实验报告-ex7

编程软件：MATLAB，notepad++

编程环境：Windows10

作业任务：本次实验需要完成K-means聚集算法来对图像进行压缩，另外，还需要利用PCA来对面部图像进行降维处理。有ex7和ex7\_pca两个函数，函数中套用了相应的函数，所以需要根据这两个目录函数的指示和实验手册，完成对应的5个部分。

1 K-means Clustering

这一部分先实现了2D维度的dataset上的K-means聚集算法，这样对K-means有了直观的理解后，再对图片进行压缩，减少图片中色彩的数量。

1.1 Implementing K-means

由前面所学可知，K-means聚集算法核心内容如下：

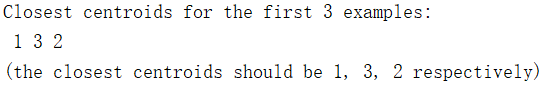
1）随机初始化centroids

2）对两步进行迭代：对每个data分配对应的距离最近的centroid；重新计算不同centroid的平均位置

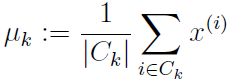
以下即分别实行迭代的两部分：

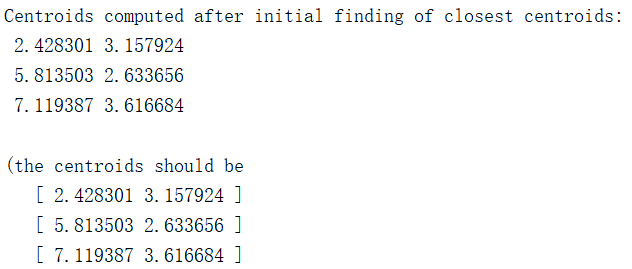
1.1.1 Finding closest centroids

这一部分实现的是对每个data分配centroid的步骤。可知公式为：。只需要用两个嵌套循环，对每个x(i)查找全部μ，找到最小距离即可。



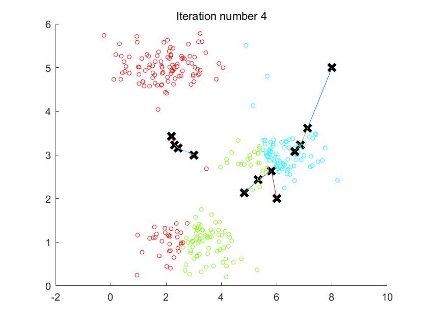
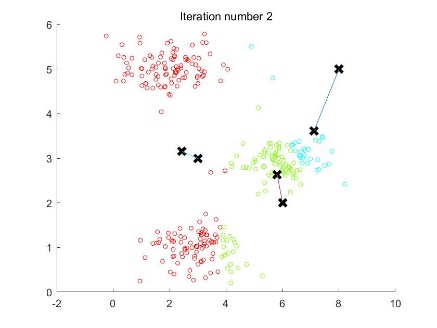
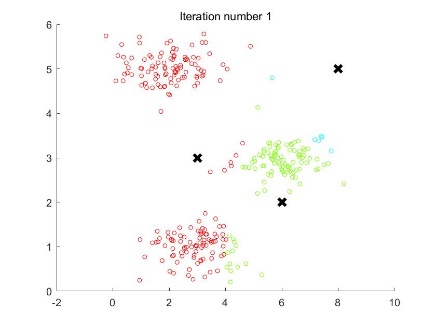
1.1.2 Computing centroid means

