降维

W.J.Z

2019.4.17

1 MDS

MDS(多维缩放) 是一种经典的线性降维。假设 m 个样本在原始空间的距离矩阵为 $D\epsilon R^{m\times m}$, 其第 i 行第 j 列的元素 $dist_{ij}$ 为样本 x_i 到 x_j 的距离。$Z\epsilon R^{d^{'}\times m}, d^{'}<d$ 为降维到 $d^{'}$ 空间的坐标，令 $B=Z^TZ$, 其中 B 为降维后样本的内积矩阵，$b_{ij}=z_i^Tz_j$.

数学模型：任意两个样本在 $d^{'}$ 空间的欧式距离等于原始空间中的距离。

$$\begin{aligned} dist_{ij} &= \|z_i\|^2+\|z_j\|^2-2z_i^Tz_j \\ &= b_{ii}+b_{jj}-2b_{ij} \end{aligned} \tag{1}$$

公式 1为数学模式，如何对数学模型求解释关键性问题，学数学的就牛逼在这里。求解 b_{ij} 也就是求矩阵 B，已知 $dist_{ij}$，未知 b_{ii}, b_{jj}，接下来就是使用数学手段将未知变量变已知变量。

令降维后的样本 Z 被中心化，即 $\sum_{i=1}^m z_i=0$, 中心化公式 $x_i^{'}=x_i-\frac{1}{m}\sum_{i=1}^m x_i$，显然 $\sum_{i=1}^m b_{ij}=\sum_{i=1}^m z_i^Tz_j=0$,$\sum_{j=1}^m b_{ij}=0$.

$$\sum_{i=1}^m dist_{ij}^2=\sum_{i=1}^m\|z_i\|^2+mb_{jj} \tag{2}$$

$$\sum_{j=1}^m dist_{ij}^2=\sum_{j=1}^m\|z_j\|^2+mb_{ii} \tag{3}$$

到了这一步可以将 b_{ii} 和 b_{jj} 消去了，但是 z_i 和 z_j 还没有解决：

$$\sum_{i=1}^m\sum_{j=1}^m dist_{ij}^2=m\sum_{i=1}^m\|z_i\|^2+m\sum_{j=1}^m\|z_j\|^2 \tag{4}$$

公式 2 和 3 乘以 $\frac{1}{m}$、公式 4 乘以 $\frac{1}{m^2}$，将其代入公式 1 中得：

$$b_{ij}=-\frac{1}{2}\left(dist_{ij}^2-\frac{1}{m}\sum_{i=1}^m dist_{ij}^2-\frac{1}{m}\sum_{j=1}^m dist_{ij}^2+\frac{1}{m^2}\sum_{i=1}^m\sum_{j=1}^m dist_{ij}^2\right) \tag{5}$$

厉害厉害，根据公式 5 就可以求出矩阵 B 了，之后对 B 进行特征值分解，假设特征举证中有 d^* 个非零特征值，他们构成对角举矩阵 Λ_*, 则 Z 可表达为：

$$Z=\Lambda_*^{\frac{1}{2}}V_*^T \tag{6}$$

```
import numpy as np
from sklearn import metrics,datasets,manifold
from scipy import optimize
from matplotlib import pyplot
```

```python
import pandas
import collections

def calculate_distance_matrix(x,y):
    d = metrics.pairwise_distances(x,y)
    return d

def calculate_B(D):
    (n1,n2)=D.shape
    DD=np.square(D)
    Di=np.sum(DD,axis=1)/n1
    Dj=np.sum(DD,axis=0)/n1
    Dij=np.sum(DD)/(n1**2)
    B=np.zeros((n1,n1))
    for i in range(n1):
        for j in range(n2):
            B[i,j]=(Dij+DD[i,j]-Di[i]-Dj[j])/(-2)
    return B

def MDS(data,k):
    D = calculate_distance_matrix(data,data)
    print(D)
    B=calculate_B(D)
    Be,Bv = np.linalg.eigh(B)
    Be_sort = np.argsort(Be)
    Be = Be[Be_sort]
    Bv=Bv[:,Be_sort]
    Bez = np.diag(Be[0:k])
    Bvz=Bv[:,0:k]
    Z=np.dot(np.sqrt(Bez),Bvz.T).T
    print(Z)
    if __name__ == '__main__':
        data = numpy.mat([[3,2,4],[2,0,2],[4,2,4]])
        Z=MDS(data,2)
```

Algorithm 1: MDS 算法

Input: 距离矩阵 D

Output: 矩阵 $Z = \Lambda_*^{\frac{1}{2}} V_*^T$

计算公式 2、3、4

根据公式 5 计算矩阵 B

对矩阵 B 做特征值分解

取 $d^{'}$ 个最大特征值所构成的对角矩阵

2 PCA

2.1 储备知识

主成分学习是最常用的一种降维方法：

1. 一个矩阵和该矩阵的非特征向量相乘是对该向量的旋转变换。
2. 一个矩阵和该矩阵的特征向量相乘是对该向量的伸缩变换。

2.2 数学推导

PCA 的目标是让投影后的散度最大，设投影的超平面为 V，样本 x_i 投影后的坐标为 $V^T \cdot X_i$，投影后的方差为：

$$S^2 = \frac{1}{m}\sum_{i=1}^{m}\left(V^T X_i - E\left(V^T X\right)\right)^2 \tag{7}$$

对变化后的坐标做中心化处理，使其期望值为零则：

$$\begin{aligned} S^2 &= \frac{1}{m}\sum_{i=1}^{m}\left(V^T X_i\right)^2 \\ &= \frac{1}{m}\sum_{i=1}^{m} V^T X_i X_i^T V \end{aligned} \tag{8}$$

$X_i X_i^T$ 为原始空间样本协方差，使用 C 表示，数学模型为：

$$argmax V^T C V$$
$$s.t. |V| = 1$$

采用拉格朗日乘子法进行求解：

$$f(V,\alpha) = V^T C V - \alpha\left(V V^T - 1\right) \tag{9}$$

对其求导并令其等于 0：

$$\frac{\delta f}{\delta V} = 2CV - 2\alpha V \tag{10}$$
$$CV = \alpha V \tag{11}$$

将公式 11 带入公式 9 种得：

$$f(V,\alpha) = \alpha \tag{12}$$

由公式 12 可知散度的值只由 α 来决定，α 的值越大，散度越大，也就是说我们需要找到最大的特征值与对应的特征向量。

Algorithm 2: PCA 算法

Input: 样本集 D

Output: 投影矩阵 W

1 对样本进行中心化
2 计算样本的协方差矩阵
3 对协方差矩阵进行特征值求解
4 取最大的 d 个特征值所对应的特征向量 $w_1, w_2, \ldots, w_d$

```
def PCA(data,k):
    meanVals=np.mean(data,axis=0)
    D = data-meanVals
    B = np.cov(D,rowvar=False)
    Be,Bv = np.linalg.eigh(B)
    Be_sort = np.argsort(Be)
    Bv=Bv[:,Be_sort]
    Bev = Bv[:,0:k]
    print(Bev)
if __name__ == '__main__':
    data = numpy.mat([[3,2,4],[2,0,2],[4,2,4]])
    Z=PCA(data,2)
```