第五次作业

姓名：樊宇

学号：2021E8018782022

Part 1

Question 1

原理：

寻找一组方差较大的方向，将原始数据在该方向进行投影，即将数据在新的坐标系下进行表示，保留少数在方差最大方向上的投影，达到数据变换，尽可能地保留原始数据信息和降维地目的

学习模型（最大可分性观点）：

要使投影后的样本点尽可能的分开，设投影后的样本点为：

$$y_i = W^T x_i \in \mathbb{R}^m$$

由于数据是零均值化的，则：

$$\sum y_i = W^T \sum x_i = 0$$

因此投影后的样本点的协方差为：

$$\sum W^T x_i x_i^T W = W^T X X^T W$$

要使数据具有最大可分性，即：

$$\max_{W\ in \mathbb{R}^{m\times d}} tr(W^T X X^T W) \qquad s.t. W^T W = I$$

问题转化为约束最优化问题，有：

$$X X^T W = \lambda W$$

选择一组特征值最大的对应的特征向量作为投影矩阵即可

算法步骤：

- 计算数据均值：$\bar{x} = \frac{1}{n} \sum x_i$
- 计算数据的协方差矩阵：$C = \frac{1}{n} \sum (x_i - \bar{x})(x_i - \bar{x})^T$
- 对矩阵C进行特征值分解，并取最大的m个特征值（$\lambda_1 \geq \lambda_2, ... \geq \lambda_m$）对应的特征向量组合成投影矩阵$W = [w_1, ..., w_m] \in \mathbb{R}^{d\times m}$
- 对每一个数据进行投影：$y_i = W^T x_i$

Question 2

原理：寻找一组投影方向，使样本在投影之后（即在新坐标系下）类内样本点尽可能靠近，类间样本点尽可能相互远离，提升样本表示的分类鉴别能力

学习模型：

两类：

设样本集为D，X_i, μ_i, Σ_i分别表示第i类的实例集合、均值向量、协方差矩阵。

将数据投影后的两类的中心分别为$w^T\mu_0, w^T\mu_1$，两类样本的协方差分别为$w^T\Sigma_0 w, w^T\Sigma_1 w$

要使同类样本的投影点尽可能接近，可让同类样本投影点的协方差尽可能小，即$w^T\Sigma_0 w + w^T\Sigma_1 w$ $尽可能小

要使异类样本的投影点尽可能原理，可让类中心点之间的距离尽可能打，即$||w^T\mu_0 - w^T\mu_1||^2$尽可能打

令目标函数为：

$$
\begin{aligned}
J &= \frac{||w^T\mu_0 - w^T\mu_1||_2^2}{w^T\Sigma_0 w + w^T\Sigma_1 w} \\
&= \frac{w^T(\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w}{w^T(\Sigma_0 + \Sigma_1)w}
\end{aligned}
$$

