深 圳 大 学 实 验 报 告

课 程 名 称：机器学习导论

实验项目名称：面向图像压缩和人脸识别的自动聚类和降维分析

学　　院：计算机与软件学院

专　　业：计算机科学与技术

指 导 教 师：贾森、徐萌

报告人：刘睿辰　学号：**2018152051**　班级：数计班

实 验 时 间：**2021 年 6 月 7 日**

实验报告提交时间：**2021.6.14**

教务处制

一、 实验目标：

1. 理解聚类和降维的基本过程；
2. 掌握聚类和降维方法中的关键参数选择和优化方法；
3. 利用图像压缩和人脸识别等应用，掌握聚类算法和降维算法，实现图像的有效压缩，理解人脸识别中降维的基本过程；
4. 利用已有的全世界多个国家的 GDP 和人口数据，掌握聚类算法和降维算法，实现发达国家/发展中国家的有效划分。

二、实验环境：

1. 操作系统：Windows 10；
2. 实验平台：MATLAB R2019a。

三、实验内容、步骤与结果

1. 根据 ex3.rar 文件中的英文 pdf 文档，完成文档中的实验内容；

1.1 理论背景

无监督学习(unsupervised learning)：现实生活中常常会有这样的问题：缺乏足够的先验知识，因此难以人工标注类别或进行人工类别标注的成本太高。很自然地，我们希望计算机能代我们完成这些工作，或至少提供一些帮助。根据类别未知(没有被标记)的训练样本解决模式识别中的各种问题，称之为无监督学习。

根据无监督学习的定义，无监督学习并没有人为地给出标签，需要我们进行分类。常见的无监督学习算法有聚类(Clustering)、主成分分析(Principal Component Analysis, PCA)等等。在本次实验中，我们主要针对这两种常见的无监督学习算法进行阐述。

事实上，在之前我们实现的机器学习算法中，线性回归(Linear Regression)、逻辑回归(Logistic Regression)以及 BP 神经网络都属于监督学习(supervised learning)。因为我们得到的数据集都是有标签的，我们是根据数据集的标签来进行分类的。这就和无监督学习产生了区别，无监督学习并没有所谓标签，我们仅仅凭借样本点“人为”地进行划分。例如我们给出例子，如图 1 所示。

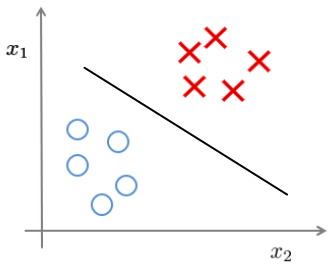


图 1(a). 逻辑回归

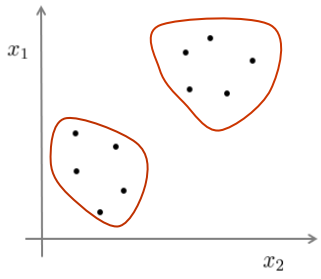


图 1(b). 聚类分析

在图 1(a)中我们模拟了逻辑回归，可以看到不同的样本标记不同，我们需要将这两类样本分开。而在图 1(b)中，所有样本都是一样的，并没有做标记，这时我们的任务就是画出图 1(b)中的圈来对样本进行人为的划分。这就是监督学习与无监督学习的区别所在。下面阐述的聚类(k均值算法)、PCA 都属于无监督学习。

1.2 $k-\text{means}$ 算法

k均值算法实际上属于一种原型聚类(prototype-based clustering)，即假设聚类结构能通过一组原型刻画。该算法的基本思想为，给定样本集$\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},...,\boldsymbol{x}^{(m)}\}$，针对聚类所得的簇划分$D=\{C_1,C_2,...,C_k\}$最小化平方误差

$$E=\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i}\|\boldsymbol{x}-\boldsymbol{\mu_i}\|^2$$

其中我们有

$$\boldsymbol{\mu}_i=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$$

被定义为簇C_i的均值向量。

所以该算法的输入就是所有的样本点$\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},...,\boldsymbol{x}^{(m)}\}$以及分类的个数$\boldsymbol{K}$，也就是有多少个簇。然后执行下面操作，如图 2 所示。

Randomly initialize K cluster centroids $\mu_1,\mu_2,\ldots,\mu_K\in\mathbb{R}^n$
Repeat {
for i = 1 to m
$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$
for k = 1 to K
μ_k := average (mean) of points assigned to cluster k
}

图 2. k均值算法流程

我们来对算法进行描述。

1） 定义$\boldsymbol{K}$个中心点，这些点采用随机初始化的方法；

2） 对于每一个样本点，计算哪一个中心点离该样本点最近，然后标记$\boldsymbol{x}^{(i)}$为$c^{(i)}$，$c^{(i)}\in\{1,2,...,K\}$；

3） 等待一轮结束之后，计算每一个类（类中标记都相等）的中心点，然后更新每一个类的中心点。

以上为一次迭代的过程。如果进行多次迭代，最终得到了收敛的结果，也就是中心点不再发生变化，就代表此时此刻我们找到了最终的聚类结果。

现在，我们尝试在 MATLAB 上编写代码来实现k均值算法。假设我们得到了一组数据集$\boldsymbol{X}$，行数代表样本个数，列数代表样本维度。该数据集$\boldsymbol{X}$是为了方便可视化的，所以维度只有 2，代表二维平面上面的一些点。然后我们随机初始化三个中心点坐标，然后我们需要对样本点进行遍历，对每个样本点标记哪一个中心点距离它最近，使用idx数组来标记每一个点的“最近中心”。这就是findClosestCentroids.m的内容。代码如图 3 所示。

```
% Set K
K = size(centroids, 1);
for i = 1:size(X,1)
    dist = realmax;
    for j = 1:K
        tag = norm(X(i,:)-centroids(j,:));  %Euclidean norm
        if dist > tag
            key = j;
            dist = tag;
        end
    end
    idx(i) = key;
end
```

图 3. k 均值算法 — 寻找最近中心点

简单来说，第三行我们使用的是欧几里得范数来计算点与点之间的距离。欧几里得范数实际上就是计算点与点之间的几何距离，即对于m维的向量$\boldsymbol{x}$和$\boldsymbol{y}$，几何距离为

$$\|\boldsymbol{x}-\boldsymbol{y}\|=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_m-y_m)^2}$$

然后对每一个点都这样计算即可，得到每一个点的最近中心点编号。在最后，idx 数组就是一个长度为样本个数且元素取值都是属于集合$\{1,2,\ldots,K\}$的。

这部分完成之后，我们再来更新每一个类的中心点坐标。这部分我们的实现思想如下所示

$$\boldsymbol{\mu}_i=\frac{1}{|C_i|}\sum_{\boldsymbol{x}\in C_i}\boldsymbol{x}$$

即计算每一个类的样本坐标之和，然后除以该类中样本个数得到样本的平均值，这样得到每一个类的中心点，然后进行更新。代码如图 4 所示，这就是computeCentroids.m 的内容。

```
counts = zeros(1,K);  % number of cluster

for i = 1:m
    % update the sample size of this cluster
    counts(idx(i)) = counts(idx(i)) + 1;
    % add this sample point
    centroids(idx(i),:) = centroids(idx(i),:) + X(i,:);
end

for i = 1:K
    % get the mean of each cluster
    centroids(i,:) = centroids(i,:) / counts(i);
end
```

图 4. k 均值算法 — 更新每一个簇的中心点

以上部分为一次迭代的过程。事实上经过一次迭代之后我们可能无法找到最优解。当然我们可以通过多次随机初始化找到很多簇划分找到最好的，但是由于随机初始化种类过多，所以我们不可能找到所有的簇划分，事实上这也是一个 NP 难问题[Aloise et al., 2009]。所以实际上k均值算法也是一种贪心算法，通过迭代优化来逼近最优解。所以我们进行多次迭代，如果最终中心点不动达到收敛态，我们就找到了当前的聚类结果。

我们随机初始化三个中心点如图 5(a)所示。经过第一次迭代之后找到一个新的中心点，如图 5(b)所示。我们将先后这两个点用直线连接起来，可以看到中心点位置变化的趋势。前 9 次迭代的结果如图 5 所示。

