机器学习概论第四次实验

PB21000302张成

1.实验原理

DPC算法是一种不需要迭代的聚类算法，通过计算每个样本点p_i的d_c领域内的点数量

$$\rho_i = \Sigma_j \chi(d_{ij} - d_c)$$

再计算

$$\delta_i = min_{j:\rho_j > \rho_i}(d_{ij})$$

对于ρ最大的点$\delta_i = max(d_{ij})$计算过程中注意记录j

随后画出$\rho - \delta$决策图

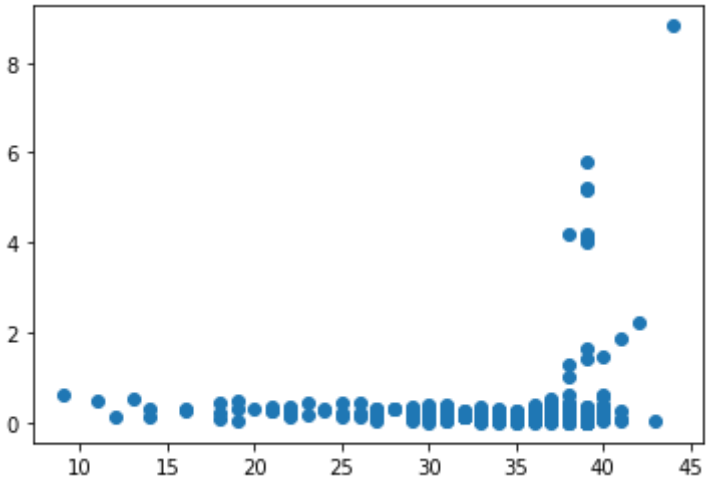


图1:R15数据集,dc=0.9

选择ρ和δ的阈值，选出阈值内的点作为中心点。

随后通过在计算δ时记录的j将每个点归到j的同一类

最后再计算异常点，通过选择每个类中和其他类中点间距再d_c以内的店，得到最大的$max_{i \in cl}(\rho_{ave}) = max_{i \in cl, j \notin cl, d_{ij} < d_c}(\rho_i + \rho_j)/2$随后将每个类中密度小于这个阈值的点作为噪声

2.代码实现

2.1数据读入

```
def read_funny_csv(filehandle):
    data = list()
    split_pattern = re.compile('\s+')
    num_columns = 2
    for line in filehandle:
        parts = split_pattern.split(line.strip())
        data.append(parts[-num_columns:])
    return data

```

```
with open("D:\\2022autumn\ml\ml_2022_f-master\lab\lab4\Datasets\Aggregation.txt", 'r') as filehandle:
    data = read_funny_csv(filehandle)
df1 = pd.DataFrame( data=data, columns=[0,1],dtype='float64')
with open("D:\\2022autumn\ml\ml_2022_f-master\lab\lab4\Datasets\D31.txt", 'r') as filehandle:
    data = read_funny_csv(filehandle)
df2 = pd.DataFrame( data=data, columns=[0,1],dtype='float64')
with open("D:\\2022autumn\ml\ml_2022_f-master\lab\lab4\Datasets\R15.txt", 'r') as filehandle:
    data = read_funny_csv(filehandle)
df3 = pd.DataFrame( data=data, columns=[0,1],dtype='float64')
```

因为这个数据集不是逗号隔开的所以需要手动分裂

2.2计算距离

```
def distance(y1,y2):
    y11=np.array(y1)
    y22=np.array(y2)
    return np.linalg.norm(y11-y22,ord=2)
def calc_dis(self):
    self.dis=np.zeros(shape=(self.m,self.m))
    for i in range(self.m):
        for j in range(self.m):
            if i>j:
                self.dis[i,j]=distance(self.ds.loc[i,:],self.ds.loc[j,:])
                self.dis[j,i]=distance(self.ds.loc[i,:],self.ds.loc[j,:])
```

