编程练习 7：

*K* 均值聚类和主成分分析

机器学习

# 介绍

在本练习中，您将实现 *K* 均值聚类分析算法，并将其应用于压缩图像。在第二部分中，您将使用主成分分析来查找人脸图像的低维表示。在开始编程练习之前，我们强烈建议您观看视频讲座并完成相关主题的评论问题。

要开始练习，您需要下载入门代码并将其内容解压缩到要完成练习的目录中。如果需要， 请在 开始本练习之前使用 Octave/MATLAB 中的 cd 命令更改为此目录。

您还可以在课程网站的“环境设置说明”中找到有关安装Octave / MATLAB的说明。

## 本练习中包含的文件

ex7.m - 用于 K-means ex7 pca.m 的第一个练习的 O ctave/MATLAB 脚本 - 用于 PCA ex7data1.mat 上第二个练习的 Octave/MATLAB 脚本 - PCA ex7data2.mat 的示例数据集 - K-means ex7faces.mat 的示例数据集 - Faces Dataset bird small.png - 示例 Image displayData.m - 显示存储在矩阵 drawLine.m 中的 2D 数据 - 在现有图形 plotDataPoints.m 上绘制一条线 - *K* 均值质心图的初始化进度Means.m - 在 K 均值继续 runkMeans.m 时绘制 *K* 均值的每一步 - 运行 K-means 算法 submit.m - 将您的解决方案发送到我们的服务器的提交脚本[*？*] pca.m - 执行主成分分析

[*?*] projectData.m - 将数据集投影到较低维的空间中

[*?*] recoverData.m - 从投影中恢复原始数据

[*?*] findClosestCentroids.m - 查找最近的质心（用于 K-means）

[*?*] computeCentroids.m - Compute 质心均值（用于 *K* 均值）

[*?*] kMeansInitCentroids.m - *初始化* K-means 质心

*?* 表示您需要完成的文件

在本练习的第一部分中，您将使用脚本 ex7.m，在第二部分中，您将使用 ex7 pca.m。这些脚本为问题设置数据集，并调用您将要编写的函数。您只需按照此赋值中的说明修改其他文件中的函数。

## 从何处获取帮助

本课程中的练习使用Octave[[1]](#footnote-1) 或MATLAB，这是一种非常适合数值计算离子的高级编程语言。如果您没有安装Octave或MATLAB，请参阅课程网站“环境设置说明”中的安装说明。

在 Octave/MATLAB 命令行中，键入 help 后跟函数名称将显示内置函数的文档。例如，帮助绘图将显示绘图的帮助信息。有关倍频程函数的更多文档，请参阅[倍频程文档页面。](http://www.gnu.org/software/octave/doc/interpreter/) MATLAB 文档可以在 [MATLAB 文档页面上](http://www.mathworks.com/help/matlab/?refresh=true)找到[。](http://www.mathworks.com/help/matlab/?refresh=true)

我们还强烈建议使用在线 **讨论** 与其他学生讨论练习。但是，不要查看他人编写的任何源代码或与他人共享您的源代码。

# *K* 均值聚类

在本练习中，您将实现 *K* 均值算法并将其用于图像压缩。您将首先从示例 2D 数据集开始，该数据集将帮助您直观地了解 *K* 均值算法的工作原理。之后，您可以通过将 图像中出现的颜色数量减少到仅该图像中最常见的颜色来使用用于图像压缩的K-means算法。您将在这部分练习中使用 ex7.m。

## 实现 *K* 均值

*K* 均值算法是一种自动将相似的数据示例聚类在一起的方法。具体来说，你会得到一个训练集 {*x*（1）*,...，x*（*m*）}（其中 *x*（*i*） ∈ R*n*），并希望将数据分组到几个有凝聚力的“聚类”中。*K-means*背后的直觉是一个迭代过程，它从猜测初始质心开始，然后通过重复将示例分配给它们最近的质心，然后根据赋值重新计算质心来完善此猜测。