同时定义类内散度矩阵：

$$S_w = \Sigma_0 + \Sigma_1$$

类间散度矩阵：

$$S_b = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T$$

于是上述求解问题也可转化为约束最优化问题：

$$max \frac{w^T S_b w}{w^T S_w w} \qquad s.t. w^T w = 1$$

有：

$$S_w^{-1} S_b w = \lambda w$$

选择一组特征值最大的对应的特征向量作为投影矩阵即可

多类：

类内散度矩阵：

$$S_w = \sum_{j=1}^{c} S_{wj}$$

其中：

$$S_{wj} = \sum_{x \in X_j} (x - \mu_j)(x - \mu_j)^T$$

$$\mu_j = \frac{1}{n_j} \sum x$$

类间散度矩阵：

$$S_b = \sum n_j(\mu_j - \mu)(\mu_j - \mu)^T$$

令：

$$max\ J = \frac{|w^T S_b w|}{w^T S_w w} \qquad s.t.\ w^T w = I$$

有解为：

$$S_b w = \lambda S_w w$$

选择一组特征值最大的对应的特征向量作为投影矩阵即可

Question 3

在数据是光滑的、密集采样以及无自交叉的假设之下

流形学习的基本思想是在高维空间相似的数据点，映射在低维空间也是相似的

LLE的基本思想：

给定数据集，通过最近邻等方式构造一个数据图。然后在每一个局部区域，高维空间种的样本线性重构关系在低维空间中均得以保持

Isomap的基本思想：

给定数据集，通过最近邻等方式构造一个数据图。然后计算任意两个点之间的最短路径（即测地距离），对于所有的任意两个点对，期望在低维空间中保持其测地距离

LE的基本思想：

给定数据集，通过最近邻等方式构造一个数据图。然后在每一个局部区域，计算点与点之间相似度，期望点对相似度在低维空间中也得到保持

Question 4

语音特征提取：

MFCCs：

- 对分帧后的语音信号进行傅里叶变换，保留幅度谱，丢弃相位谱
- 根据梅尔刻度，利用频域三角窗对独立也幅度谱进行求和
- 对求和后的幅度取对数
- 离散余弦变换，对取对数后的幅度信号进行离散余弦变换，得到MFCCs特征

文本特征提取：

向量空间模型：

- 一个维度对应于一个此项，如果一个词项出现在一篇文档中，它在向量中的值是非零的，否则为零

TF-IDF：

- 语料库即为D，即一个由若干文档组成的集合；文档记为d；词语记为t
- 词频TF(t, d)：在文档d中词语t出现的次数
- 文档频率DF(t, D)：语料库D中包含词语t的文档次数

- 逆向文档频率IDF(t, D)：衡量语料库D中词语t提供的信息量
- 词频-逆向文档频率，利用在语料库中的词语所含的信息量及其词频作为其特征

Word2Vec：

- 利用一个连续想想来表示一个词项，相似的单词具有相似的向量表示

视觉特征提取：

SIFT：

- 图像中可辨识度高的点，容易在同一物体的不同图像中重复出现
- 尺度不变、旋转不变、对视角变化、光照变化鲁棒
- 为特征点计算"个性签名"，区分是否属于同一个物理点

Haar：

- 一个Haar特征由一组方形滤波器组成
- 滤波器响应值为对应区域内像素值的和
- 一个Haar特征的响应值为白色滤波器响应值减去灰色滤波器响应值
- 三个臭皮匠，顶个诸葛亮
- 继承大量Haar特征的判定结果，来模式分类

HoG：

- HoG计算的是图像梯度方向直方图，本质和SIFT特征描述子一样，但空间统计直方图方式不一样

Question 5

穷举法：

从给定的d个特征中，挑选出最优特征子集，若采用穷举法，需要遍历2^d个子集

分支限界法：

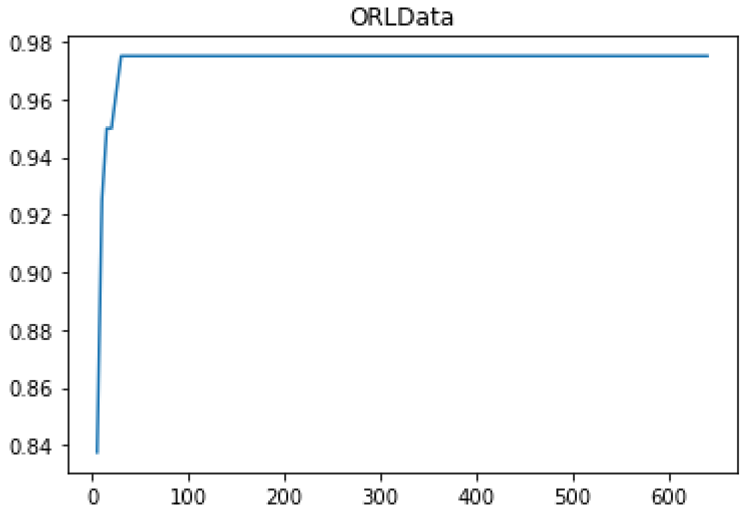
将所有可能的特征选择组合以数的形式进行表示，采用分支限界方法对树进行搜索，使得搜索过程尽早达到最优解，而不必搜索整个树