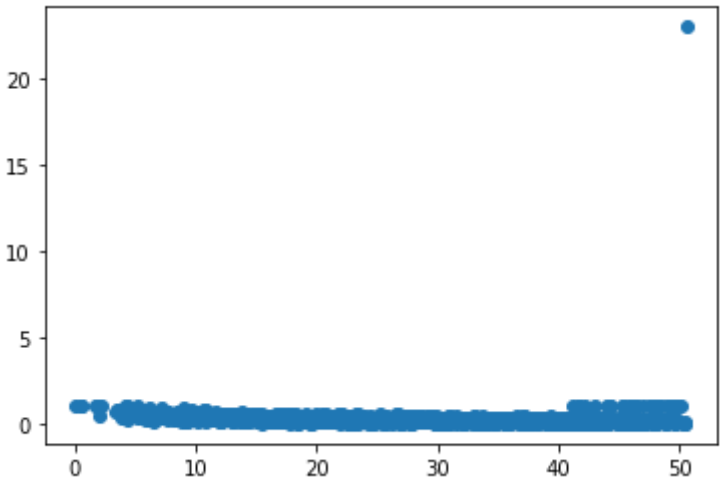
将距离存起来有利于加快后续计算，内存开销相对高一点，后续步骤都可以不预先存储距离，但那样每个新参数都要计算一次距离。怎么做主要在于时间和空间的取舍

2.3算法的中间变量

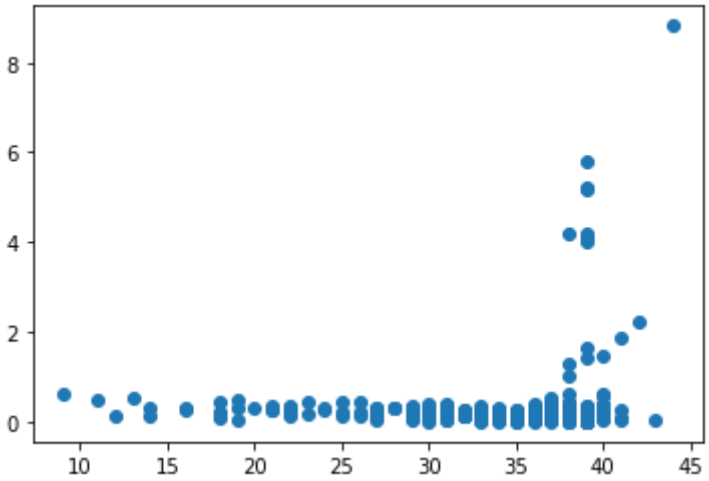
```
class clusters(object):
    def __init__(self,dataset):
        self.ds=dataset#原始数据集
        self.centers=None#记录中心点
        self.m=len(dataset.index)#数据大小
        self.d=np.ones(self.m)*float('inf')#δ
        self.master=np.zeros(self.m)#记录类
        self.ro=np.zeros(self.m)#ρ
        self.dc=0.1#超参数
        self.sort_index=None
        self.num=0
        self.brodro=None#用于确定ood
        self.dis=None#距离数组
        self.ro_threshold=20#超参数
        self.delta_threshold=10#超参数
```

2.4计算δ和ρ

计算δ和ρ的过程中如果直接使用计算好的dis矩阵，时间是很短的。如果不用，时间大约是计算dis的两倍，因为需要计算两个循环。这里也可以选择使用核函数比如高斯核，这样分类结果会更明显。同时最大值的δ像论文中那样设定的话，对整个决策图的观感有很大影响，就它一个的δ远远大于其他点，会看不清下方的点，在使用高斯核函数的时候尤为明显。