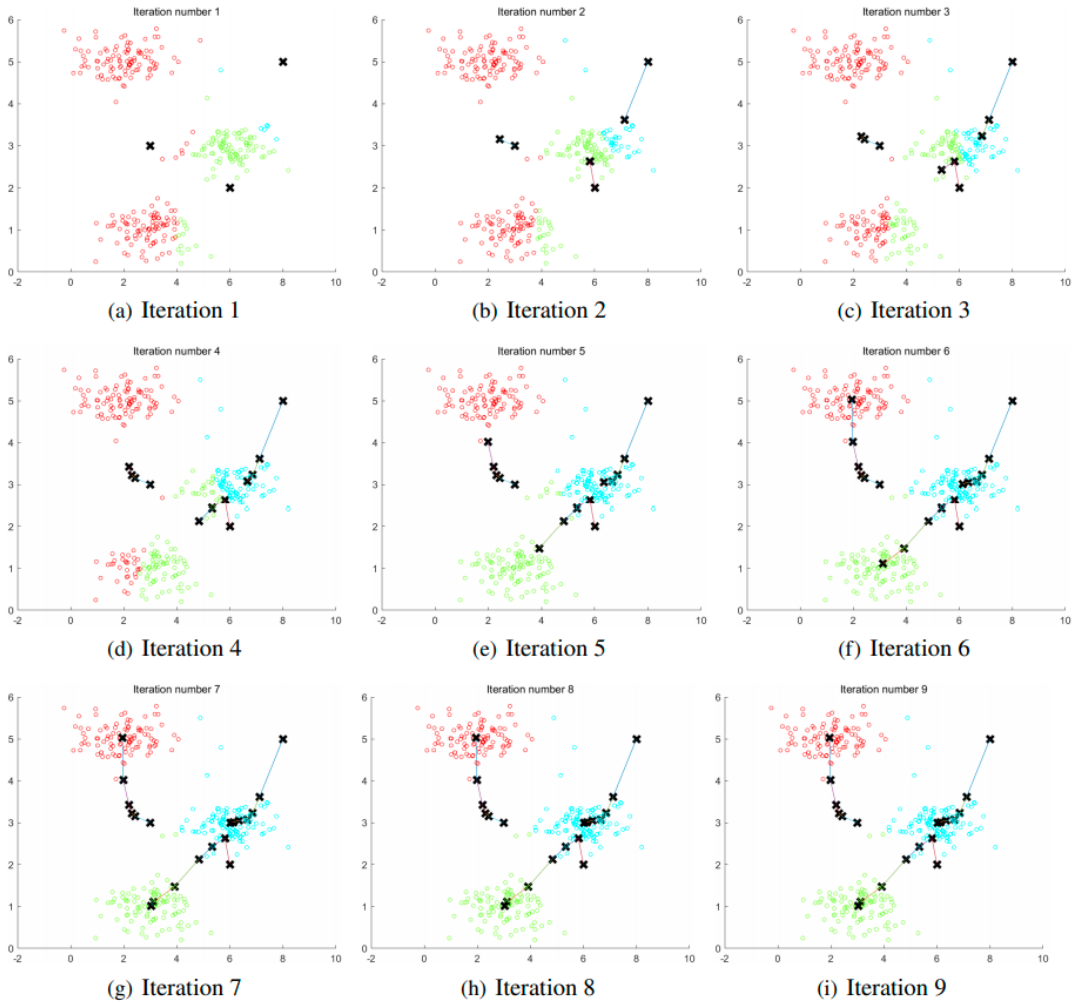


图 5. k 均值算法 — 前 9 次聚类结果

图 5 的每一张图中，距离同一个中心点最近的点都被染上了同样的颜色。在最开始的时候，看起来有两个类已经被划分为一个簇了；但是在最后我们可以看到这两个类被分为两个簇了。而且中心点的变化趋势就是向类中心移动的。

另外，从第 7 次迭代之后，点的变化在图中就很难看到了，所以 7 次迭代之后就已经趋近于收敛的状态。最终在第 10 次我们得到的图如图 6 所示。

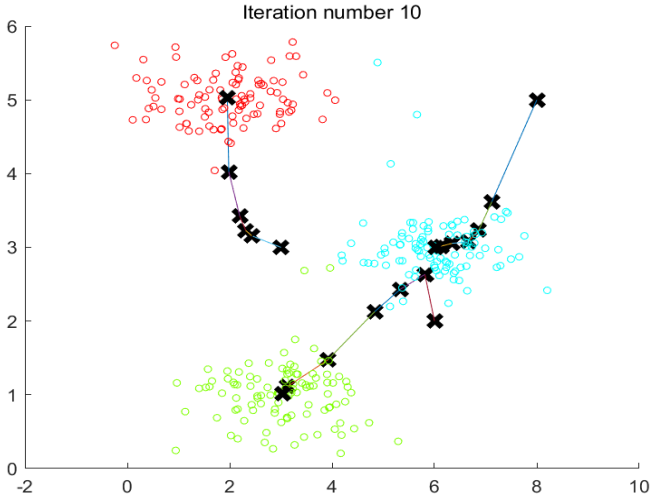


图 6. k 均值算法 — 聚类结果

实际应用：图像压缩处理

刚才我们使用较为直观的二维平面上的聚类来阐述了聚类的过程，在实际应用中，聚类的作用非常多，这里引入一个图像压缩处理的例子。

如图 7 所示是一张128×128的图像。这是一个 24 位 RGB 图象，每一个像素由三个 8 位的无符号整数来表示。例如，白色的情况下 R=255, G=255, B=255；黑色的情况下三者就都为 0。

图 7. k均值算法应用 — 待压缩图片

如果从图片存储的角度出发，这张图片需要$128\times128\times24=393216\text{bits}=48\text{KB}$来进行存储。但是，如果将其压缩为 16 色图片，也就是整张图片仅仅包括 16 个颜色，每一个颜色都有24bits的存储要求（把它们作为索引列表以便进行解压），那么每一个像素仅需要 4 位来标记其颜色（$2^4=16$）。所以，整张图片的大小就由$16\times24+128\times128\times4=65920\text{bits}\approx8\text{KB}$来存储，大大缩减了存储要求。

了解了图片压缩的原理之后，我们尝试用聚类的思想来实现上述功能。经过分析之后，我们可以认为，每一个像素点都是一个样本，该样本可以认为是 RGB 三维空间的样本，正如几何平面上的点是二维样本一样。然后我们设置聚类个数为 16，这样每一个像素都可以被距离它最近的中心点的颜色值所表示。这样一来，整张图片就只有 16 个颜色，实现了我们上述的目的。

这部分我们需要调用之前写过的函数。唯一不同的是我们需要编写随机初始化的函数kMeansInitCentroids.m。由于是随机初始化，一种随机策略是随机地抽取K个样本集中的样本作为初始中心点。但是由于样本集矩阵不能被随意破坏，所以我们使用MATLAB 的randperm 函数，它可以将一个序列中的数打乱顺序。然后取前K个作为初始中心点序号，这样同样实现随机初始化，代码如图 8 所示。

```
randidx = randperm(size(X,1));
for i = 1:K
    res = zeros(1,size(X,2));
    for j = 1:size(X,2)
        res(j) = X(randidx(i),j);
    end
    centroids(i,:) = res;
end
```

图 8. k均值算法应用 — 随机初始化

随机初始化之后，设置$K=16$，同样地先设置迭代次数为 10。这里面要注意，由于MATLAB 的imread 函数读取图片我们得到的是一个$128\times128\times3$的三维矩阵，所以我们要通过reshape 函数将其转化为$(128\times128)\times3$的二维数组便于处理。在样本数组被中心点数组所代替之后，还要通过reshape 函数将其转换回三维的形式，这样才能正确地