基本前提：

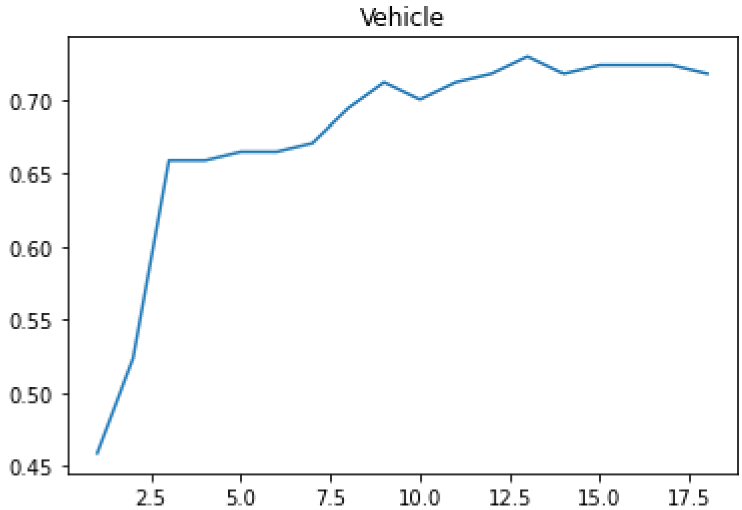
特征的评价准则盘踞对特征具有单调性，即特征增多时，判据值不会减少

Part 2

PCA+KNN：

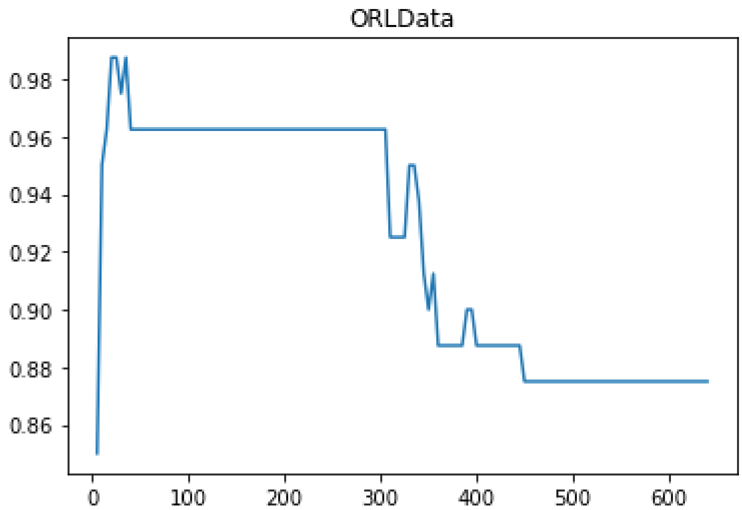


ORLData
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0
100
200
300
400
500
600

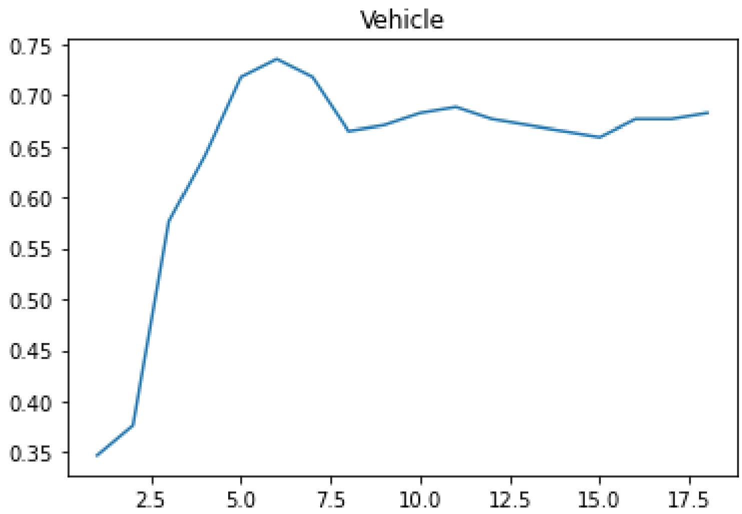


Vehicle
0.70
0.65
0.60
0.55
0.50
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5

