模式识别作业三

姓名：李宁　　学号：2120180459　　专业：计算机科学与技术

问题一

一、问题描述

给定一个数据文件 PRHW#3Data.mat，其中包含六个矩阵：三个训练集样本矩阵 c1、c2、c3，分别对应于三个不同类别；三个测试集样本矩阵 t1，t2，t3，分别为相应三个类别的测试样本。六个数据矩阵，每个矩阵均有 500 数据样本，每个样本有 2 维特征。实验要求如下：

1）使用 k 近邻分类器解决该模式分类问题；

2）给出对应最佳分类结果的 k 值和总的分类错误率；

3）给出并解释当 k=1、最佳 k 值和 k=50 时分类器的分类错误率。

二、基本思路

① 数据分析

由原题目所述，六个矩阵中 c1、c2、c3 为第 i 个类的训练样本集合，t1，t2，t3 为第 i 类的测试样本集合，六个矩阵尺寸大小均为 500*2。下图 1 是 c1 矩阵的部分数据信息。

```
print(c1)
None
[[ 8.18617308e-01 -1.24631764e+00]
 [ 1.24205237e+00 -1.00636226e+00]
 [ 6.07695618e-01  1.51405584e+00]
 [-7.56317920e-01  2.19635985e-01]
 [ 7.72703329e-01  1.40469198e+00]
 [ 5.11893563e-01 -3.13829810e-01]
 [ 1.48662938e+00  1.57180481e+00]
 [-1.56271088e-01  1.66155418e+00]
 [-5.52274830e-01  1.25945833e-01]
 [ 8.35054570e-01  7.92971798e-02]
 [ 4.25680256e-01  3.67056491e-01]
 [-6.47147611e-01 -3.83594626e-01]
 [-7.01417340e-01  5.78803077e-01]
 [ 7.70330245e-01  9.26718492e-01]
```

图 1 c1 矩阵部分数据信息

可以看到，六个矩阵的读入形式均为 500*2 的二维数组。在分类问题，对于训练数据集，除了数据本身，还要有与数据相对应的类标签，即我们需要在 c1、c2、c3 原有数据之后追加标签数据。而作为测试数据集，我们需要在 t1、t2、t3 原有数据之后追加用于存放预测结果的数据空间，从而将预测结果与它们的真实标签进行比对，进而计算出分类错误率。

② 数据预处理

根据上一步骤数据分析的结果，我们需要在原有的数据集上对 c1、c2、c3、t1、t2、t3 的原始数据进行预处理工作。

对于训练集样本，我们在三个训练集矩阵 c1、c2、c3 之后分别追加一列类别标签，在 c1 矩阵后追加类别 1 标签，c2 矩阵后追加类别 2 标签，c3 矩阵后追加类别 3 标签。预处理完的样本数据如下图 2 所示：

```
print(c1)
None
[[ 0.81861731 -1.24631764  1.        ]
 [ 1.24205237 -1.00636226  1.        ]
 [ 0.60769562  1.51405584  1.        ]
 ...
 [-0.73947096  0.69788936  1.        ]
 [-0.38523295 -0.36186442  1.        ]
 [-0.15158061 -0.99798603  1.        ]]
```

```
print(c2)
None
[[ 0.64294395  2.91282993  2.        ]
 [ 2.1285129  -1.78349417  2.        ]
 [ 0.93104078  2.52975937  2.        ]
 ...
 [ 1.55830172 -2.01954533  2.        ]
 [-2.8592236   0.13079663  2.        ]
 [-2.93575394  0.73576683  2.        ]]
```

```
print(c3)
None
[[ 3.8614633   0.53796056  3.        ]
 [ 0.63836365 -3.66916656  3.        ]
 [ 1.48247191 -4.12813708  3.        ]
 ...
 [ 2.38127175  3.16414714  3.        ]
 [ 3.80165016 -1.78796082  3.        ]
 [ 2.36318432  3.66711742  3.        ]]
```