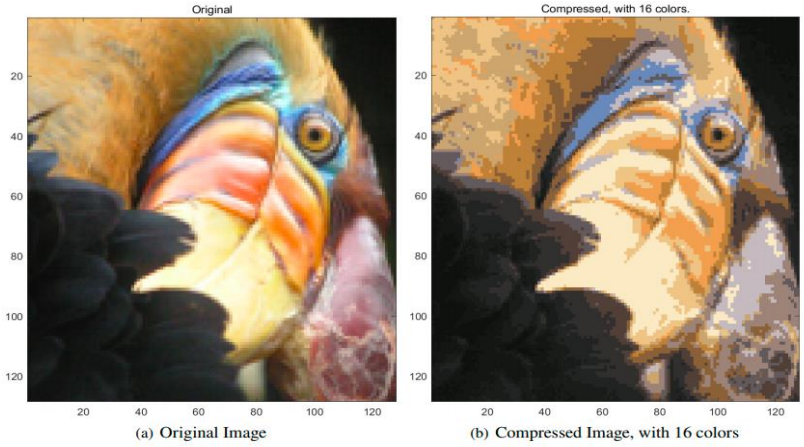
显示出图片。

代码如图 9 所示。

```
% Read the image
A = double(imread('bird_small.png'));

% Divide by 255 so that all values are in the range 0 - 1
A = A / 255;

% Size of the image
img_size = size(A);

% Reshape the image into an Nx3 matrix where N = number of pixels.
% Each row will contain the Red, Green and Blue pixel values
% This gives us our dataset matrix X that we will use K-Means on.
X = reshape(A, img_size(1) * img_size(2), 3);

K = 16;
max_iters = 10;

% Initialize randomly
initial_centroids = kMeansInitCentroids(X, K);

% Run K-Means
[centroids, idx] = runkMeans(X, initial_centroids, max_iters);

% Find closest cluster members
idx = findClosestCentroids(X, centroids);

% Replace the sample matrix by their centroids
X_recovered = centroids(idx,:);

% Reshape the recovered image into proper dimensions
X_recovered = reshape(X_recovered, img_size(1), img_size(2), 3);

% Display the original image
subplot(1, 2, 1);
imagesc(A);
title('Original');

% Display compressed image side by side
subplot(1, 2, 2);
imagesc(X_recovered)
title(sprintf('Compressed,_with_%d_colors.', K));
```

图 9. k 均值算法应用 — 聚类过程并显示压缩图片

运行代码，结果如图 10 所示。



(a) Original Image (b) Compressed Image, with 16 colors

图 10. k 均值算法应用 — 通过聚类压缩图片结果

对比图 10(a)和图 10(b)我们可以发现，图 10(b)像素上的差别更加明显，这就体现压缩

后的图片更加“单调”，因为它只有 16 个颜色。我们将压缩得到的矩阵进行去重，得到所有的像素的颜色 RGB 分量，如下所示。

Color 1: R:17; G:19; B:16;
Color 2: R:26; G:28; B:26;
Color 3: R:39; G:41; B:38;
Color 4: R:60; G:57; B:56;
Color 5: R:89; G:86; B:93;
Color 6: R:102; G:72; B:50;
Color 7: R:105; G:124; B:167;
Color 8: R:133; G:100; B:58;
Color 9: R:145; G:121; B:107;
Color 10: R:176; G:157; B:142;
Color 11: R:178; G:128; B:55;
Color 12: R:186; G:188; B:202;
Color 13: R:199; G:162; B:95;
Color 14: R:227; G:192; B:128;
Color 15: R:237; G:152; B:72;
Color 16: R:247; G:233; B:194;

这不仅证明了整张图片仅有这 16 种颜色，还分别将 16 个颜色的 RGB 值列举出来。

1.3 主成分分析(Principal Component Analysis, PCA)

主成分分析是一种常见的数据分析方式，常用于高维数据的降维，可用于提取数据的主要特征分量。我们试想在高维空间中，样本采集有时候是非常困难的。此外，在高维空间中，甚至计算内积都是一件很困难的事情。所以，所有的机器学习方法都面临着高维空间的“维数灾难”(curse of dimensionality)。

解决维数灾难最直观的想法就是进行数据的降维(dimension reduction)。PCA 是最常见的一种降维方法。我们先来思考一下对于二维空间$\mathbb{R}^2$中的样本，如何用一条直线，也就是一维空间来恰当地表达？

一般来说，这样的直线应该具有以下的性质：

1） 最近重构性：样本点到这条直线的距离都足够近；

2） 最大可分性：样本点在这个直线上的投影尽可能分开。

如图 11 所示，假设这是两种投影方法，构造出一维平面l_1和l_2。

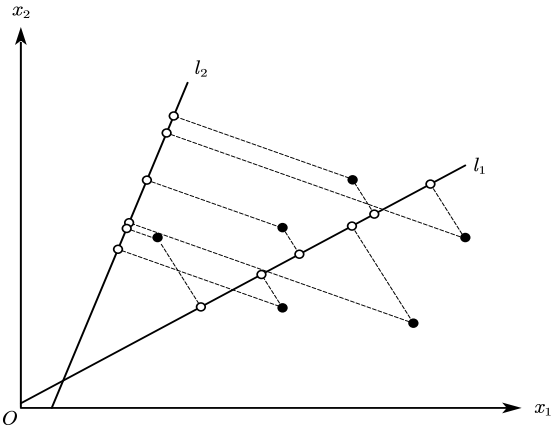


图 11. PCA 应使得所有样本投影尽可能分开

在一维平面l_1上，样本点投影相对比较分散；

而在一维平面l_2上，样本点投影相对比较集中。根据最大可分性，我们在二者中选择一维平面l_1作为我们的降维结果。根据概率论的观点，样本中的数据越分散，该数据集的方差就越大。所以我们找到了一个学习目标，即最大化各分量方差。同时，由于各分量之间不存在相关性，所以各分量之间得协方差自然也应为 0，即

$$Cov(\boldsymbol{x}_1,\boldsymbol{x}_2)=\frac{\sum_{i=1}^{m}\left(x_1^{(i)}-\overline{\boldsymbol{x}_1}\right)\left(x_2^{(i)}-\overline{\boldsymbol{x}_2}\right)}{m-1}=0$$

所以最终我们的目标就是将一组N维向量降为K维（$0<K<N$），其目标是选择K个单位（模为 1）正交基，使得原始数据变换到这组基上后，各字段两两间协方差为 0，而字段的方差则尽可能大（在正交的约束下，取最大的K个方差）。
现在优化目标被确认，但是他并不是一个量化的指标。事实上，对于N维空间，样本矩阵表示成

$$\boldsymbol{X}=\begin{pmatrix}x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)}\\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(m)}\\ \vdots & \vdots & \ddots & \vdots\\ x_N^{(1)} & x_N^{(2)} & \cdots & x_N^{(m)}\end{pmatrix}$$

如果我们首先将其标准化，那么我们有

$$\frac{1}{m-1}\boldsymbol{X}\boldsymbol{X}^T=\begin{pmatrix}s_{\boldsymbol{x}_1}^2 & Cov(\boldsymbol{x}_1,\boldsymbol{x}_2) & \cdots & Cov(\boldsymbol{x}_1,\boldsymbol{x}_N)\\ Cov(\boldsymbol{x}_2,\boldsymbol{x}_1) & s_{\boldsymbol{x}_2}^2 & \cdots & Cov(\boldsymbol{x}_2,\boldsymbol{x}_N)\\ \vdots & \vdots & \ddots & \vdots\\ Cov(\boldsymbol{x}_N,\boldsymbol{x}_1) & Cov(\boldsymbol{x}_N,\boldsymbol{x}_2) & \cdots & s_{\boldsymbol{x}_N}^2\end{pmatrix}$$

由于常数项不影响优化目标，所以我们的目标矩阵就是

$$\boldsymbol{\Sigma}=\frac{1}{m}\boldsymbol{X}\boldsymbol{X}^T$$

该目标矩阵我们称之为协方差矩阵，它是 PCA 一个非常重要的优化目标。根据我们之前的描述，我们要做的就是让其非对角线元素为 0，也可以称之为对角化。
那么设原始样本集$\boldsymbol{X}$经过线性变换之后得到矩阵$\boldsymbol{Y}$，设$\boldsymbol{P}$为一组基按行组成的矩阵，即

$$\boldsymbol{P}=(\boldsymbol{P}_1,\boldsymbol{P}_2,...,\boldsymbol{P}_N)^T$$

其中$\boldsymbol{P}_1,\boldsymbol{P}_2,...,\boldsymbol{P}_N$为行向量，那么我们有

$$\boldsymbol{Y}=\boldsymbol{P}\boldsymbol{X}$$

那么，我们有

$$\boldsymbol{D}=\frac{1}{m}\boldsymbol{Y}\boldsymbol{Y}^T=\frac{1}{m}(\boldsymbol{P}\boldsymbol{X})(\boldsymbol{P}\boldsymbol{X})^T=\frac{1}{m}\boldsymbol{P}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{P}^T=\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^T$$