LDA+KNN：



ORLData
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
100
200
300
400
500
600


Vehicle
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2.5
5.0
7.5
10.0
12.5
15.0
17.5

```python
import scipy.io as scio
```

```python
data__ORLD_File = './data/ORLData_25.mat'
data_vehicle_File = './data/vehicle.mat'
```

```python
data_ORL = scio.loadmat(data__ORLD_File)
data_vehicle = scio.loadmat(data_vehicle_File)
```

```python
data_ORL
```

```
{'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN, Created on: Fri Sep 15
22:49:51 2006',
 '__version__': '1.0',
 '__globals__': [],
 'ORLData': array([[ 42,  80,  57, ..., 122, 118, 121],
        [ 46,  67,  40, ..., 123, 117, 128],
        [ 49,  70,  61, ..., 126, 134, 124],
        ...,
        [ 43,  37,  32, ...,  42,  97,  36],
        [ 47,  37,  29, ...,  38,  90,  36],
        [  1,   1,   1, ...,  40,  40,  40]], dtype=uint8)}
```

```python
data = data_ORL['ORLData'][:-1,:]
labels = data_ORL['ORLData'][-1,:]
```

```python
data.shape
```

```
(644, 400)
```

```python
labels.shape
```

```
(400,)
```

```python
import numpy as np
```

```
index = np.arange(400)
np.random.shuffle(index)
```

```
train_data = data[:, index[:320]]
train_labels = labels[index[:320]].reshape(-1, 1)
```

```
test_data = data[:, index[320:]]
test_labels = labels[index[320:]].reshape(-1, 1)
```

```
train_data.shape
```

```
(644, 320)
```

```
train_labels.shape
```

```
(320, 1)
```

```
test_data.shape
```

```
(644, 80)
```

```
test_labels.shape
```

```
(80, 1)
```

```
mu = np.mean(train_data, 1, keepdims = True)
```

```
sigma = (train_data - mu).dot((train_data - mu).T)
```

```python
def PCA(sigma, N):
    eigv, eigvec = np.linalg.eig(sigma)
    index = np.argsort(eigv)
    index = index[::-1]
    threshold = N# int(0.9 * len(eigv))
    eigvec = eigvec[:, index[0: threshold]]
    eigvec = eigvec.real
    return eigvec
```