*K* 均值算法如下：

|  |
| --- |
| % Initialize centroids centroids = kMeansInitCentroids（X， K）; for iter = 1：迭代  % 群集分配步骤：将每个数据点分配给  % 最近质 心。 idx（i） 对应于索引 cˆ（i）  分配给示例 i idx 的质心的百分比 = findClosestCentroids（X， 质心）;  % 移动质心步长：基于质心的计算均值  % 赋值 质心 = computeMeans（X， idx， K）;  结束 |

算法的内部循环重复执行两个步骤：（i）将每个训练示例*x*（*i*）分配给其最近的质心，以及（ii）使用分配给它的点重新计算每个质心的平均值。*K* 均值算法将始终收敛于质心的某个最终均值集。请注意，收敛解可能并不总是理想的，这取决于质心的初始设置。因此，在实践中，*K* 均值算法通常使用不同的随机初始化运行几次。在来自不同随机初始化的这些不同解决方案之间进行选择的一种方法是选择具有最低成本函数值（失真）的解决方案。

您将在下一秒中分别实现 K 均值算法的两个阶段。

### 查找最近的质心

在 *K* 均值算法的“聚类分配”阶段，该算法在给定质心的当前位置的情况下，将每个训练示例 *x*（*i*） 分配给其最近的质心。具体来说，对于 我们设置的每个示例 *i*

*c*（*i*） ：= *j* 最小化 ||*x*（*i*） − *μj*||2*、*

其中 *c*（*i*）  是最接近 *x*（*i*） 的质心的索引，*μj*  是第 j'th 质心的位置（值 ）。请注意，*c*（*i*） 对应于起始代码中的 idx（i）。

您的任务是完成 findClosestCentroids.m 中的代码。此函数采用数据矩阵 X 和质心内所有质心的位置 ，并应输出一个保存索引的一维数组 idx（{1 中的值*,...，K*}，其中 *K* 是最接近每个训练示例的质心的 tota l 个质心）。

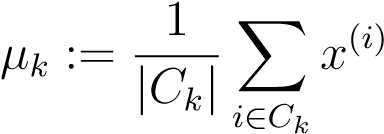
您可以使用每个训练示例和每个质心上的循环来实现这一点。

在 findClosestCentroids.m 中完成代码后，脚本 ex7.m 将运行您的代码，您应该会看到与前 3 个示例的质心赋值相对应的输出 [1 3 2]。

*您现在应该提交您的解决方案。*

### 计算质心装置

给定每个点对质心的赋值，算法的第二阶段为每个质心重新计算分配给它的点的平均值。具体来说，对于我们设置的每个质心 *k*



其中 *Ck* 是分配给质心 k 的示例集。具体来说，如果两个例子说*x*（3）和*x*（5）被分配到质心*k* = 2，那么你应该更新 

现在，您应该在 computeCentroids.m 中完成代码。您可以使用质心上的循环来实现此函数。您也可以在示例中使用循环;但是，如果您可以使用不使用此类循环的矢量化 imp lementation，则代码可能会运行得更快。