图二：R15数据集使用高斯核函数，使用论文中ρ最大的δ选择方法



图三：R15数据集使用线性核函数，使用论文中ρ最大的δ选择方法

```
    def calculate_dis_row(self):
        self.ro=np.zeros(self.m)
        for i in range(self.m):
            for j in range(self.m):
                if i>=j:
                    continue
                self.ro[i]+=math.exp(-self.dis[i,j]**2)
                self.ro[j]+=math.exp(-self.dis[i,j]**2)
                # if (self.dis[i,j]<self.dc):
                #     self.ro[i]+=1
                #     self.ro[j]+=1
                #线性核的做法
        self.biggest_ro=np.max(self.ro)
        self.sort_index=np.argsort(-self.ro)
        self.d=np.ones(self.m)
        self.d[self.sort_index[0]]=0
        self.master[self.sort_index[0]]=self.sort_index[0]
        for i in range(self.m):
```

```
            ii=self.sort_index[i]
            for j in range(self.m):
                jj=self.sort_index[j]
                if j==0:
                    self.d[jj]=max(self.d[jj],self.dis[ii,jj])
                    continue
                t=self.dis[ii,jj]
                if i<j and t<self.d[jj]:
                    self.d[jj]=t
                    self.master[jj]=ii
```

2.4选择聚类中心和聚类

通过

```
        mm =np.where(self.ro>self.ro_threshold)
        nn=np.where(self.d>self.delta_threshold)
        self.centers=np.intersect1d(mm,nn)
```

来选择同时满足$\rho_i > \rho_0$和$\delta_i > \delta_0$的元素，当然也有通过选择$\rho * \delta$较大者，但我并不认为这是个好主意，因为不太好控制不把一个类分成多个，比如一些类密度很高，一些较低，为了分出密度较低的类，乘积的阈值会较低，最终可能密度高的类中出现多个分类中心。这个分类问题中，δ的大小往往是差之毫厘，谬之千里，而ρ只需要是一个差不多的值

通过

```
def determine_every_point_cluster(self):
        for i in self.centers:
            self.master[i]=i
        for i in range(self.m):
            ii = self.sort_index[i]
            if ii in self.centers:
                continue
            else:
                self.master[ii]=self.master[(int)(self.master[ii])]
```

将每个元素的类改成最近的比它密度大的元素的类，因为是密度从大到小进行，可以保证没有除了center以外的类。

```
def determine_every_point_cluster(self):
        for i in self.centers:
            self.master[i]=i
        for i in range(self.m):
            ii = self.sort_index[i]
            if ii in self.centers:
                continue
            else:
                self.master[ii]=self.master[(int)(self.master[ii])]
```

2.5 分类去除噪声

通过计算临近其他类的平均密度最大点的密度作为基准，剔除所有低于这个标准的点。

```
def reduce_noise(self):
        for i in range(self.m):
            for j in range(self.m):
                if i>j and (self.dis[i,j]<=self.dc) and
self.master[i]!=self.master[j]:
                    ii=self.centers==self.master[i]
                    jj=self.centers==self.master[j]
                    ro_ave=(self.ro[i]+self.ro[j])/2
                    if (ro_ave>self.brodro[ii]):
                        self.brodro[ii]=ro_ave
                    if (ro_ave>self.brodro[jj]):
                        self.brodro[jj]=ro_ave
        for i in range(self.m):
            ii=self.centers==self.master[i]
            if (self.ro[i]<self.brodro[ii]):
                self.master[i]=-1
```