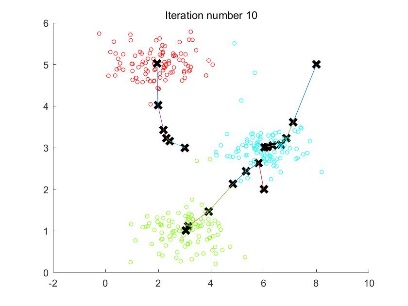
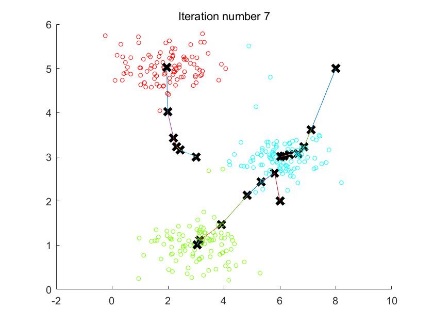
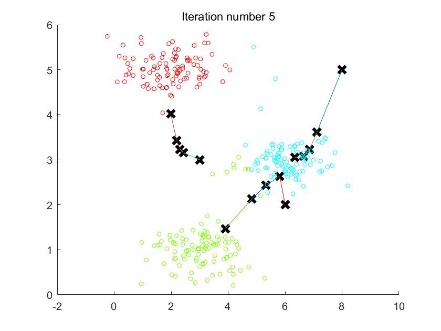
这一部分实现的是计算分配后每种centroid的平均位置，公式为：。方法类似于上面，用两个嵌套循环，判断当前data的centroid是否是i，如果是则加上，最后取平均值即可。



1.2 K-means on example dataset

这一部分利用上述已经完成的两个函数进行K-means算法的运用。ex7调用了runkMeans函数进行了10次迭代，每次都画出对应的centroid点的变化。我们发现从第7次迭代开始，centroid点基本上不发生很大的改变，说明已经成功实现：





1.3 Random initialization

在ex7中的centroid的位置是事先给出的，但是在实际情况中，我们需要对centroid进行随机分配位置。而kMeansInitCentroids函数就实现了这一点。

首先对X的所有行随机打乱行号，然后取前K行即可。

1.4 Image compression with K-means

这一部分我们将利用已经实现好的K-means聚集算法实现对图片的压缩。已知一个彩色图片，我们用RGB编码将其抽象成数字化表示。我们要做的就是将丰富的色彩压缩成16中颜色。

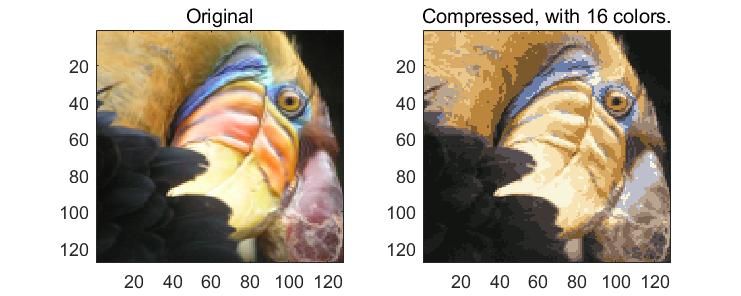
其中，表示图片的时候，我们只需要存储这16种颜色的RGB值，对图片中的每一个像素，我们只需要存储该像素颜色的索引值。（比如4×4可以表示16种可能性）、

抽象成K-means问题，我们需要利用该算法将全部的图片颜色信息分成16类。

1.4.1 K-means on pixels

这一部分我们需要先将图片用A = imread('bird small.png');转化成三列矩阵。其中前两列指示了像素的位置，最后一列指示了代表的红绿蓝哪种颜色。所以对于本实验中用到的128×128的图片，抽象出来应该是16384×3的矩阵，而我们的K-means聚集算法就是作用于该矩阵上。

没有运用算法之前，每一个128×128的像素点需要24位，所以一共128×128×24=393216位。而运用了聚集算法之后，我们虽然另外需要16中颜色的存储，但是每个像素点只需要4位即可，即16×24+128×128×4=65920位。所以我们对图片进行了压缩。

