# Lab4 Report

## 理论背景

### PCA

PCA是一种通过线性变换将数据转换到新坐标系统以实现降维的方法，主要目的是减少数据集中变量的数量，从而使其更易于分析和可视化，同时尽量保留原始数据的信息。

对于正交属性空间中的样本点，用一个超平面对所有样本进行恰当的表达，这样的超平面需要满足以下性质：

- 最近重构性：样本点到这个超平面的距离都足够近
- 最大可分性：样本点在这个超平面上的投影能尽可能分开

接下来用最近重构性推导目标

首先对样本中心化$\sum_{i=1}^{m} x_i = 0$

若丢弃新坐标系中的部分坐标，即将维度降低到d'<d，则样本点在低维坐标系中的投影是$z_i$，基于$z_i$来重构$x_i$，可得到

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{d'} z_{ij}\mathbf{w}_j = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{d'})\mathbf{z}_i = \mathbf{W}\mathbf{z}_i$$

$$W^\top W = I_{d'}$$

所以最小化原样本点和基于重构的样本点之间的距离

$$\min_W \left( \sum_{i=1}^{m} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2 \right) = \sum_{i=1}^{m} \mathbf{z}_i^\top \mathbf{z}_i - 2\sum_{i=1}^{m} \mathbf{z}_i^\top W^\top \mathbf{x}_i + \sum_{i=1}^{m} \mathbf{x}_i^\top \mathbf{x}_i$$

等价于

$$\max_W \sum_{i=1}^{m} \mathbf{z}_i^\top W^\top \mathbf{x}_i = \sum_{i=1}^{m} (W^\top \mathbf{x}_i)^\top W^\top \mathbf{x}_i = \sum_{i=1}^{m} \mathbf{x}_i^\top W W^\top \mathbf{x}_i = \mathrm{tr}(W^\top \sum_{i=1}^{m} (\mathbf{x}_i \mathbf{x}_i^\top) W)$$

$$= \mathrm{tr}(W^\top X X^\top W)$$

故PCA的优化目标为

$$\max_W \mathrm{tr}(W^\top X X^\top W) \quad \text{s.t.} \quad W^\top W = I_{d'}$$

使用拉格朗日乘子法可得

$$X X^\top W = \Lambda W$$

只需对协方差矩阵 $\mathbf{X}\mathbf{X}^\top$ 进行特征值分解，并将求得的特征值排序：$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$，再取前 $d'$ 个特征值对应的特征向量构成 $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{d'})$。这就是主成分分析的解。

### MDS

若要求原始空间中样本之间的距离在低维空间中得以保持，即得到“多维缩放”（MDS）

假定有$m$个样本，在原始空间中的距离矩阵为$\mathbf{D} \in \mathbb{R}^{m \times m}$，其第$i$行$j$列的元素$dist_{ij}$为样本$i$到$j$的距离。目标是获得样本在$d'$ ($\ll d$)维空间中的欧氏距离等于原始空间中的距离，即 $\|\mathbf{z}_i - \mathbf{z}_j\| = dist_{ij}$。

令$\mathbf{B} = \mathbf{Z}^\top \mathbf{Z}$，其中$\mathbf{B}$为降维后的内积矩阵，$b_{ij} = \mathbf{z}_i^\top \mathbf{z}_j$，有

$$dist_{ij}^2 = \|\mathbf{z}_i\|^2 + \|\mathbf{z}_j\|^2 - 2\mathbf{z}_i^\top \mathbf{z}_j = b_{ii} + b_{jj} - 2b_{ij}$$

经推导可得出

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

进行特征值分解,$\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$，其中$\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_d)$为特征值构成的对角矩阵， $\mathbf{V}$为特征向量矩阵

在现实应用中为了有效降维，此时可取d'≪d个最大特征值构成对角矩阵$\hat{\mathbf{\Lambda}} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_{d'})$，令$\hat{\mathbf{V}}$表示相应的特征向量矩阵，$\mathbf{Z} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}^{1/2}$