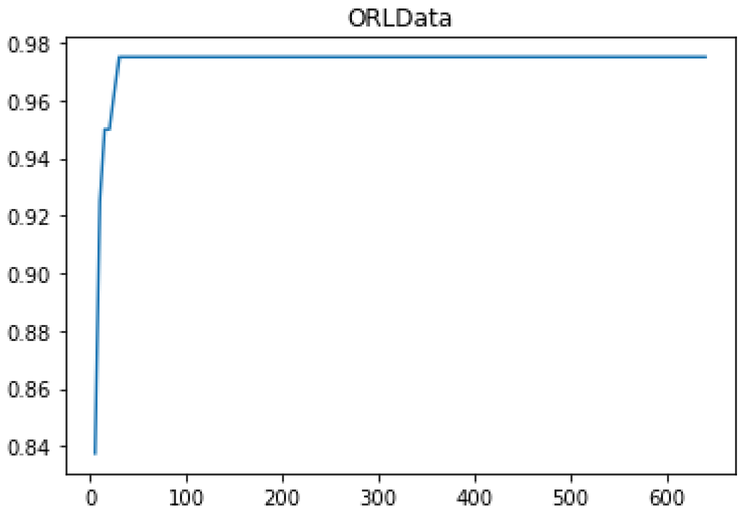
```python
def discrimination(train_data, train_labels, test_data, test_labels, N):
    convertM = PCA(sigma, N)
    train_data = convertM.T.dot(train_data)
    test_data = convertM.T.dot(test_data)
    length = test_data.shape[-1]
    correct = []
    for i in range(length):
        dist = np.sum(np.square(train_data - test_data[:, i].reshape(-1, 1)),
axis = 0)
        index = np.argmin(dist)
        if(train_labels[index] == test_labels[i]):
            correct.append(i)
    return len(correct) / length
```

```python
index = []
acc = []
for i in range(128):
    N = (i + 1) * 5
    index.append(N)
    acc.append(discrimination(train_data, train_labels, test_data, test_labels,
N))
```

```python
import matplotlib.pyplot as plt
```

```python
plt.plot(index, acc)
plt.title('ORLData')
```

```
Text(0.5, 1.0, 'ORLData')
```




```
data = data_vehicle['UCI_entropy_data']['train_data'][0, 0]
```

```
index = np.arange(846)
np.random.shuffle(index)
```

```
train_data = data[0: -1, index[:int(0.8 * 846)]]
train_labels = data[-1 , index[:int(0.8 * 846)]].reshape(-1, 1)
```

```
test_data = data[0: -1, index[int(0.8 * 846):]]
test_labels = data[-1 , index[int(0.8 * 846):]].reshape(-1, 1)
```

```
train_data.shape
```

```
(18, 676)
```

```
train_labels.shape
```

```
(676, 1)
```

```
test_data.shape
```

```
(18, 170)
```

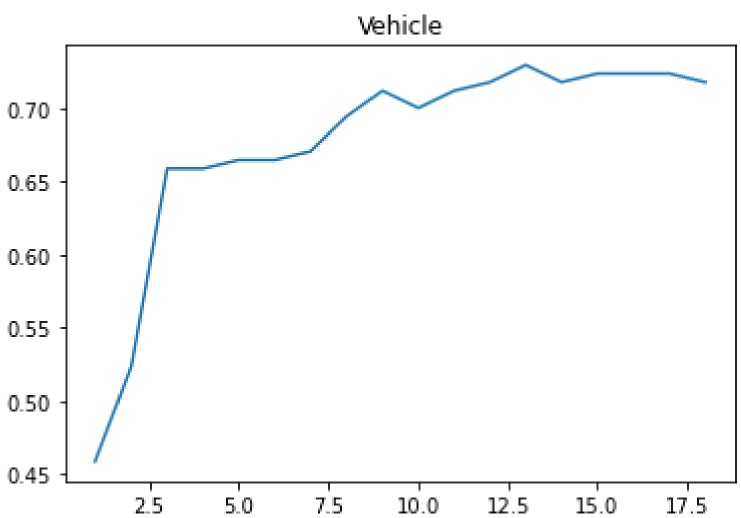
```
test_labels.shape
```

```
(170, 1)
```

```
mu = np.mean(train_data, 1, keepdims = True)
sigma = (train_data - mu).dot((train_data - mu).T)
```

```
index = []
acc = []
for i in range(18):
    N = i + 1
    index.append(N)
    acc.append(discrimination(train_data, train_labels, test_data, test_labels,
N))
plt.plot(index, acc)
plt.title('Vehicle')
```

```
Text(0.5, 1.0, 'Vehicle')
```




```
```

```python
import scipy.io as scio
from scipy import linalg
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```python
data__ORLD_File = './data/ORLData_25.mat'
data_vehicle_File = './data/vehicle.mat'
data_ORL = scio.loadmat(data__ORLD_File)
data_vehicle = scio.loadmat(data_vehicle_File)
```