由于规范正交矩阵有

$$\boldsymbol{P}^{-1}=\boldsymbol{P}^T$$

所以上式可化为

$$\boldsymbol{\Sigma}=\boldsymbol{P}^T\boldsymbol{D}\boldsymbol{P}$$

所以$\boldsymbol{X}$的协方差矩阵和经过线性变换之后得到的协方差矩阵之间的关系就得到了。由于

我们想要将$\boldsymbol{\Sigma}=\frac{1}{m}\boldsymbol{X}\boldsymbol{X}^{T}$进行对角化，而矩阵特征分解正是基于对角化来实现的。而我们要寻找的矩阵就是这个$\boldsymbol{P}$。对$\boldsymbol{\Sigma}$经过特征分解之后，我们有

$$\boldsymbol{\Sigma}=\boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{T}$$

所以

$$\boldsymbol{P}=\boldsymbol{U}^{T}$$

接下来，我们将$\boldsymbol{P}$中的特征向量按照它们特征值大小重新进行排列，然后取前K行乘以原始样本集，就可以得到降维的结果。

实际应用中，我们一般使用奇异值分解来代替特征分解。奇异值分解的主要形式为

$$\boldsymbol{\Sigma}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{T}$$

其中$\boldsymbol{U}$和$\boldsymbol{V}$都是酉矩阵。奇异值分解的好处在于对$\boldsymbol{\Sigma}$并不要求为方阵。

设$\boldsymbol{\Sigma}$为$m\times n$的矩阵，那么$\boldsymbol{U}$为$m\times m$的矩阵，$\boldsymbol{V}$为$n\times n$的矩阵。由于$\boldsymbol{U}$和$\boldsymbol{V}$都是酉矩阵，所以有

$$\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}=\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}=\boldsymbol{E}$$

$$\boldsymbol{V}\boldsymbol{V}^{\boldsymbol{T}}=\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{V}=\boldsymbol{E}$$

由于$(\lambda\boldsymbol{E}-\boldsymbol{A})^{T}=\lambda\boldsymbol{E}-\boldsymbol{A}^{T}=\boldsymbol{0}$，所以$\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\boldsymbol{T}}$和$\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{\Sigma}$的特征值相同，但并不代表它们的特征向量也相同。

由于$\boldsymbol{\Sigma}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{T}$，则$\boldsymbol{\Sigma}^{T}=\boldsymbol{V}\boldsymbol{S}^{T}\boldsymbol{U}^{T}$,则有

$$\boldsymbol{\Sigma}^{T}\boldsymbol{\Sigma}=\boldsymbol{V}\boldsymbol{S}^{T}\boldsymbol{U}^{T}\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{T}=\boldsymbol{V}\boldsymbol{S}^{T}\boldsymbol{S}\boldsymbol{V}^{T}$$

同理，

$$\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{T}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^{T}\boldsymbol{V}\boldsymbol{S}^{T}\boldsymbol{U}^{T}=\boldsymbol{U}\boldsymbol{S}^{T}\boldsymbol{S}\boldsymbol{U}^{T}$$

所以$\boldsymbol{V}$和$\boldsymbol{U}$事实上分别是$\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{\Sigma}$和$\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\boldsymbol{T}}$的特征向量矩阵。所以我们有

$$\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{T}\right)\boldsymbol{v}_{i}=\lambda_{i}\boldsymbol{v}_{i}$$

$$\left(\boldsymbol{\Sigma}^{T}\boldsymbol{\Sigma}\right)\boldsymbol{u}_{i}=\lambda_{i}\boldsymbol{u}_{i}$$

所以

$$\boldsymbol{\Sigma}\boldsymbol{V}=\boldsymbol{U}\boldsymbol{S}$$

故

$$\boldsymbol{\Sigma}\boldsymbol{v}_{i}=s_{i}\boldsymbol{u}_{i}$$

这样我们就求得了$\boldsymbol{S}$的对角元素，也就是奇异值。

在 MATLAB 中，函数svd可用于进行奇异值分解，且得到的矩阵U是**按照奇异值大小来进行排序**的。然后取前K个乘以原始样本集，就可以得到降维的结果。

降维的结果我们用$\boldsymbol{z}$来表示。我们在奇异值分解部分得到了U矩阵，所以截取前K个列乘以原始样本集得到降维结果，即

$$\boldsymbol{z}=\boldsymbol{P}_{reduced}\boldsymbol{x}=\boldsymbol{U}_{reduced}^{T}\boldsymbol{x}$$

这里面$\boldsymbol{U}_{reduced}$是N(原维度)$\times K$(降到的维数)的矩阵，需要转置之后乘以原始样本集，得到K(降到的维数)$\times m$(样本容量)的矩阵。这就是我们降维的目标。

如果想要对降维之后的样本点进行可视化或者得到它们的坐标，我们可以用$\boldsymbol{U}_{reduced}$矩阵来乘以$\boldsymbol{z}$来近似。举例来说，如果二维降低到一维，$\boldsymbol{z}$就代表样本点距离中心的距离，是一个常数。那么如果想要得到这个点的几何坐标，需要用长度乘以方向，这个方向就是$\boldsymbol{U}_{reduced}$。这点不难理解，如果是二维降一维的情况，$\boldsymbol{U}_{reduced}$就是一个2×1的矩阵，代表一个二维平面的向量。用这个向量乘以各样本点降维之后对应的常数，也等于一个向量，代表这个投影点的坐标，即

$$\boldsymbol{x}_{approx} = \boldsymbol{U}_{reduced}\boldsymbol{z}$$

$\boldsymbol{U}_{reduced}$矩阵非常重要，它代表着我们降维得到的直线的方向，称之为“主成分”。在高维情况下，该矩阵代表着“超平面”的方向。

接下来我们来对二维平面上的降维进行可视化。首先我们得到一组数据集，先进行标准化。标准化的方法和之前类似，我们只需用

$$\boldsymbol{x}^* = \frac{\boldsymbol{x} - \bar{x}}{\sigma}$$

来代替原来的数据集即可。代码如图 12 所示。

```
mu = mean(X);
X_norm = bsxfun(@minus, X, mu);

sigma = std(X_norm);
X_norm = bsxfun(@rdivide, X_norm, sigma);
```

图 12. PCA — 特征标准化

然后求得被预处理后的数据集的协方差矩阵，并进行奇异值分解。这里面由于$\boldsymbol{X}$是m(样本容量)$\times N$(原维度)的矩阵，需要进行转置，然后再计算协方差矩阵。奇异值分解直接调用 MATLAB 函数svd，其函数原型为$[\boldsymbol{U},\boldsymbol{S},\boldsymbol{X}]=\text{svd}(\boldsymbol{\Sigma})$，保证$\boldsymbol{\Sigma}=\boldsymbol{USX}^T$。代码如图 13 所示。

```
%covariation matrix
Sigma = (1/m)*(X'*X);

%Singular value decomposition
[U, S, X] = svd(Sigma);
```

图 13. PCA — 协方差矩阵与奇异值分解

接下来，我们将得到的U矩阵中的向量选择两个进行绘制，如图 14 所示。可以发现，这两个向量彼此正交，这也表示实对称矩阵不同的特征值对应的特征向量是相互正交的。事实上，协方差矩阵是一个是对称矩阵，因为

$$\left(\boldsymbol{XX}^T\right)^T = \left(\boldsymbol{X}^T\right)^T\boldsymbol{X}^T = \boldsymbol{XX}^T$$