## 实验步骤

### PCA

1. 对所有样本进行中心化
2. 计算样本的协方差矩阵$\mathbf{X}\mathbf{X}^\top$
3. 对协方差矩阵做特征值分解
4. 取最大的d'个特征值所对应的特征向量
5. 输出投影矩阵
6. 利用不同数量的主成分进行数据重构，分析重构误差

### MDS

1. 计算距离矩阵$D_{ij}$
2. 计算$dist_{i\cdot}^2, dist_{\cdot j}^2, dist_{\cdot\cdot}^2$
3. 根据式子 $b_{ij} = \frac{1}{2}\left(dist_{i\cdot}^2 + dist_{\cdot j}^2 - dist_{\cdot\cdot}^2 - dist_{ij}^2\right)$ 计算矩阵$B$
4. 对B做特征值分解
5. 取最大的两个或三个特征值作为对角矩阵 $\hat{\mathbf{\Lambda}}$，$\hat{\mathbf{V}}$表示相应的特征向量矩阵,$\mathbf{Z} = \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}^{1/2}$即为样本的低维坐标

## 关键代码

### PCA

中心化数据并计算样本协方差矩阵

```
    X_mean = np.mean(X, axis=0)
    X_centered = np.subtract(X, X_mean)

    cov = np.cov(X_centered.T)
```

对矩阵特征值分解，然后设置一个重构阈值t，将排序后的特征值依次相加，直至其占总特征值的比例大于这个阈值，这些特征值的数量就是降维后的维度

```
    # 特征值和特征向量
    eigenvalues, eigenvectors = np.linalg.eigh(cov)

    idx = np.argsort(eigenvalues)[::-1]
    sorted_eigenvectors = eigenvectors[:, idx]
    sorted_eigenvalues = eigenvalues[idx]

    # calculate d'
    total_variance = sum(sorted_eigenvalues)
    variance_sum = 0
    n_components = 0
    for value in sorted_eigenvalues:
        variance_sum += value
        n_components += 1
        if variance_sum / total_variance >= variance_threshold:
            break

    components = sorted_eigenvectors[:, :n_components]
```

进行数据重构

```
    X_transformed = np.dot(X_centered, components)
    # 重构
    X_reconstructed = np.dot(X_transformed, components.T) + X_mean
```

用sklearn写一个PCA

```
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error

def sklearn_pca(X, variance_threshold):
    pca = PCA(n_components=variance_threshold)
    X_transformed = pca.fit_transform(X)

    X_reconstructed = pca.inverse_transform(X_transformed)
    return X_reconstructed
```

计算重构误差，并对比sklearn

```
variance_threshold = 0.95

X_reconstructed = pca(X, variance_threshold)
X_reconstructed_sklearn = sklearn_pca(X, variance_threshold)
reconstructed_error = np.mean(np.square(X - X_reconstructed))
print(f"manual_reconstructed_error  = {reconstructed_error}")
reconstructed_error = np.mean(np.square(X - X_reconstructed_sklearn))
print(f"sklearn_reconstructed_error = {reconstructed_error}")
```

## MDS

用lab3同样的方法，利用矩阵运算，快速计算距离矩阵

```
    m, n = X.shape
    X_square = np.square(X)
    ones = np.ones((m, n))
    D_square = X_square @ ones.T + ones @ X_square.T - 2 * X @ X.T
    distance_square = np.abs(D_square)
```

计算矩阵B时，观察到如下的矩阵centering_matrix：

$$\begin{pmatrix} 1-\frac{1}{m} & -\frac{1}{m} & \cdots & -\frac{1}{m} \\ -\frac{1}{m} & 1-\frac{1}{m} & \cdots & -\frac{1}{m} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{m} & -\frac{1}{m} & \cdots & 1-\frac{1}{m} \end{pmatrix}$$