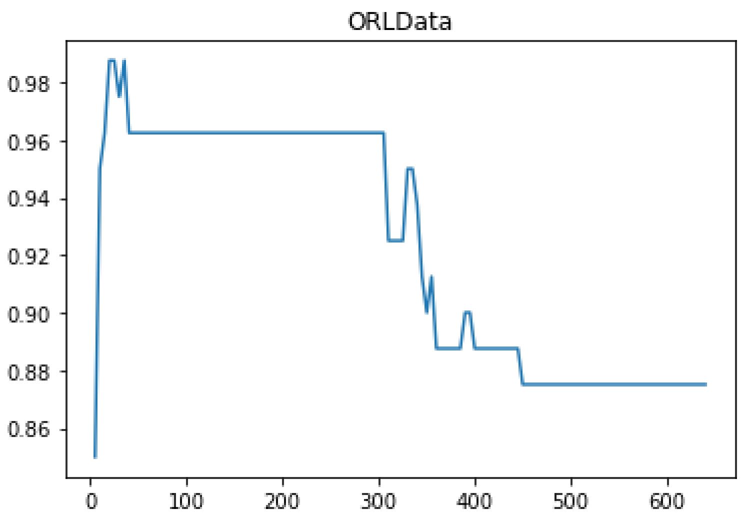
```python
def LDA(train_data, train_labels, c, N):
    # c是类别总数
    dim = train_data.shape[0]
    S_w = np.zeros(shape = (dim, dim))
    S_b = np.zeros(shape = (dim, dim))
    mu = np.mean(train_data, axis = 0)
    for i in range(c):
        index = np.argwhere(train_labels.reshape(-1) == (i+1)).reshape(-1)
        mu_i = np.mean(train_data[:, index], axis = 1).reshape(-1, 1)
        S_w += (train_data[:, index] - mu_i).dot((train_data[:, index] -
mu_i).T)
        S_b += len(index) * (mu_i - mu).dot((mu_i - mu).T)
    lamb = 0.05 * np.identity(S_w.shape[0])
    eigv, eigvec = linalg.eig(S_b, S_w + lamb)
    index = np.argsort(eigv).reshape(-1)
    index = index[::-1]
    threshold = N
    eigvec = eigvec[:, index[0: threshold]]
    eigvec = eigvec.real
    return eigvec
```

```python
def discrimination(train_data, train_labels, test_data, test_labels, c, N):
    convertM = LDA(train_data,train_labels, c, N)
    train_data = convertM.T.dot(train_data)
    test_data = convertM.T.dot(test_data)
    length = test_data.shape[-1]
    correct = []
    for i in range(length):
        dist = np.sum(np.square(train_data - test_data[:, i].reshape(-1, 1)),
axis = 0)
        index = np.argmin(dist)
        if(train_labels[index] == test_labels[i]):
            correct.append(i)
    return len(correct) / length
```

```python
data = data_ORL['ORLData'][:-1,:]
labels = data_ORL['ORLData'][-1,:]
index = np.arange(400)
np.random.shuffle(index)
train_data = data[:, index[:320]]
train_labels = labels[index[:320]].reshape(-1, 1)
```

```
test_data = data[:, index[320:]]
test_labels = labels[index[320:]].reshape(-1, 1)
index = []
acc = []
for i in range(128):
    N = (i + 1) * 5
    index.append(N)
    acc.append(discrimination(train_data, train_labels, test_data, test_labels,
40, N))
plt.plot(index, acc)
plt.title('ORLData')
```

```
Text(0.5, 1.0, 'ORLData')
```




```
data = data_vehicle['UCI_entropy_data']['train_data'][0, 0]
index = np.arange(846)
np.random.shuffle(index)
train_data = data[0: -1, index[:int(0.8 * 846)]]
train_labels = data[-1 , index[:int(0.8 * 846)]].reshape(-1, 1)
test_data = data[0: -1, index[int(0.8 * 846):]]
test_labels = data[-1 , index[int(0.8 * 846):]].reshape(-1, 1)
index = []
acc = []
for i in range(18):
    N = i + 1
    index.append(N)
    acc.append(discrimination(train_data, train_labels, test_data, test_labels,
4, N))
plt.plot(index, acc)
plt.title('Vehicle')
```

```
Text(0.5, 1.0, 'Vehicle')
```