在 computeCentroids.m 中完成代码后，脚本 ex7.m 将运行您的代码，并在 K 均值的第一步之后输出质心。

*您现在应该提交您的解决方案。*

## 示例数据集上的 K 均值

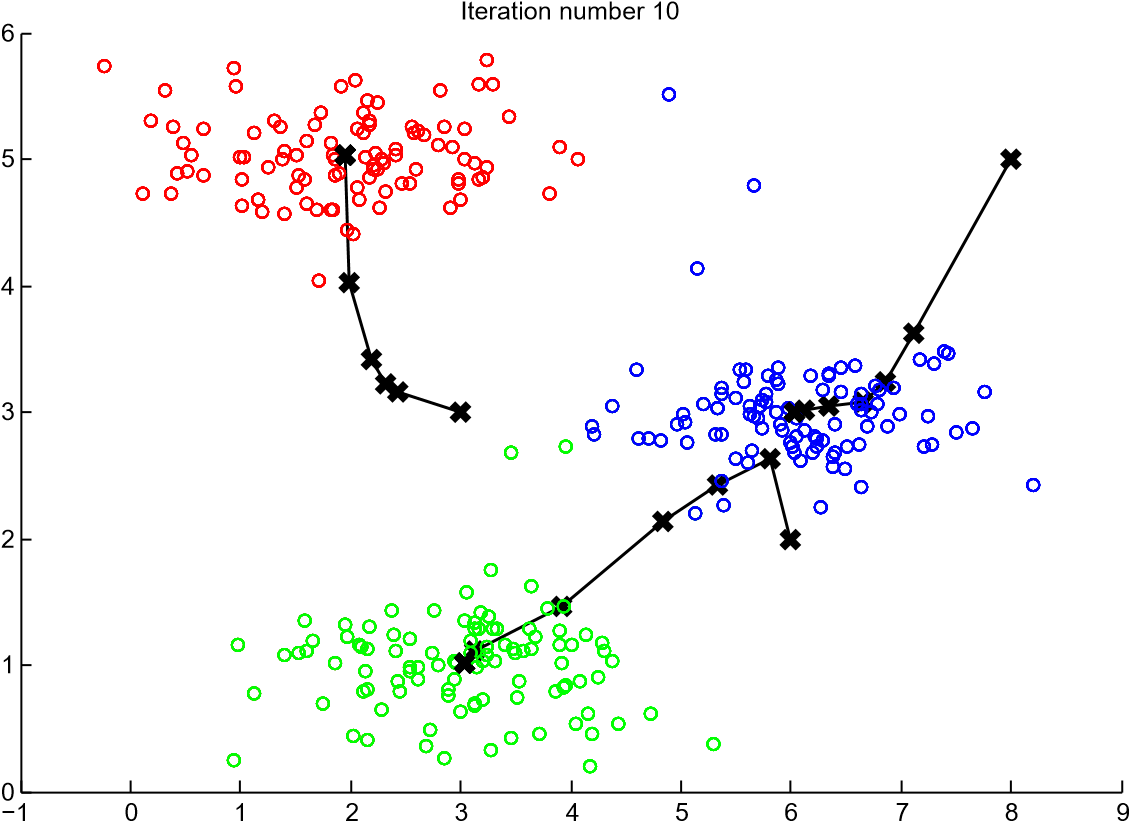


图 1：预期输出。

完成两个函数（findClosestCentroids 和 computeCentroids）后，ex7.m 中的下一步将对玩具 2D 数据集运行 K-means 算法，以帮助您了解 K-means 的工作原理。您的函数是从 runKmeans.m 脚本内部调用的。我们鼓励您查看该功能以了解其工作原理。请注意，该代码调用您在循环中实现的两个函数。

运行下一步时，K-means 代码将生成一个可视化效果，引导您在每次迭代时完成算法的进度。多次按 *Enter* 键可查看 *K* 均值算法的每个步骤如何记录质心和聚类分配。最后，您的图形应如图 1 所示。

## 随机初始化

ex7.m 中示例数据集的质心初始分配经过精心设计，因此您将看到与图 1 中相同的图形。在实践中，初始化质心的一个好策略是从训练集中选择随机示例。

在本部分练习中，您应该使用以下代码完成函数 kMeansInitCentroids.m：

% 将 c枚举初始化为随机示例

% 随机重新排序示例 的索引 randidx = randperm（size（X， 1））; % 将前一个 K 个示例作为质心 质心 = X（randidx（1：K），:);

上面的代码首先随机置换示例的索引（使用 randperm）。然后，它 根据索引的随机排列选择前 K 个示例。这允许随机选择示例，而不会有选择 same 示例两次的风险。

*您无需为本部分练习提交任何内容。*

## 使用 *K* 均值进行图像压缩



图 2：原始 128x128 图像。

在本练习中，您将对图像压缩应用 *K* 均值。 在

图像的直接 24 位 col或表示形式，[[2]](#footnote-2) 每个像素表示为三个 8 位无符号整数（范围从 0 到 255），用于指定红色、绿色和蓝色强度值。此编码通常 称为 RGB 编码。我们的图像包含数千种of颜色，在这部分练习中，您将把颜色数量减少到16种颜色。

通过进行这种简化，可以有效地表示（压缩）照片。具体来说，您只需要存储 16 种选定颜色的 RGB 值，对于图像中的每个像素，您现在只需在该位置存储颜色的索引 （只需 4 位即可表示 16 种可能性）。