图 2 预处理后的训练数据

与训练集数据预处理类似，对于测试样本，本次实验在三个测试集矩阵 t1、t2、t3 之后分别追加两列数据，第一列为各个测试集数据的真实类标签，第二列用于存放预测类标签，其中预测类标签初始化为 0。预处理后的测试数据如下图 3 所示：

```
print(t1)
None
[[-0.1835806  -0.3188704   1.        0.       ]
 [ 0.60484635  1.03162384  1.        0.       ]
 [-1.07730354 -0.85500386  1.        0.       ]
 ...
 [ 0.55855513  0.98579998  1.        0.       ]
 [ 1.32378702  0.29506091  1.        0.       ]
 [ 0.11530387  0.30766303  1.        0.       ]]
```

```
print(t2)
None
[[-0.12866044  3.18940012  2.        0.       ]
 [-2.10498973 -2.02676986  2.        0.       ]
 [-2.87653764 -1.52018875  2.        0.       ]
 ...
 [-1.47955005 -2.3998401   2.        0.       ]
 [ 1.73420539  2.50015305  2.        0.       ]
 [ 0.82921555  3.09302288  2.        0.       ]]
```

```
print(t3)
None
[[-1.00416504 -3.76123948  3.        0.       ]
 [-3.44178414  2.52260706  3.        0.       ]
 [-3.88544724  0.99136656  3.        0.       ]
 ...
 [ 2.67003254  3.20203636  3.        0.       ]
 [-3.84094876  1.57379477  3.        0.       ]
 [ 2.97040329 -2.19973026  3.        0.       ]]
```

图 3 预处理后的测试数据

③ 构建分类器

根据题意，本次实验需要构建一个 k-NN 最近邻分类器，使用上述两步骤获得的预处理数据，提取测试集数据每个样本的前两维数据，作为待分类样本点的坐标，依次遍历该样本点与三个训练集 c1、c2、c3 所有点的曼哈顿距离（或者欧式距离），判断在 k 个最近邻训练样本中哪一类样本的数据占比最多，则把待分类样本点的类别（第四维数据）置为占比最多数据所属的标签。算法具体流程在之后介绍。

三、原理及算法

K-NN 算法的具体流程如下：

1）准备数据，对数据进行预处理获得训练样本 c1、c2、c3；测试样本 t1、t2、t3；

2）设定参数 k，本次实验 k=range(1，51)，对每个 k 值依次进行之后的操作；

3）初始化一个大小为 k 的按距离由大到小的优先队列，用于存储最近邻训练元组。随机从训练元组中选取 k 个（实验中选取 c1 训练集的前五组数据）元组作为初始的最近邻元

组，分别计算测试元组到 k 个元组的距离，将训练元组类标号和距离存入优先队列，形式如 queue([[label，distance]])；

4）遍历训练元组集合（c1、c2、c3），计算当前元组与测试元组的距离，将所得距离 L 与优先队列中最大距离 Lmax 进行比较。若 L>=Lmax，则舍弃该元组，遍历下一个元组；若 L<Lmax，删除优先队列中最大距离的元组，将当前训练元组存入优先队列；