$(centering\_matrix@distance\_square@centering\_matrix)_{ij} =$

$\sum_j(((-1/m * \sum_i dist_{ij}^2) + dist_{ij}^2) * (-1/m)) + ((-1/m * \sum_i dist_{ij}^2) + dist_{ij}^2) =$

$dist_{..}^2 - dist_{i.}^2 - dist_{.j}^2 + dist_{ij}^2$

正好就是 $-(dist_{i.}^2 + dist_{.j}^2 - dist_{..}^2 - dist_{ij}^2)$，所以这样可以快速计算得到B矩阵

```
    # Calculate matrix B
    centering_matrix = np.eye(m) - np.ones((m, m)) / m
    B = -0.5 * centering_matrix @ distance_square @ centering_matrix
```

对B特征值分解然后对特征值进行排序，选出最大的两个或者三个，最后计算样本的低维坐标并返回

```
    # 特征值和特征向量
    eigenvalues, eigenvectors = np.linalg.eigh(B)

    # Sorting
    idx = np.argsort(eigenvalues)[::-1]
    sorted_eigenvectors = eigenvectors[:, idx]
    sorted_eigenvalues = eigenvalues[idx]

    # select dimension
    eigenvectors_selected = sorted_eigenvectors[:, :dimension]
    eigenvalues_selected = sorted_eigenvalues[:dimension]

    X_reduced = eigenvectors_selected @ np.diag(np.sqrt(eigenvalues_selected))
    return X_reduced
```

# 结果分析

## PCA

**重构阈值t = 0.95**

manual_reconstructed_error  = 222.27690578580916

sklearn_reconstructed_error = 222.27690578580916

**重构阈值t = 0.9**

manual_reconstructed_error  = 446.80245727695075

sklearn_reconstructed_error = 446.802457276951

**重构阈值t = 0.98**

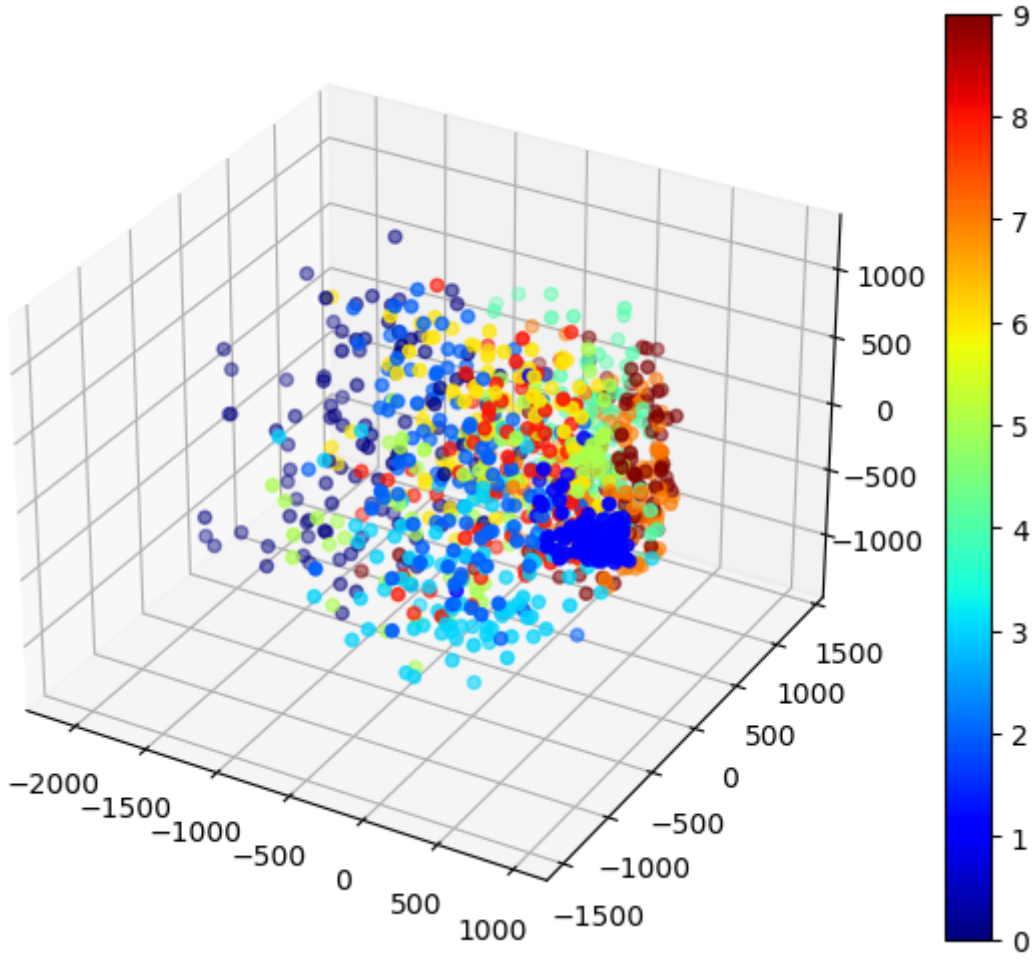
manual_reconstructed_error  = 88.8077386205223