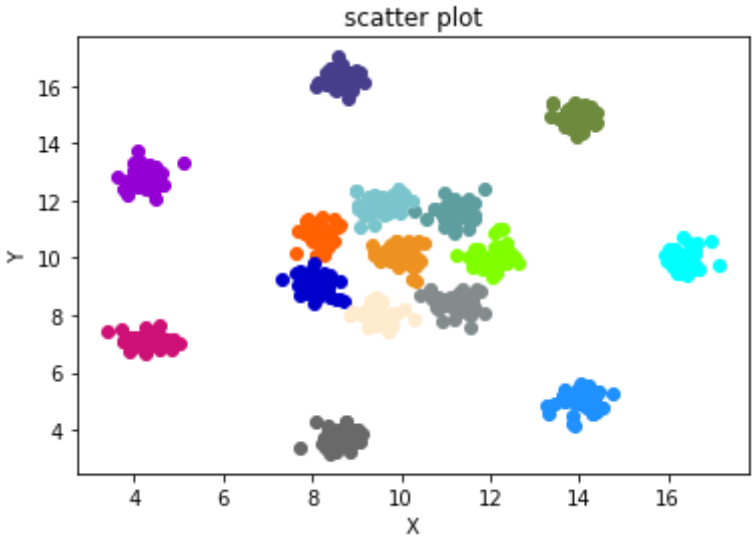
但是由于

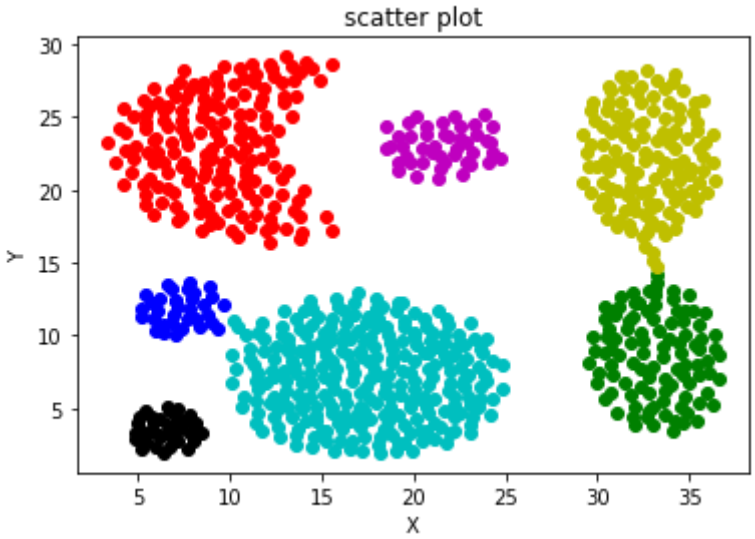
3.结果展示

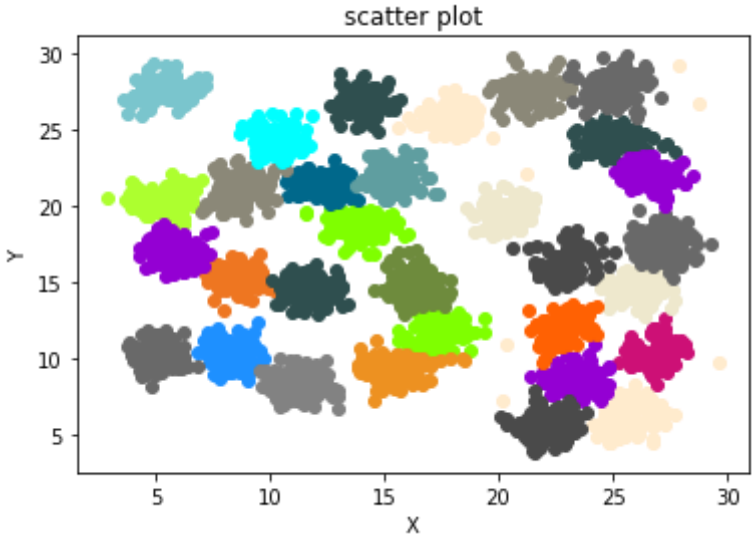
最佳参数展示

数据集	参数	DBI	DBI after reducing noise
Aggregation	dc=2 ρ0=5 δ0=6,linear kernel	0.50360836	1.30297795
D31	dc=0.15,ρ0=20,δ0=0.99,gaussian kernel	0.601807663	1.515299956
R15	dc=0.45,ρ0=30,δ0=1,linear kernel	0.31493561	0.8916153980752852