所以其特征向量两两正交，正如图 14 所示。

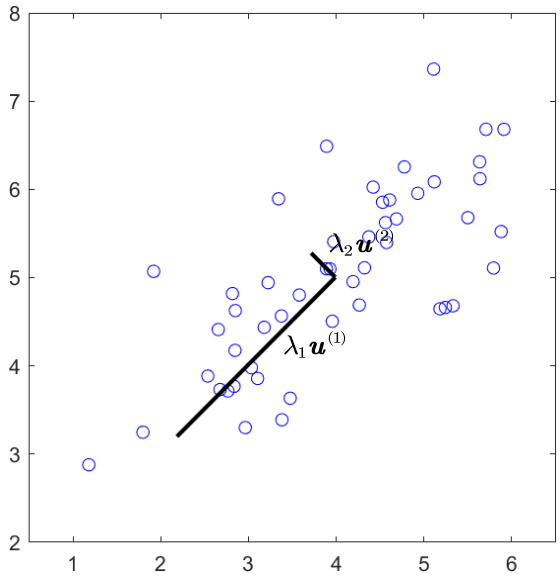


图 14. PCA — 实对称矩阵不同的特征值对应的特征向量是相互正交的

仔细观察图 14，我们选取的是U矩阵的前两个特征向量，也就是U矩阵的全部特征向量。在降维的过程中，我们选取的是$\boldsymbol{u}^{(1)}$，因为这个特征长度更长，也就是特征值更大，属于“主成分”。假如我们仅仅输出特征向量，由于两个特征向量都是经过标准化的，所以长度应该一样长，如图 15 所示。

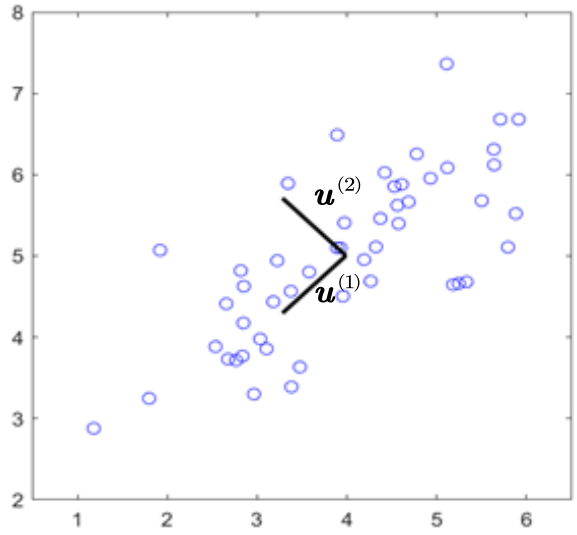


图 15. PCA — 特征值大小影响“主成分”的选择，特征向量的长度都是 1

对比两张图，我们可以发现图 14 中的$\boldsymbol{u}^{(1)}$是主成分方向，所以我们降维的超平面也是图 14 中的$\boldsymbol{u}^{(1)}$方向。

接下来我们进行降维，也就是选取U矩阵的第一列，乘以原数据集矩阵，得到降维后的$\boldsymbol{z}$矩阵。这里面$\boldsymbol{z}$矩阵里面都是一维向量，也就是常数，代表该点在一维平面上距离中心点的距离。代码如图 16 所示。得到的$\boldsymbol{z}$矩阵的行数为 1，代表降维之后的样本都已经变成了一维的了。

```
U_reduce = U(:,1:K);  %dimension reduction, 2*1

Z = U_reduce' * X';  %(1*2) * (2*50) = 1*50

Z = Z';  %50*1
```

图 16. PCA — 降维

如果想要将降维后的投影可视化出来，我们需要将$\boldsymbol{U}_{reduced}$乘以$\boldsymbol{z}$，得到投影点的坐标。代码如图 17 所示。

```
U_reduce = U(:,1:K);  % 2*1
U_reduce = U_reduce'; % 1*2
X_rec = Z * U_reduce; % (50*1)*(1*2)=(50*2)
```

图 17. PCA — 投影点可视化

我们将$\boldsymbol{u}^{(1)}$方向进行可视化，并将样本点和投影点可视化，可以得到如图 18 所示的结果。这个结果和我们之前的分析完全一致，样本点及其在$\boldsymbol{u}^{(1)}$方向的投影都被可视化了。

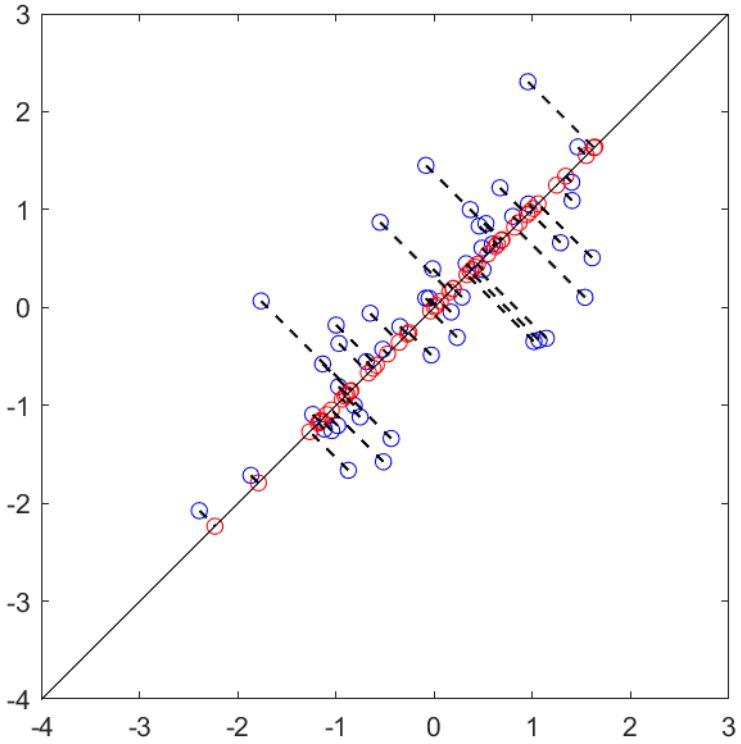


图 18. PCA — 样本点及其在超平面上的投影点

实际应用：人脸识别(Face Recognition)

PCA 的一个实际应用就是基于人脸的识别。人脸是有很多特征的，人脸被识别出来之后保存在矩阵中，可以看作是一个维度比较高的向量，如果后续有识别人脸的需求，就需要将其降维处理。

首先我们得到一组数据集，数据集容量为 5000，每一张人脸是一个32×32的灰度图，所以每一张人脸矩阵都被展开形成一个1×1024的行向量。可视化之后如图 19 所示。

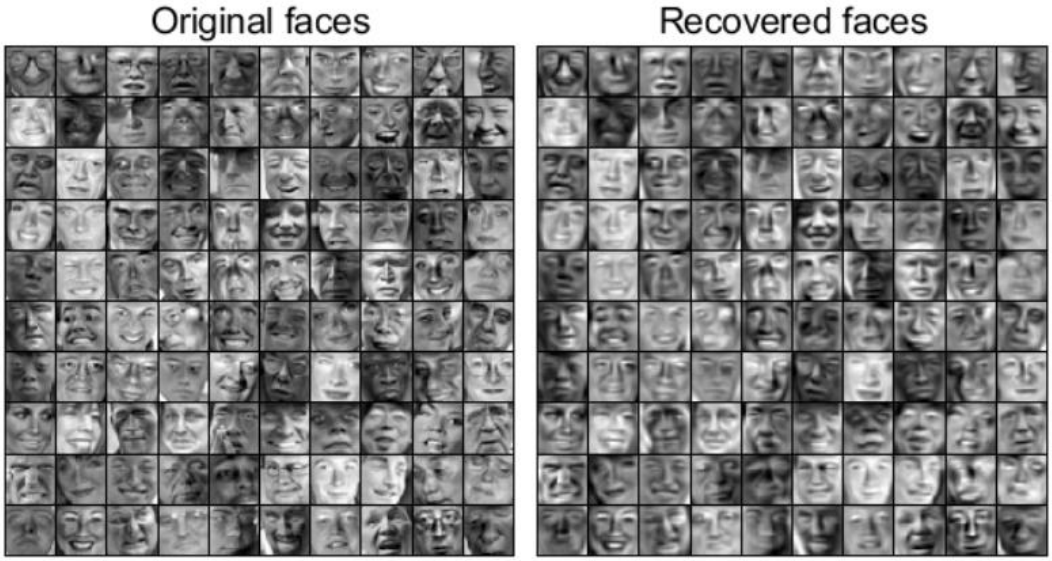
图 19. PCA — 人脸识别样本集

读取信息之后可以得到$\boldsymbol{X}$是一个5000×1024的矩阵。根据 PCA 的流程，我们首先求得它的协方差矩阵，然后进行奇异值分解得到主成分矩阵。经过奇异值分解之后得到的主成分矩阵为1024×1024的。在二维情况下我们将特征向量矩阵可视化之后发现是两个垂直的向量。那么在 1024 维的空间，特征向量必然也是两两正交的。我们从 1024 个特征向量中选取 36 个进行可视化，如图 20 所示。事实上，这 1024 个特征向量彼此正交，构成了 1024 维空间的基。

