1

降维的一个主要目的就是防止过拟合，维度越低，模型的假设空间越简单。

2

输入：样本集 $D=\{x_1,x_2,\ldots,x_n\}$; 低维空间维数 m 过程:
1: 对所有样本进行中心化: $x_i \leftarrow x_i-\frac{1}{n} \sum_{i=1}^n x_i$
2: 计算样本的协方差矩阵 $X X^T$
3：对协方差矩阵 $X X^T$ 做特征值分解
4: 取最大的 m 个特征值所对应的单位特征向量 $w_1,w_2,\ldots,w_m$ 输出: 投影矩阵
$W=(w_1,w_2,\ldots,w_m)$

3

设正交基 u_j ，数据点 x_i 在孩基底上的投影距离为 $x_i^T \cdot u_j$ ，所以所有数据在该基底 上投影的方差 J_j 为:

$$J_j=\frac{1}{m} \sum_{i=1}^m\left(x_i^T u_j-x_{center}^T u_j\right)^2$$

其中: m 为样本数量，在数据运算之前对数据 x 进行 0 均值初始化，即 $x_{\text{center}}=0$ ，从而:

$$J_j=\frac{1}{m} \sum_{i=1}^m\left(x_i^T u_j\right)^2=\frac{1}{m} \sum_{i=1}^m\left(u_j^T x_i \cdot x_i^T u_j\right)=u_j^T \cdot \frac{1}{m} \sum_{i=1}^m\left(x_i x_i^T\right) \cdot u_j$$

所以:

$$J_j=u_j^T \cdot \frac{1}{m}\left(x_1 x_1^T+x_2 x_2^T+\cdots+x_m x_m^T\right) \cdot u_j=u_j^T \cdot \frac{1}{m}\left(\begin{bmatrix}x_1 & \cdots & x_m\end{bmatrix} \cdot \begin{bmatrix}x_1 \\ \vdots \\ x_m\end{bmatrix}\right) \cdot u_j==\frac{1}{m} u_j^T X X^T u_j$$

由于 $\frac{1}{m} X X^T$ 为常数，这里假设 $S=\frac{1}{m} X X^T$ ，则： $J_j=u_j^T \cdot S \cdot u_j$ ，根据PCA目标，我 们需要求解 J_j 最大时对应的 u_j

$$\begin{gathered}J_j=u_j^T S u_j \\ \text{s.t. } u_j^T u_j=1\end{gathered}$$

则构造函数:

$$F\left(u_j\right)=u_j^T S u_j+\lambda_j\left(1-u_j^T u_j\right)$$

求解 $\frac{\partial F}{\partial u_j}=0$ ，得:

$$2S \cdot u_j - 2\lambda_j \cdot u_j = 0 \Rightarrow Su_j = \lambda_j u_j$$

则：当 u_j、λ_j 分别为 S 矩阵的特征向量、特征值时， J_j 有极值，把上述结果带回公式 得:

$$J_{j_m} = u_j^T \lambda_j u_j = \lambda_j$$

所以对于任意满足条件的正交基，对应的数据在上面投影后的方差值为S矩阵的特征向量，从而:

$$J_{\max} = \sum_{j=1}^{k} \lambda_j, \lambda \text{ 从大到小排序}$$

所以投影正交基为 S 的特征向量中的前 k 个最大特征值对应的特征向量。

4

特征脸是指用于机器视觉领域中的人脸识别问题的一组特征向量。

1. 准备一个训练集的人脸图像。 构成训练集的图片需要在相同的照明条件下拍摄的,并将所有图像的眼睛和嘴对齐。他们还必须在预处理阶段就重采样到一个共同的像素分辨率 (RxC) 。简单地将原始图像的每一行的像素串联在一起，产生一个具有RxC个元素的行向量,每个图像被视为一个向量。假定所有的训练集的图像被存储在一个单一 的矩阵T中，矩阵的每一行是一个图像。
2. 减去均值向量. 均值向量a要首先计算,并且 T中的每一个图像都要减掉均值向量。
3. 计算协方差矩阵S的特征值和特征向量。 每一个特征向量的维数与原始图像的一 致, 因此可以被看作是一个图像。 因此这些向量被称作特征脸。他们代表了图像与均值图像差别的不同方向。通常来说，这个过程的计算代价很高(如果可以计算的话)。
4. 选择主成分。一个DxD的协方差矩阵会产生D个特征向量,每一 个对应RxC图像空间中的一个方向。具有较大特征值的特征向会被保留下来，一般选择最大的N个， 或者按照特征值的比例进行保存，如保留前95%。

5

PCA旨在找出一套数据中能够表示最多方差的线性组合,而CCA旨在找出两套数据中能够最大程度表示其相关性的线性组合.

6

输入: 各为 m 个的样本 X 和 Y ， X 和 Y 的维度都大于 1
输出: X, Y 的相关系数 ρ, X 和Y的线性系数向量a和b

1. 计算X的方差 S_{XX} ， Y的方差 S_{YY}，X 和Y的协方差 S_{XY} ， Y和X的协方差 $S_{YX} = S_{XY}^T$
2. 计算矩阵 $M = S_{XX}^{-1/2} S_{XY} S_{YY}^{-1/2}$
3. 对矩阵 M 进行奇异值分解，得到最大的奇异值 ρ ， 和最大奇异值对应的左右奇异向量 u, v
4. 计算X和Y的线性系数向量 a 和b, $a = S_{XX}^{-1/2} u, b = S_{YY}^{-1/2} v$

7

```
from math import sqrt

import numpy as np


class CCA:

    def __init__(self, x_dataset, y_dataset):

        # 需要对数据转置一下，才能跟上文对上

        self.x_dataset = np.array(x_dataset).T

        self.y_dataset = np.array(y_dataset).T

    def fit(self):

        A = []

        for sample in self.x_dataset:

            A.append(list(sample))

        for sample in self.y_dataset:

            A.append(list(sample))

        # 构造上面提到的A矩阵

        A = np.array(A)

        for i in range(A.shape[0]):

            aver = np.mean(A[i])

            std = np.std(A[i])

            A[i] = (A[i] - aver) / std



        Cov = np.cov(A, bias = True)

        n = self.x_dataset.shape[0]

        R_11 = np.matrix(Cov[:n, :n])

        R_12 = np.matrix(Cov[:n, n:])
```

```
        R_12 = np.matrix(Cov[:n, n:])

        R_21 = np.matrix(Cov[n:, :n])

        R_22 = np.matrix(Cov[n:, n:])

        M = np.linalg.inv(R_11) * R_12 * np.linalg.inv(R_22) * R_21

        N = np.linalg.inv(R_22) * R_21 * np.linalg.inv(R_11) * R_12

        eig1, vector1 = np.linalg.eig(M)

        data = []

        for i in range(len(eig1)):

            if abs(eig1[i]) < 1e-10:

                continue

            rho = np.round(sqrt(eig1[i]), decimals = 5)

            alpha = np.round(vector1[:, i], decimals = 5)

            k = 1 / (alpha.T * R_11 * alpha)

            alpha *= sqrt(k)

            beta = np.round(np.linalg.inv(R_22) * R_21 * alpha / rho, decimal

            data.append((rho, alpha, beta))

        data.sort(key = lambda x: x[0], reverse = True)

        return data
```