5）遍历完毕，计算优先队列中 k 个元组的多数类，并将其作为测试元组的类别；

6）测试元组集（t1、t2、t3）测试完毕后计算误差率，继续设定不同的 k 值回到步骤 2）进行计算。最终获得误差率最小的 k 值。

四、实验结果与分析

1、实验结果

本次实验结果如图 4 所示。

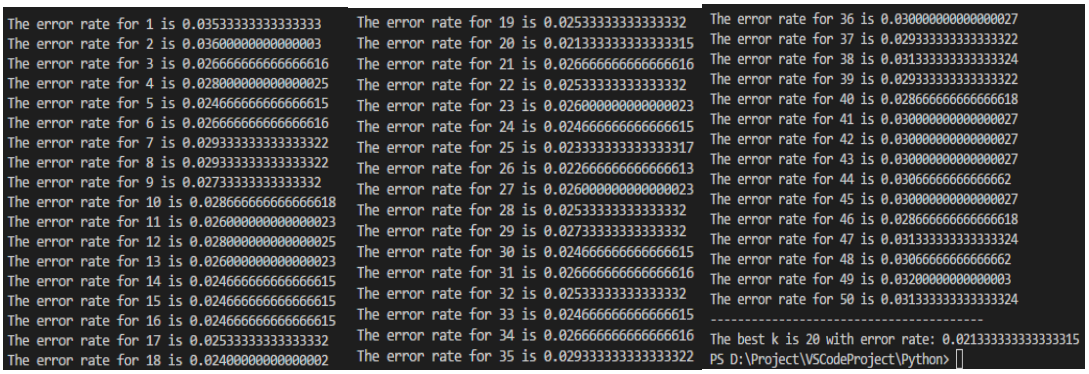


图 4 k-NN 分类结果（k=1,2,3……,50）

K-NN 测试结果：

The error rate for 1 is 0.03533333333333333
The error rate for 2 is 0.03600000000000003
The error rate for 3 is 0.026666666666666616
The error rate for 4 is 0.028000000000000025
The error rate for 5 is 0.024666666666666615
The error rate for 6 is 0.026666666666666616
The error rate for 7 is 0.029333333333333322
The error rate for 8 is 0.029333333333333322
The error rate for 9 is 0.02733333333333332
The error rate for 10 is 0.028666666666666618
The error rate for 11 is 0.026000000000000023
The error rate for 12 is 0.028000000000000025
The error rate for 13 is 0.026000000000000023
The error rate for 14 is 0.024666666666666615
The error rate for 15 is 0.024666666666666615
The error rate for 16 is 0.024666666666666615
The error rate for 17 is 0.02533333333333332

The error rate for 18 is 0.02400000000000002
The error rate for 19 is 0.02533333333333332
The error rate for 20 is 0.021333333333333315
The error rate for 21 is 0.026666666666666616
The error rate for 22 is 0.02533333333333332
The error rate for 23 is 0.026000000000000023
The error rate for 24 is 0.024666666666666615
The error rate for 25 is 0.023333333333333317
The error rate for 26 is 0.022666666666666613
The error rate for 27 is 0.026000000000000023
The error rate for 28 is 0.02533333333333332
The error rate for 29 is 0.02733333333333332
The error rate for 30 is 0.024666666666666615
The error rate for 31 is 0.026666666666666616
The error rate for 32 is 0.02533333333333332
The error rate for 33 is 0.024666666666666615
The error rate for 34 is 0.026666666666666616
The error rate for 35 is 0.029333333333333322
The error rate for 36 is 0.030000000000000027
The error rate for 37 is 0.029333333333333322
The error rate for 38 is 0.031333333333333324
The error rate for 39 is 0.029333333333333322
The error rate for 40 is 0.028666666666666618
The error rate for 41 is 0.030000000000000027
The error rate for 42 is 0.030000000000000027
The error rate for 43 is 0.030000000000000027
The error rate for 44 is 0.03066666666666662
The error rate for 45 is 0.030000000000000027
The error rate for 46 is 0.028666666666666618
The error rate for 47 is 0.031333333333333324
The error rate for 48 is 0.03066666666666662
The error rate for 49 is 0.03200000000000003
The error rate for 50 is 0.031333333333333324

The best k is 20 with error rate: 0.021333333333333315

通过将 k 的值从 k=1 到 k=50 范围内依次遍历，所得到的 K-NN 分类器最优的 k=20，相应的错误率为 2.13%

2、结果分析

当 k=1 时，分类错误率为 3.53%；

当 k=20 时，分类错误率为 2.13%，为最佳分类 k 值；

当 k=50 时，分类错误率为 3.13%。

从实验结果可以看出，k 值的选择对于最近邻决策的结果有很大影响。当 k 值较小时，意味着只有与待分类样本比较近的训练样本才会对预测结果起作用，因此分类结果容易受噪声点的影响，如 k=1 时出现的分类结果，分类误差率增加；当 k 值增大时，可以减少分类的误差，但 k 值过大的时候，如 k=50，使得分类模型过于简单，忽略了训练样本中大量的局部信息，也会导致分类错误率增加，因此也不可取。所以，综上结果分析，便得到了最佳的分类 k 值为 20