图 20. PCA — 人脸 1024 维空间的基

在这里，特征向量可视化之后并不是那么直观，但是我们可以将它们类比为二维平面上

两个彼此正交的基向量。
接下来我们考虑降维。例如，我们降维至 100 维，所以$K=100$，也就是取主成分矩阵的前K列，然后转置并乘以数据集矩阵X得到 100 维空间坐标矩阵z，再通过数据还原得到投影矩阵。这个投影矩阵就是 100 维空间下的降维结果。我们选取前 100 个投影点进行可视化，如图 21 所示。



(a) Original Image of Faces　(b) Reconstructed from top 100 principle components

图 21. PCA — 人脸降维前后对比

仔细对比降维前后的人脸图像，我们可以得出降维的效果。很明显，降维之后，人脸的大概外貌被保留，但是一些具体的细节被丢弃了。

最后我们来看一下 PCA 的可视化问题。
我们以之前聚类方法来压缩图片的样本为例。我们将之前 RGB 三维空间的样本在三维坐标系下绘制出来，每一个出现在三维坐标系中的点的坐标都相当于该样本点的 RGB 分量，如图 22 所示。

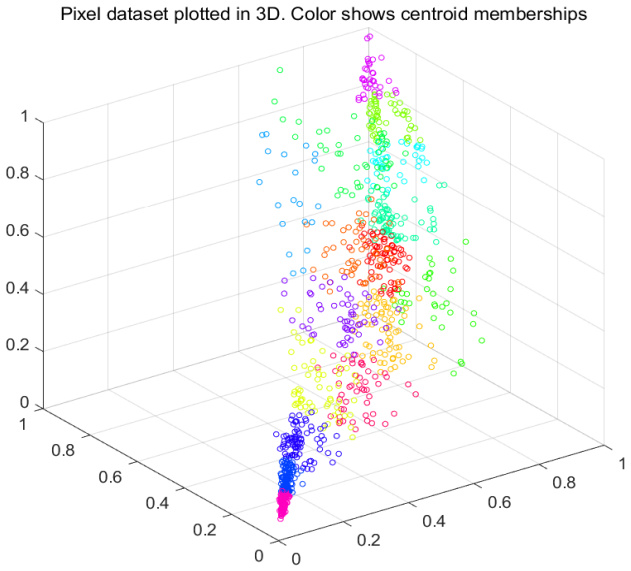


图 22. PCA — 原始数据集可视化

那么经过主成分提取然后进行降维之后得到的数据集在三维空间进行可视化的方法在之前已经叙述过了，也就是可以通过

$$\boldsymbol{x}_{approx} = \boldsymbol{U}_{reduced}\boldsymbol{z}$$

来还原。但是如果打算将其放入二维空间内进行可视化，我们就不必用上式来还原了。事实上，$\boldsymbol{z}$矩阵代表着其在二维空间的坐标。正如我们之前二维降到一维的情况一样，在那种情况下，$\boldsymbol{z}$矩阵代表一个个常数，也可以称之为一维空间的样本坐标。那么如果在一维空间进行可视化，那么就直接在数轴上可视化那些点的位置就可以了例如，图23(a)中所示的样本点，降维之后的坐标为$(x_1^{(i)}, x_2^{(i)}),\ 1 \leqslant i \leqslant m$，这依然是二维坐标。但是，投射到一维空间之后，就会显示出如图23(b)所示的情况。

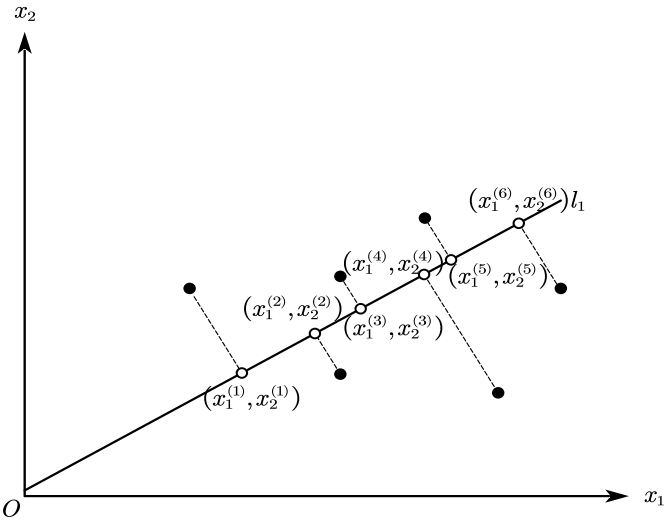


图 23(a). PCA — 投影的二维空间坐标

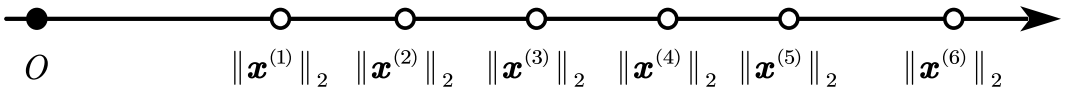


图 23(b). PCA — 投影的一维空间坐标

当然，有时候数轴方向可能会和图23(b)中相反，但是并不影响降维。

我们可以发现，图23(b)中的数值，正是降维之后$\boldsymbol{z}$矩阵的含义。所以我们直接在低维空间中对$\boldsymbol{z}$矩阵进行可视化即可。

所以图 23 中三维空间的降维表达，可以在二维平面上对$\boldsymbol{z}$矩阵进行可视化。我们将其可视化并于原数据集相对比，如图 24 所示。

这个结果是基本符合预期的，只不过方向正好相反。事实上，只要保持该面的法向量方向不变，不管这个二维平面如何翻转，投影都是正确的。

此外，我们控制了三维点及其二维投影点的颜色是相同的。这点并不难做到，因为三维空间中每一个点都有一个最近中心，然后由于投影的时候各样本之间顺序不变，一一对应，所以三维点与它的投影点之间是一一对应的。所以我们可以控制三维空间样本点和二维空间投影点的标号对应，颜色相同。

投影前后对比图如图 24 所示。

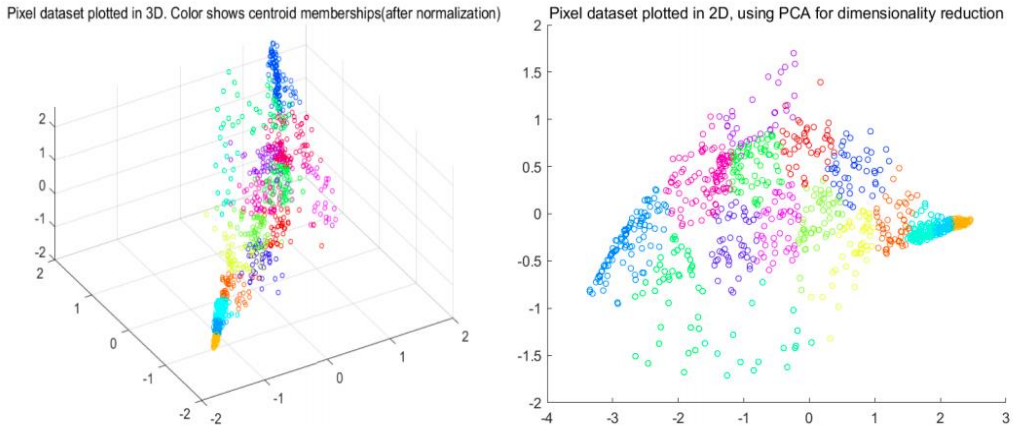


图 24. PCA — 投影对比

2. 通过自行收集的全世界多个国家 GDP 和人口数据（参考网站：https://data.worldbank.org/indicator/SP.POP.TOTL），实现发达国家/发展中国家/贫穷国家（或更多类别）的有效划分。

