample2, train_sample3, \
        train_label1, train_label2, train_label3, \
        valid_sample, valid_label = train_valid_split(data, label, fold, i)
        pred = pred_label(valid_sample, train_sample1, train_sample2,train_sample3,h)
        correct = 0
        for j in range(valid_label.shape[0]):
            if pred[j] == valid_label[j]:
                correct += 1
        ##print(h,"   ",i, 'acc', correct/valid_label.shape[0])
        acc.append(correct/valid_label.shape[0])

```

```
    print("when h is ",h," , the acc is",np.mean(acc))
    return np.mean(acc)
```

三、 实验结果展示与分析

（一） 实验结果1

实验结果如图2所示

```
0 acc 0.8627450980392157
1 acc 0.9019607843137255
2 acc 0.9411764705882353
3 acc 0.9696969696969697
4 acc 0.9292929292929293
5 acc 0.9090909090909091
6 acc 0.9494949494949495
7 acc 0.9494949494949495
8 acc 0.9191919191919192
9 acc 0.8787878787878788
0.921093285799168
```

图 1: 交叉验证结果

（二） 实验结果2

实验结果如图2所示

```
when h is  0.1  , the acc is 0.9028041610131161
when h is  0.5  , the acc is 0.9368611488014473
when h is  1  , the acc is 0.9388511985526911
when h is  1.5  , the acc is 0.9378561736770692
when h is  2  , the acc is 0.9368611488014473
```

图 2: 交叉验证结果

我们可以看到对于随机生成数据集来说，h在[0.5,1,1.5,2]这几个h值下，准确率都比较高，但是当h为1时，平均准确率最高，达到0.93885。