问题二

一、问题描述

由于最近邻算法需要遍历整个样本数据集，是计算密集型的，因此为了降低最近邻算法的计算量，可以只保留训练数据集中的边缘部分来进行分类。以下是该算法的一个简单描述。

```
1 begin initialize j = 0, D = data set, n = #prototypes
2         construct the full Voronoi diagram of D
3         do j ← j + 1;  for each prototype x'_j
4             Find the Voronoi neighbors of x'_j
5             if any neighbor is not from the same class as x'_j then mark x'_j
6         until j = n
7   Discard all points that are not marked
8   Construct the Voronoi diagram of the remaining (marked) prototypes
9 end
```

要求完成以下两个证明。

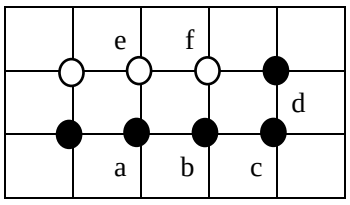
1）通过例举反例来证明该算法不会产生最小的点集。

2）建立一个顺序编辑算法，在算法中，依次考虑每个样本点，并且在考虑下一个样本点之前决定是否保留该样本点。证明所给的算法是否依赖于所考虑样本点的顺序。

二、问题求解

1、举反例证明该算法不会产生最小的点集

证明：由题目所述，可知该题需要例举一个反例来说明上述算法所得的物体边界点中存在冗余的点。因此可以作如下坐标系。



如上图，可以看作一个二分类模型，其中白点为 A 类，黑点为 B 类，图中所示的为一个样本集的分界线。以 8 邻域为例，按照题目中的算法，只要在某一点的 8 邻域内存在不属于本类的样本点，则将该邻域的中心点视为边界点保存下来。

可以看到，类别 A 的边界点为 e，f；类别 B 的边界点为 a，b，d，即图中的点 c 实际上并没有为确定边界提供有用的信息。但是如果按照本题中所给的算法时，当判断 c 点是否为边界点时，由于 c 的 8 邻域中存在 f 点，c 点也会被认为是类别 B 的边界点之一，而 c 点正是我们之前所说的边界样本点中的冗余点。

因此，可以证明上述算法不会得到一个最小的边界点集合。

2、算法设计

所设计的算法如下：

1）**begin initialize** j←0，D=data set，n=number of prototypes

2）　　do j←j+1

3)　　pick up x_j' from D

4)　　label_pre ← **classify** x_j'

5)　　**if** label_pre != label，**then** restore x_j' for D

6）　　**else** remove x_j'

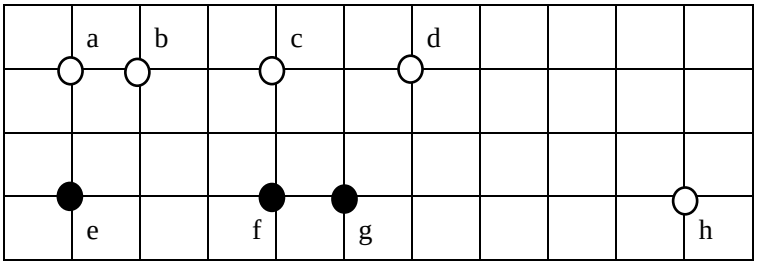
7)　　**until** j==n

8)　　**return** D

9）**end**

上述算法解释如下，首先在所有样本点集合中获取一个训练样本点，然后对该样本点进行某种分类决策，如果该样本点可以被数据集合 D 很好地分类且预测类别正确，那么它就不是边界点，给予舍弃；如果该样本点不能很好地被数据集合 D 很好地分类，即分类错误，说明该点周围存在噪声点（其他类别的点），因此该点需要被视为边界点保存下来。

另外需要证明，该算法是依赖于所考虑的样本点的顺序的。首先，如下图所示。



如上图所示，白点代表 A 类别的样本点，黑点代表 B 类别的样本点。

以 a，b，c 三点为例，如果先对 a 点进行操作，a 点的预测类别正确（离 b 点最近）从而被舍弃，而 b 点由于离 c 点最近，从而也被正确分为类别 A 而被舍弃，最终保留下了 c 点；但是，如果首先对 b 点进行上述算法操作，b 点被正确地分为 A 类被舍弃掉，之后对 a 点和 c 点的类别预测均是被预测为 B 类，从而 a、b 两点由于不能被很好地分类而被当作边界点被保留下来。

综上，可以证明自己所设计的算法是依赖于所考虑的样本点的顺序的。