我们本次实验是要通过不同国家的人口以及 GDP 信息来进行分类。通过查资料发现，发达国家的分类标准有人均 GDP 以及社会发展水平等有关。所以我们不仅仅搜集了各国的人口以及 GDP，还有有一个量来代表社会发展水平。我们选取“居住在贫民窟中的人口占总人口的百分比”这个量来体现社会发展水平。这样人均 GDP 和社会发展水平组成了一个二维空间。我们的想法是，首先直接利用聚类方法来进行聚类得到一个结果，然后采用先降维然后聚类得方法得到一组结果，对比这两个方法的结果，从而实现发达国家/发展中国家/贫穷国家的有效划分。

数据集说明：由于个别国家数据并不完善，所以经统计，我们搜集了 135 个国家的人口(population)、GDP（US dollar）以及居住在贫民窟中的人口占总人口的百分比(population living in slums, %)，保存在 Excel 数据表中。

直接聚类：首先我们将找到数据的 135 个国家的样本并计算人均 GDP，并通过散点图来绘制人均 GDP 和在贫民窟中的人口占总人口的百分比之间的关系。可以看出二者有一个比较明显的负相关，如图 25 所示。

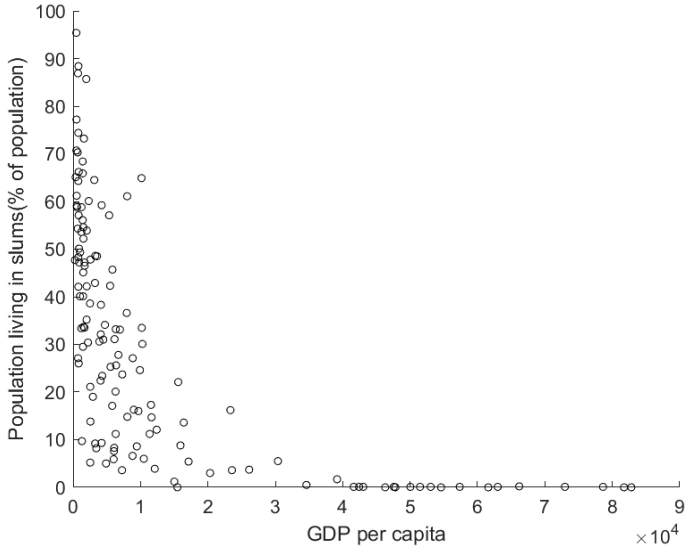


图 25. 样本分布情况

我们可以看到，样本点分布非常不均匀，只划分为三个类的话肯定会造成一个类中样本点很多而另两个类的样本很少。该图经过放大之后，依然显示分布非常不均匀，所以我们采取的方法就是将样本分为多个类，然后对每一个分类结果来进行评价。这样我们保证每一个类样本数不会太多，这样中心点才能较好地反应整个类的特征。

我们先将样本分为 4 类，然后进行聚类分析。我们要注意，我们将数据保存在 Excel 中读取保存成元组形式，然后再提取数据来进行聚类，达到收敛状态之后，得到的结果如下所示：

第 1 类国家有：Afghanistan, Angola, Armenia, Burundi, Benin, Burkina Faso, Bangladesh, Belize, Bolivia, Central African Republic, Cote d'Ivoire, Cameroon, Congo, Comoros, Djibouti, Egypt, Ethiopia, Georgia, Ghana, Guinea, Gambia, Guinea-Bissau, Guatemala, Honduras, Haiti, Indonesia, India, Jordan, Kenya, Kyrgyz Republic, Cambodia, Lao PDR, Liberia, Lesotho, Morocco, Moldova, Madagascar, Mali, Myanmar, Mongolia, Mozambique, Mauritania, Malawi, Niger, Nigeria, Nicaragua, Nepal, Pakistan, Philippines, Rwanda, Sudan, Senegal, Sierra Leone, El Salvador, Sao Tome and Principe, Eswatini, Chad, Togo, Tajikistan, Timor-Leste, Tunisia, Tanzania, Uganda, Ukraine, Uzbekistan, Vietnam, Yemen, Zambia, Zimbabwe, 共计 69 个，该类国家的人均 GDP 为 **1926.98** 美元，居住在贫民窟的人占总人口的比例为 **46.2188%**；

第 2 类国家有：Chile, Spain, Greece, Croatia, Hungary, Italy, Panama, Poland, Portugal, Saudi Arabia, Slovenia, Trinidad and Tobago, 共计 12 个，该类国家的人均 GDP 为 **21157.2** 美元，居住在贫民窟的人占总人口的比例为 **6.9675%**；

第 3 类国家有：Australia, Austria, Belgium, Canada, Switzerland, Germany, Denmark, Finland, France, United Kingdom, Ireland, Iceland, Japan, Netherlands, Norway, New Zealand, Singapore, Sweden, United States, 共计 19 个，该类国家的人均 GDP 为 **56914.9** 美元，居住在贫民窟的人占总人口的比例为 **0.159632%**；

第 4 类国家有：Argentina, Bosnia and Herzegovina, Belarus, Brazil, China, Colombia, Costa Rica, Caribbean small states, Cuba, Dominican Republic, Ecuador, Fiji, Gabon, Equatorial Guinea, Grenada, Guyana, Iran, Iraq, Jamaica, Lebanon, St. Lucia, Maldives, Mexico, North Macedonia, Montenegro, Namibia, Peru, Paraguay, Romania, Russian Federation, Serbia, Suriname, Thailand, Turkey, South Africa, 共计 35 个，该类国家的人均 GDP 为 **8165.32** 美元，居住在贫民窟的人占总人口的比例为 **23.5451%**。

分类图如图 26 所示。

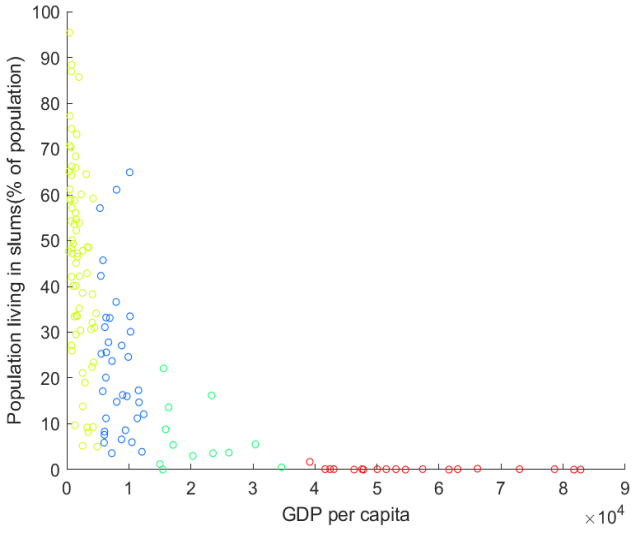


图 26. 样本分类情况

一般来说，人均 GDP 值越高、居住在贫民窟中的人口占总人口的百分比越低，这个国家就越发达。所以，我们有如下结论：

1） 四类样本中，最靠左的红色样本区域对应第 1 类国家，人均 GDP 为 **1926.98** 美元，居住在贫民窟的人占总人口的比例为 **46.2188%**，被归类为贫穷国家，共 69 个；

2） 与红色样本右相邻的簇，属于第 4 类国家，人均 GDP 为 **8165.32** 美元，居住在贫民窟的人占总人口的比例为 **23.5451%**，被归类为发展中国家，共 35 个；

