林之阳 3118311071

1. **Principal Component Analysis 算法填空并排序**

   输入：数据集(维度>k)，降到的维度数k

   输出：PCA降维后的样本集D

   C 原始数据进行标准化,均值为(0),方差为1　A 计算协方差矩阵Cov D计算协方差矩阵的(特征值)和(特征向量) E 选取特征值中的前(K)列（假设要降为K维） B 用特征(向量)矩阵乘原数据得到降维结果

2. **Linear Discriminant Analysis　算法填空并排序**

   输入：数据集，降到的维度数d

   输出：LDA降维后的样本集D

   B计算类(内)散度矩阵S_w E计算类(间)散度矩阵S_b A计算矩阵$S_w^{-1}S_w$ D计算$S_w^{-1}S_w$的最大的(d)个特征值和对应的(特征向量),得到投影矩阵W C得到输出的样本

3. **Canonical Correlation Analysis 算法填空并排序**

   输入：样本X和Y数量都为N ，其中X和Y的维度都> 1

   输出：X,Y的相关系数,X和Y的线性系数向量w和v

   E 原始数据进行标准化,均值为(0),方差为(1) B 计算X的S_{xx},Y的S_{yy}，X和Y的(S_{xy}), Y和X的(S_{yx}) D 计算矩阵$S_{xx}^{-1}S_{xy}S_{yy}^{-1}S_{yx}$, $S_{yy}^{-1}S_{yx}S_{xx}^{-1}S_{xy}$ C 对矩阵$S_{xx}^{-1}S_{xy}S_{yy}^{-1}S_{yx}$, $S_{yy}^{-1}S_{yx}S_{xx}^{-1}S_{xy}$进行(奇异值分解) A 由特征值计算X和Y的线性系数向量w和v

4. **PCA**

```python
# -*- coding: utf-8 -*-
# @Time     : 2018/10/10 9:47
# @Author   : vickylzy
import numpy as np


def normalized(msg):
    dataNormed = (msg - msg.mean(0)) / msg.std(0)
    return dataNormed


# @para msg_received: shape(a,b)
# @para keep 保留维度数量
def pri_com_ana(msg_received, keep):
    component = keep
    selected = range(-component, 0)  # 降维保留的维度

    # normalized
    msgNormalized = normalized(msg_received)  # shape = (a, b)
    # Covariance matrix
    covMatrix = np.cov(msgNormalized.T)  # shape = (b, b)
    print(covMatrix)
    # 提取前n个特征值特征向量
```

```
    eigVal, eigVector = np.linalg.eig(covMatrix)  # eigVal.shape = (b,1) eigVector.shape = (b,b)
    sortList = np.argsort(eigVal)
    print(sortList[selected])
    selectedVector = eigVector[:, sortList[selected]]  # selectedVector.shape=(b,keep)
    # 产生主成分结果
    return np.dot(msg_received, selectedVector)  # shape = (a, keep)
```

5. **LDA**

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

import numpy as np


# para: msg: shape=(a,b)
def lin_dis_ana(msg):
    lda = LinearDiscriminantAnalysis(n_components=2)
    target = np.zeros((1024,))  # shape=(a,1)
    for i in range(0, 1024):
        target[i] = i % 4
    return lda.fit_transform(msg, target)  # shape=(a,2)
```

6. **CCA**

```
from sklearn.cross_decomposition import CCA


def can_cor_ana(msg1, msg2):
    cca = CCA(n_components=1)
    cca.fit(msg1, msg2)
    w, v = cca.transform(msg1, msg2)
    return w, v
```