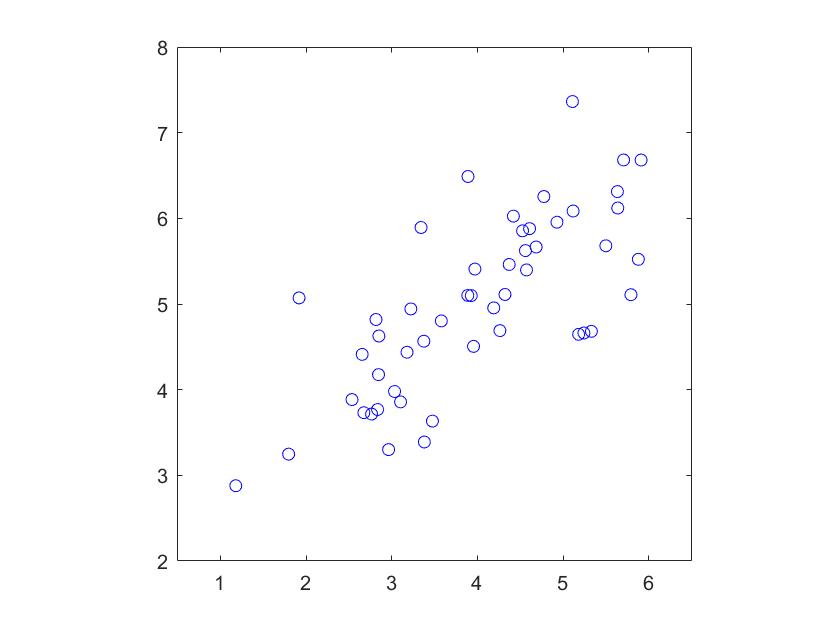


2 Principal Component Analysis

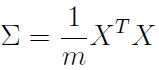
这一部分先实现了2D维度的PCA实现降维，这样对PCA有了直观的理解后，再对面部图片进行处理。

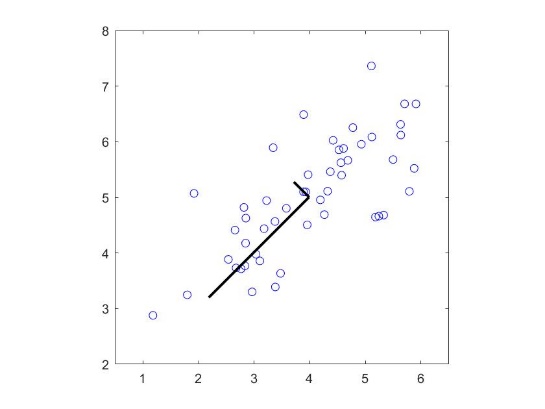
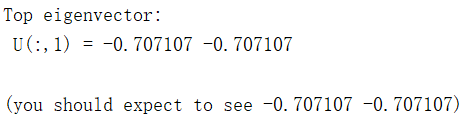
2.1 Example Dataset

首先我们处理的是一个2D的数据。，如下为其可视化：



2.2 Implementing PCA

PCA一共有三大部分，首先对数据进行mean normalization和feature scaling处理，在本实验中已经完成。其次计算covariance matrix ，最后利用[U, S, V] = svd(Sigma)计算。

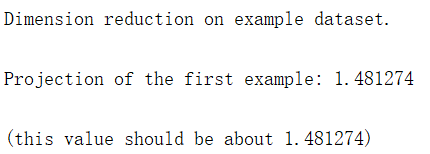
 

2.3 Dimensionality Reduction with PCA

这里就利用刚刚实现的PCA来对dataset进行降维处理。

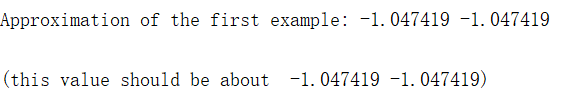
2.3.1 Projecting the data onto the principal components