sklearn_reconstructed_error = 88.80773862052229

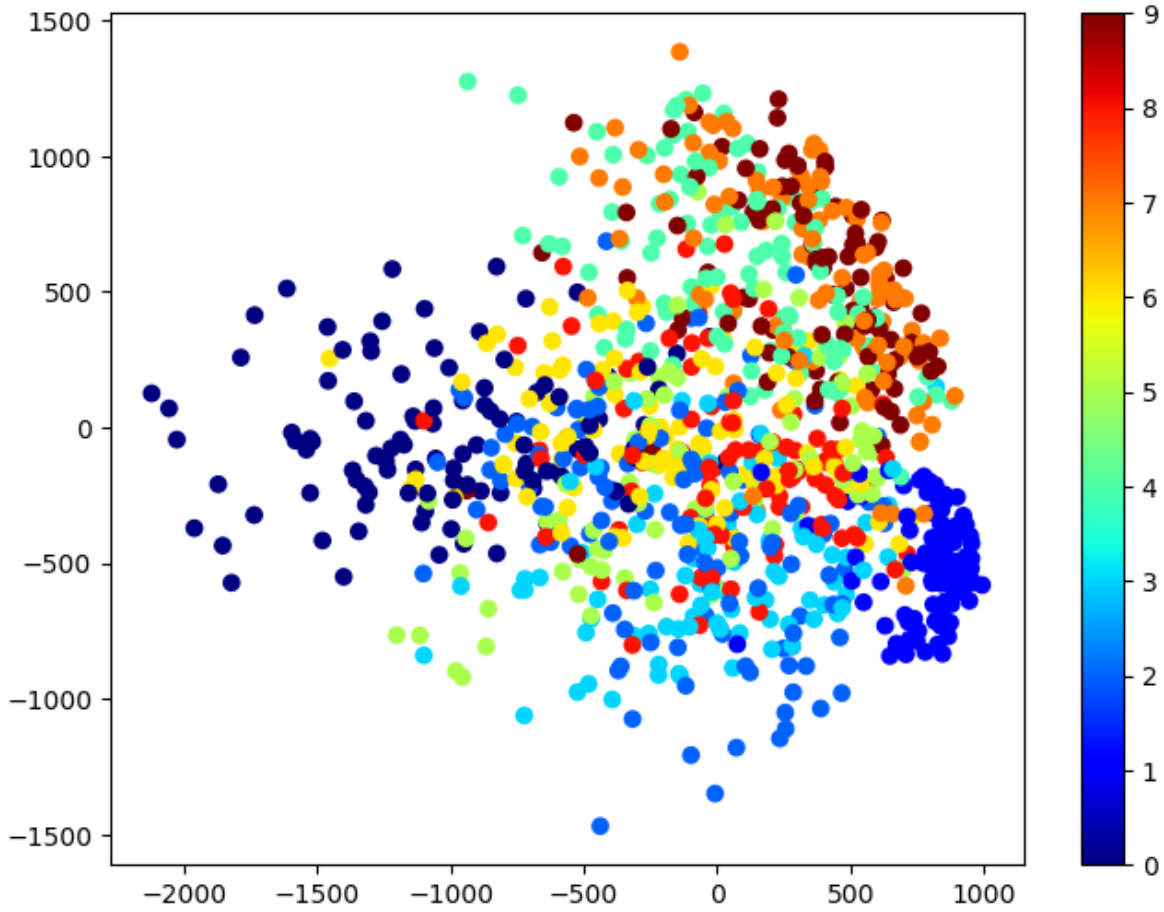
可以看出，自己写的PCA和sklearn提供的PCA结果差距非常接近，很好的实现了PCA的功能