3） 剩下的两个簇，分别属于第 2、3 类国家，人均 GDP 为 20000-50000 美元，居住在贫民窟的人占总人口的比例低于 10%，被归类为发达国家，共计 31 个；其中第 3 类国家，人均 GDP 超过 50000 美元，居住在贫民窟的人占总人口的比例低于 1%，可以被列为主流发达国家，共 19 个；第 2 类国家人均 GDP 为 20000 美元左右，居住在贫民窟的人占总人口的比例为 5%-10%，可被列为中等发达国家。

我们再通过 PCA 的方法来看一下是否可以得到同样的结果。

我们将样本从二维空间降到一维空间，然后在一维空间作聚类，再反馈到二维空间。经过一维空间的聚类之后，还原到二维空间，可以看到分类如图 27 所示。

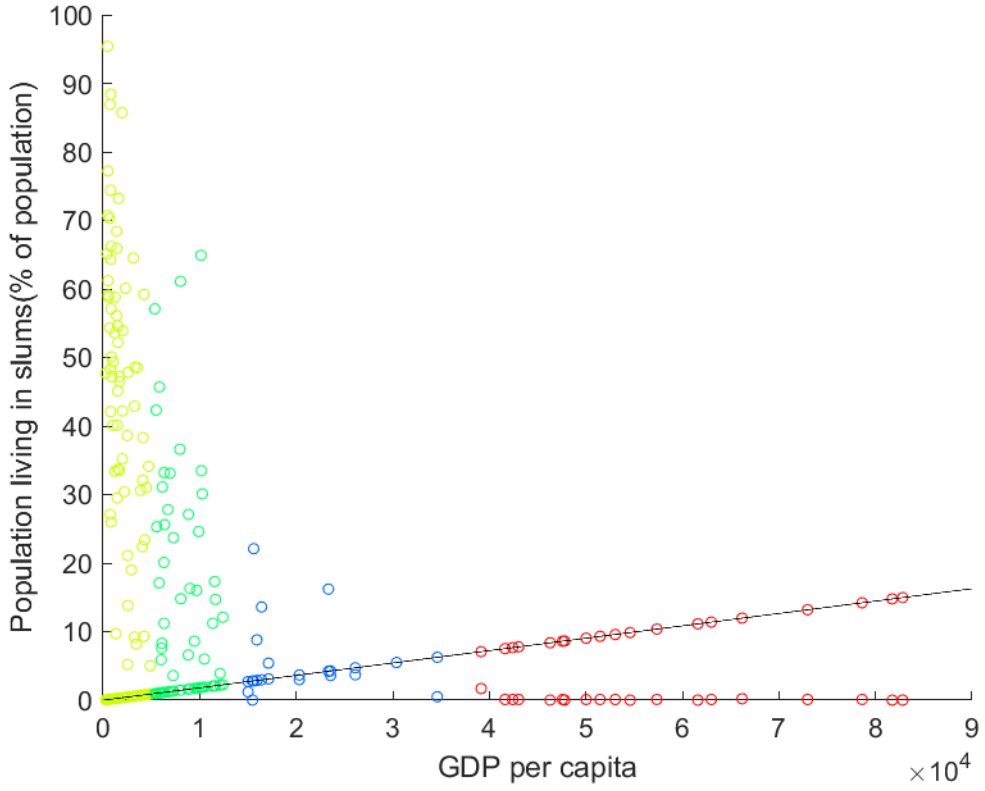


图 27. 先降维再分类结果

我们对比前后的聚类结果，如图 28 所示。我们可以看到，颜色仅仅是为了作出区分，不考虑两边颜色的不同，两种分类方法从图中反映出来是基本一致的。事实上，经过各类国家也可以看出两次分类的结果完全一致。那么降维之后聚类得出的各个类，我们如何用中心点来反映它们的平均水平呢？

对于每一个类，我们在输出它们的国家名的时候，用 MATLAB 的find 函数来找到它们在样本集中的索引，然后就可以通过逐次求和并除以该类样本总数得到中心点坐标。事实上这也就是中心点的定义，我们只不过用手工计算的方法来还原这个中心点。

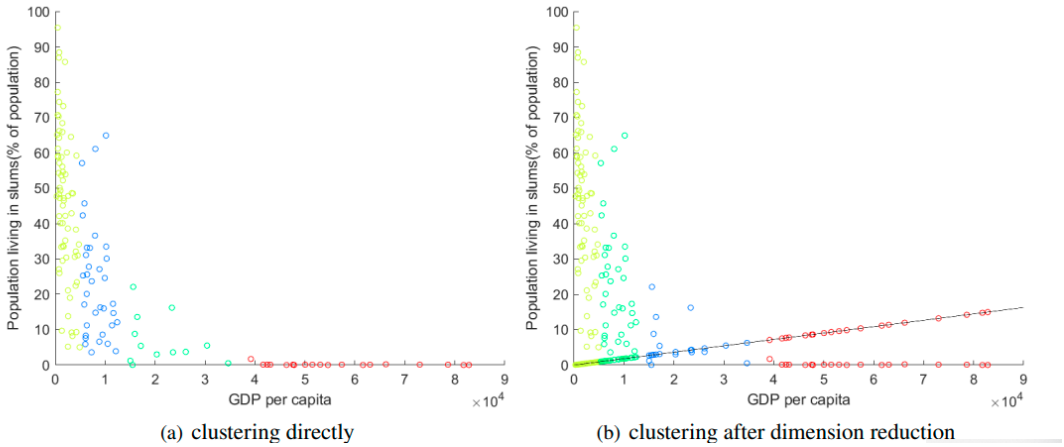


图 28. 两次分类结果对比，发现结果基本一致

分析这次的输出结果，我们发现中心点坐标有一些差别，这是由于计算保留舍入的误差造成的。结果对比如下：

1） 主流发达国家（19 个）：

直接聚类结果：人均 GDP 为 56914.9 美元，居住在贫民窟的人占总人口的比例为 0.159632%；

降维后聚类结果：人均 GDP 为 53599.3 美元，居住在贫民窟的人占总人口的比例为 0.154368%；

2） 中等发达国家（12 个）：

直接聚类结果：人均 GDP 为 21157.2 美元，居住在贫民窟的人占总人口的比例为 6.9675%；

降维后聚类结果：人均 GDP 为 19729.7 美元，居住在贫民窟的人占总人口的比例为 6.5175%；

3） 发展中国家（35 个）：

直接聚类结果：人均 GDP 为 8165.32 美元，居住在贫民窟的人占总人口的比例为 23.5451%；

降维后聚类结果：人均 GDP 为 7983.25 美元，居住在贫民窟的人占总人口的比例为 22.8137%；

4） 贫穷国家（69 个）：

直接聚类结果：人均 GDP 为 1926.98 美元，居住在贫民窟的人占总人口的比例为 46.2188%；

降维后聚类结果：人均 GDP 为 1902.58 美元，居住在贫民窟的人占总人口的比例为 45.7333%；

至此，我们通过直接聚类和降维后聚类的方法完成了国家的划分。

五、实验总结与体会

本次实验首先通过实验文档实现了聚类以及 PCA 算法，并分别将其应用于图像压缩与人脸特征提取。在实验第二部分，我们通过自己寻找数据，分别用直接聚类和聚类与 PCA 的结合来实现了国家的划分。

本次实验内容比较多，但是通过这次实验，我又学习了两个机器学习的基本算法。值得一提的是，实验中涉及的聚类和 PCA 算法是最基本的聚类和 PCA，在聚类领域中还有高斯混合聚类、密度聚类以及层次聚类等等，而 PCA 属于降维及度量学习的内容，降维方法还有很多，类似核化线性降维(kernelized PCA)等等。掌握好最基本的算法有助于在以后接触一些更加深入的算法。此外，在机器学习的学习过程中，数学知识非常重要，只有经过数学公式的推导，才能很好地掌握算法。

六、实验附件

关于国家划分的部分：

1. data.xlsx 记录的是 135 个国家的人口数、GDP 以及在贫民窟中的人口占总人口的百分比；
2. nationdividing.m 是直接聚类方法来对国家进行划分；
3. nationdividingPCA.m 是结合了降维方法来聚类从而实现国家的划分。

指导教师批阅意见：

成绩评定：

指导教师签字： 贾森、徐萌

2021 年 6 月 25 日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。