该部分比较简单，直接取U的前K列，然后与X进行矩阵相乘即得到。



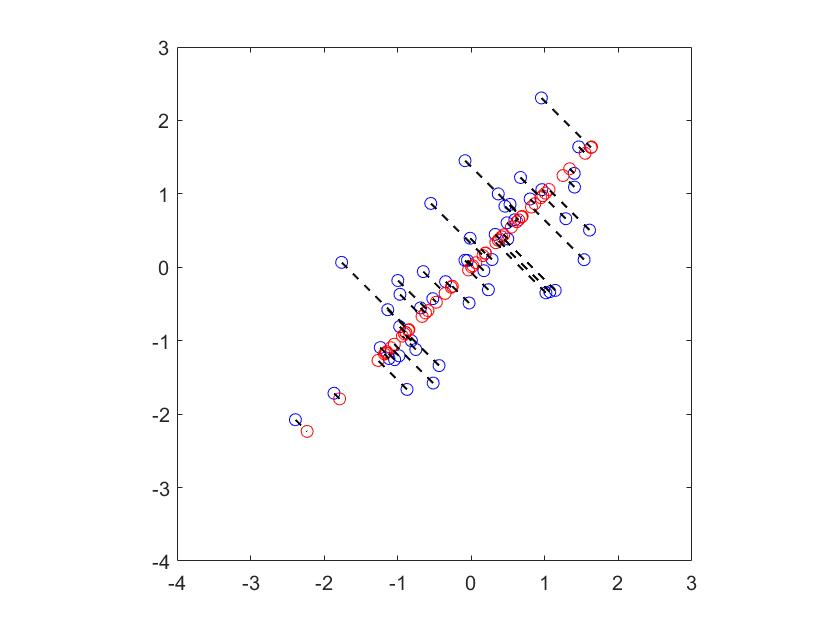
2.3.2 Reconstructing an approximation of the data

除了将二维映射到一维上，还可以重构回二维。



2.3.3 Visualizing the projections

这一部分在图上标注了映射（红色）和压缩重现（蓝色）的点



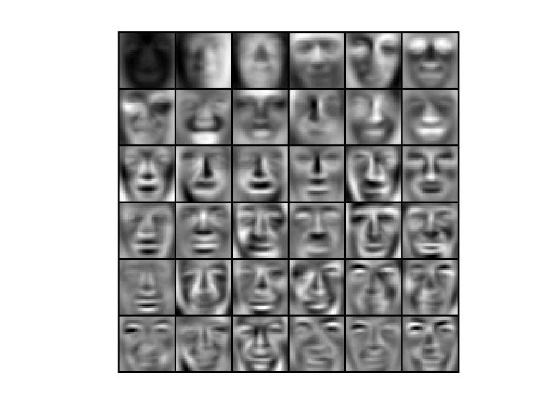
2.4 Face Image Dataset

这一部分就将PCA应用到了实际问题上。对于新的X，每一行代表一个脸部的图片，现在对图片进行可视化。



2.4.1 PCA on Faces

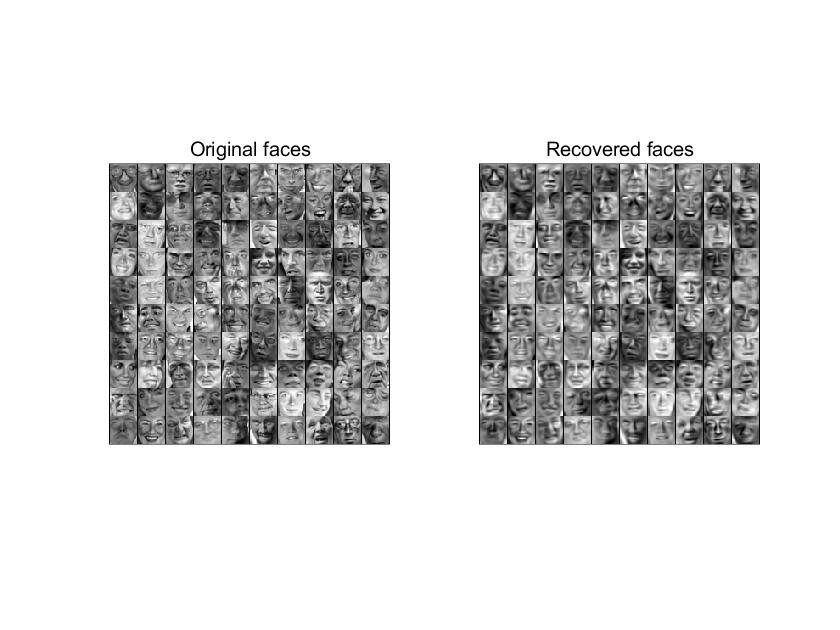
这一部分将对于新的X实现PCA，从mean normalization到计算U。其中，U的每一行代表一个主成分，因此我们可以将其可视化。