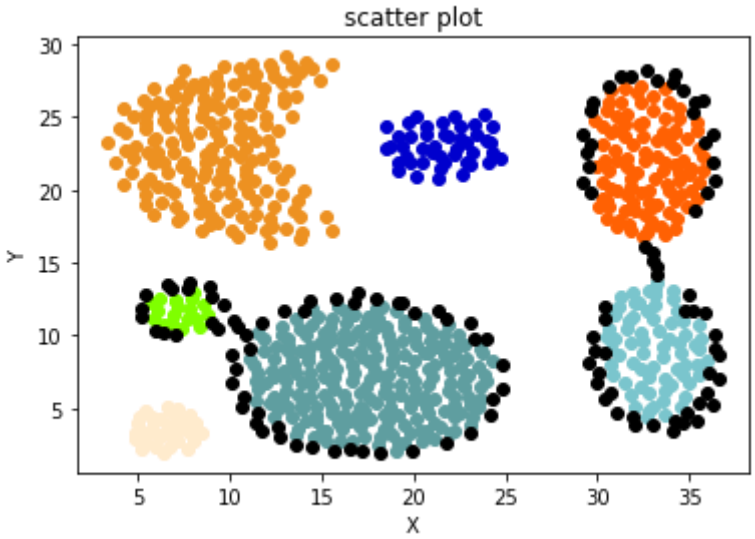
分类图片：

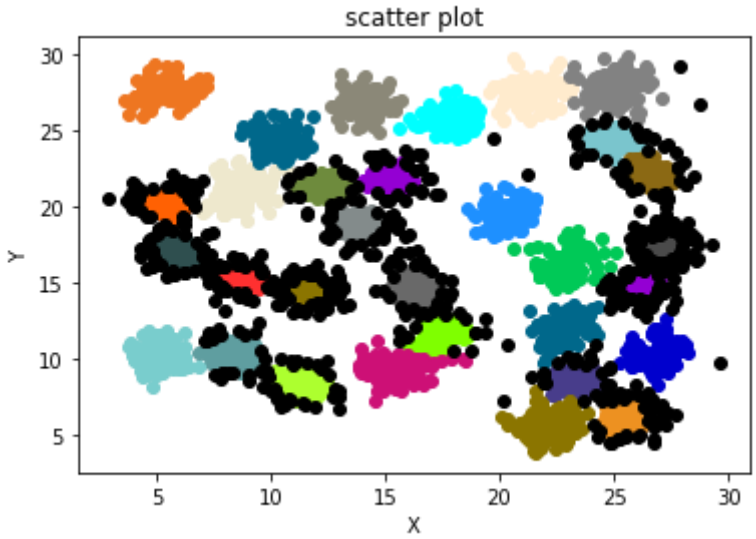


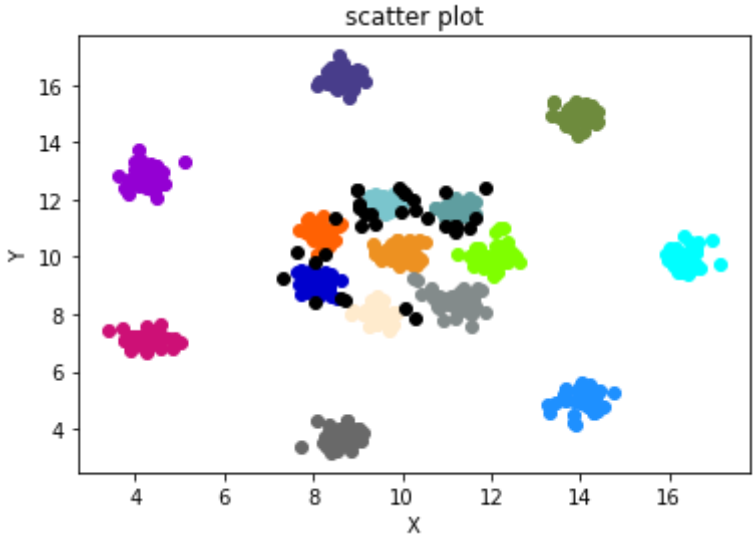




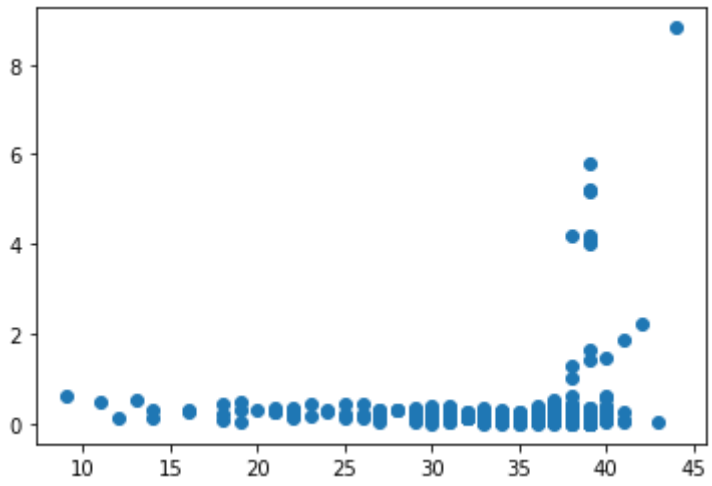
去除噪声后的图片（黑色是被判定为噪声的点）：



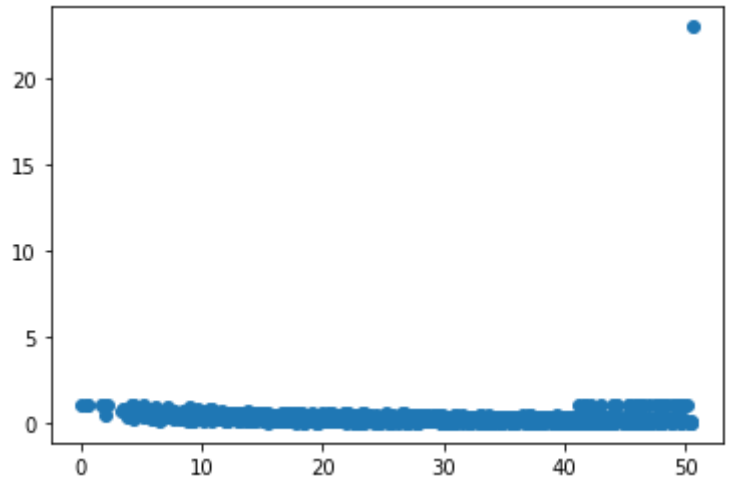

scatter plot
Y
X
30
25
20
15
10
5
5
10
15
20
25
30


scatter plot
Y
X
16
14
12
10
8
6
4
4
6
8
10
12
14
16

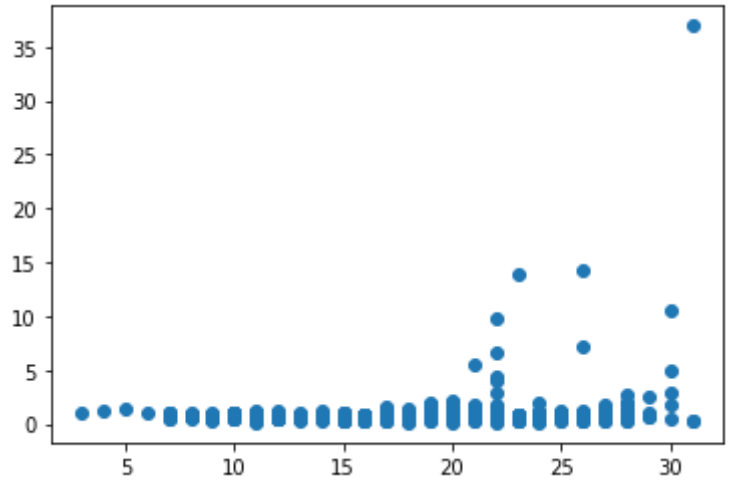
决策图：


8
6
4
2
0
10
15
20
25
30
35
40
45

R15决策图



D31



aggregation

中心点编号：

D31 :[31，83，188，292，393，484，556，688，789，837，963，1077，1132，1273，1373，1444，1520，1677，1766，1820，1933，2006，2181，2227，2314，2492，2576，2683，2740，2804，2972，3016]

R15 :[35, 40, 104, 147, 166, 223, 275, 319, 344, 368, 404, 449, 490, 538, 593]

aggregation:[44, 191, 347, 552, 602, 743, 767]