## MDS

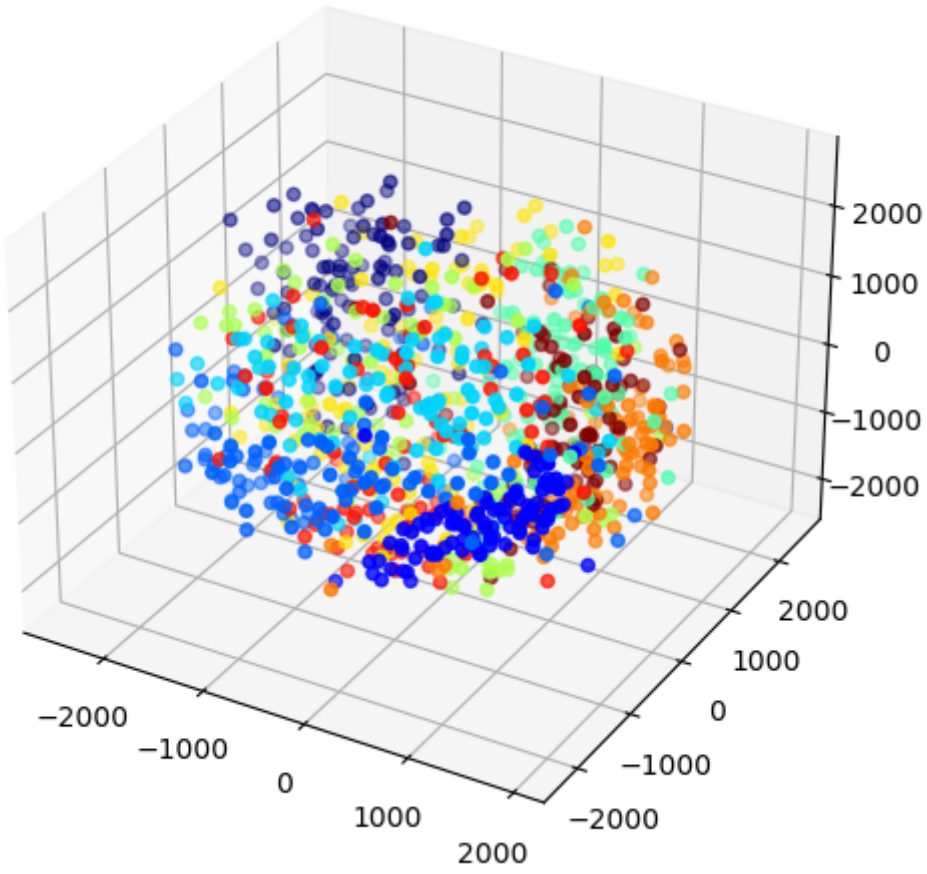
我自己实现的MDS，降维至三维的结果可视化如图

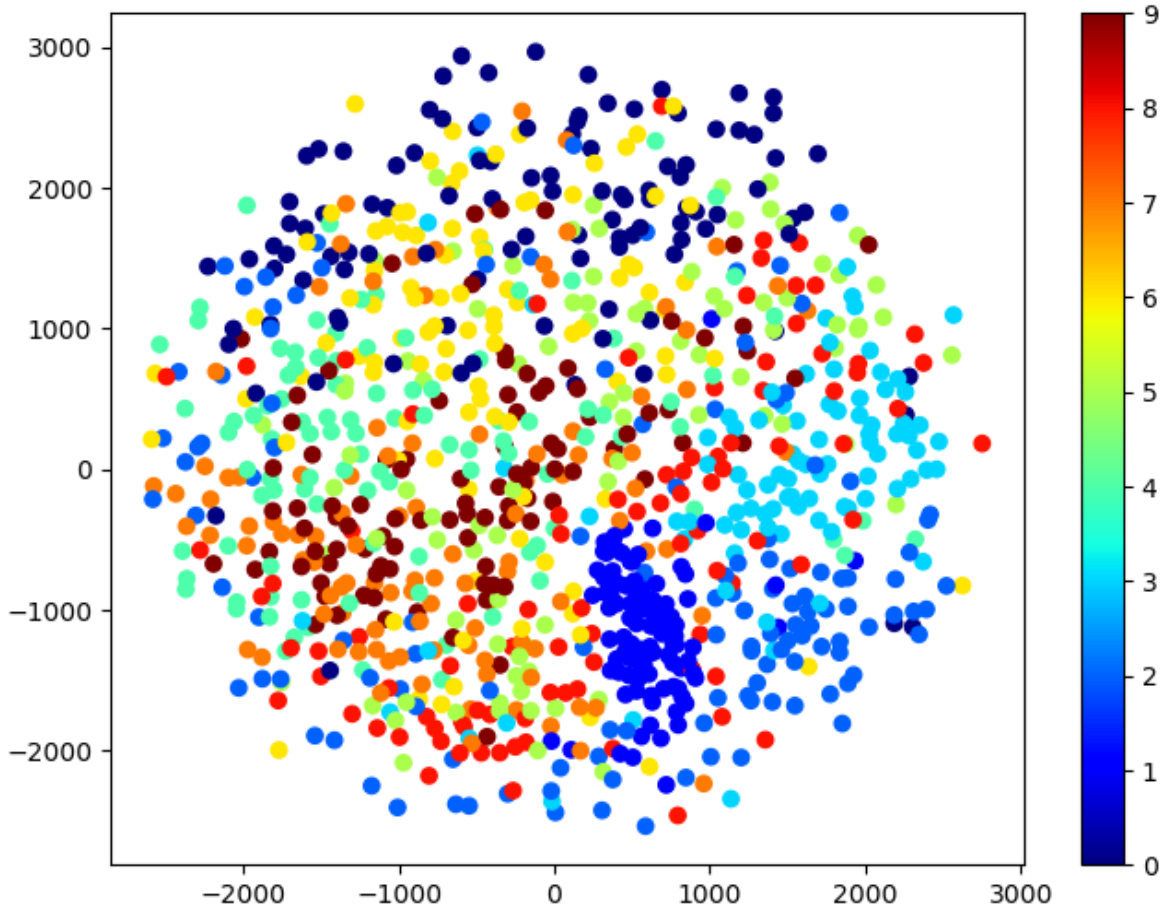



二维的结果如图


1500
1000
500
0
−500
−1000
−1500
−2000
−1500
−1000
−500
0
500
1000
9
8
7
6
5
4
3
2
1
0

**sklearn**


2000
1000
0
−1000
−2000
−2000
−1000
0
1000
2000
2000
1000
0
−1000
−2000




由于sklearn中的MDS会随机初始化，所以跟我自己的MDS比起来三维的可视化像是“旋转”了一个角度，但其实都是正确的，都保证了原始空间中样本之间的距离在低维空间中得以保持，故也很好的实现了MDS的功能

## 结论

PCA是一种通过线性变换将数据转换到新坐标系统以实现降维的方法，MDS要求原始空间中样本之间的距离在低维空间中得以保持，本次实验均成功实现了这两种降维的方法