2.4.2 Dimensionality Reduction

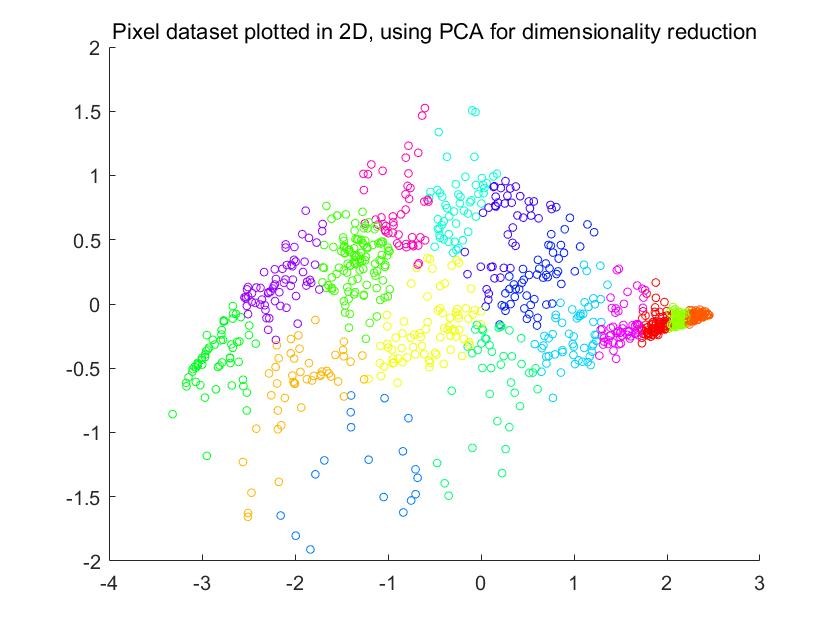
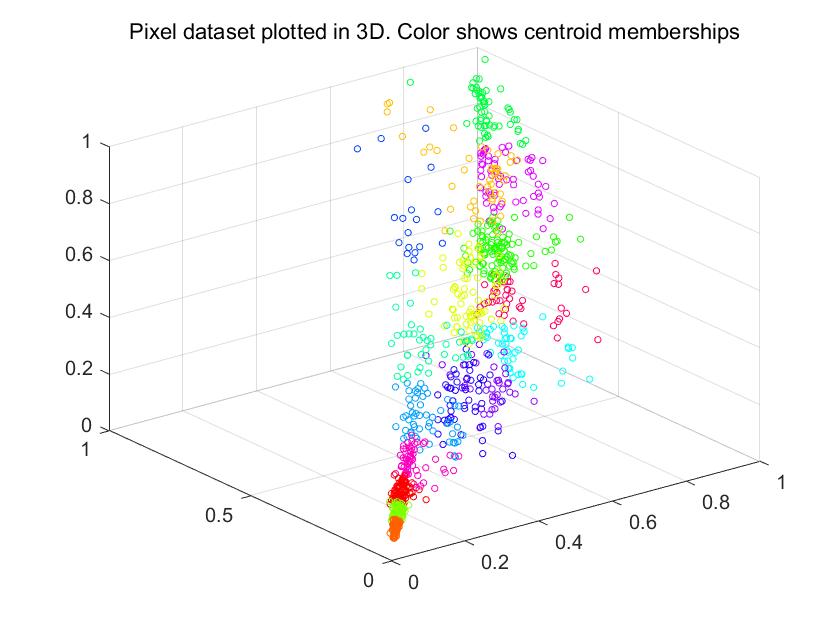
这一部分我们就对原先1024维度进行降维，降到100维，即只有100个主成分。为了看到降维后再压缩重现有什么改变，我们将两次的都进行了可视化。发现大体轮廓相似，只是一些具体的细节丢失。

这种方法帮助我们在实现神经网络等算法的时候加快速度。



2.5 Optional (ungraded) exercise: PCA for visualization

分别用PCA降维，将数据可视化为3D和2D。不同颜色表示不同的centroid。



○ 至此实验成功完成