在本练习中，您将使用 *K* 均值算法来选择将用于表示压缩图像的 16 种颜色。具体而言，您将原始图像中的每个像素视为数据示例，并使用 *K* 均值算法查找 16 种颜色，这些颜色对 3 维 RGB 空间中的像素进行最佳分组（聚类）。计算图像上的聚类质心后，将使用 16 种颜色替换原始图像中的像素。

### 像素上的 K 均值

在Octave/MATLAB中，可以按如下方式读取图像：

|  |
| --- |
| % 负载 128x128 彩色图像（小鸟.png）  A = imread（'birdsmall.png'）;  % 您需要安装映像包才能使用  % imread.如果未安装映像包，则  % 应 改为将以下行更改为  %  % load（'birdsmall.mat'）; % 将图像加载到变量 A 中 |

这将创建一个三维矩阵 A，其前两个索引标识像素位置，最后一个索引表示红色、绿色或蓝色。例如，A（50， 33， 3） 给出了第 50 行和第 33 列处像素的蓝色强度。

ex7.m 中的代码首先加载图像，然后对其进行重新调整以创建 m × 3 个像素颜色矩阵（其中 *m* = 16384 = 128 × 128），并对其调用 K-means 函数。

找到表示图像的顶部 *K* = 16 种颜色后，现在可以使用 findClosestCentroids 函数将每个像素位置指定给其最近的质心 。这允许您使用每个像素的质心分配来表示原始图像。请注意，您已经大大减少了描述图像所需的位数。原始图像对于 128×128 个像素位置中的每个位置都需要 24 位，因此总大小为 128×128×24 = 393*，*216 位。新的表示形式需要一些以字典形式存储的开销，每种颜色都需要24位，但图像本身只需要每个像素位置4位。因此，最终使用的位数为16 × 24 + 128 ×128 × 4 = 65*，*920位，这相当于将原始图像压缩约6倍。

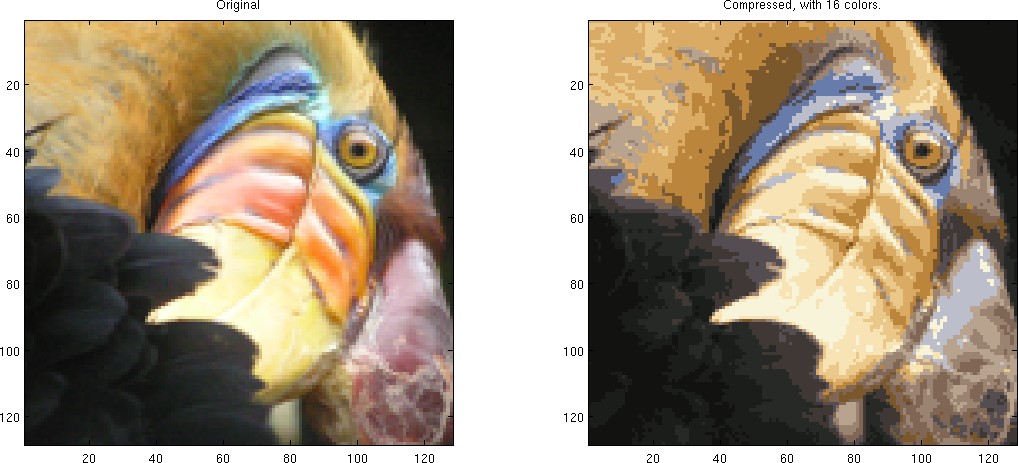
1 

图3：原始和重建的图像（当使用K-means压缩图像时）。

最后，您可以通过仅基于质心分配重建图像来查看压缩的效果。具体而言，您可以将每个像素位置替换为分配给它的质心的平均值。图 3 显示了我们观察到的重建。尽管生成的图像保留了原始图像的大部分特征，但我们也看到了一些压缩伪像。

*您无需为本部分练习提交任何内容。*

## 可选（未评分）练习：使用您自己的图像

在本练习中，修改我们提供的代码以在您自己的映像之一上运行。请注意，如果您的图像非常大，则 *K* 均值可能需要很长时间才能运行。因此，我们建议您在 运行 cod e 之前将图像大小调整为可管理的大小。您还可以尝试改变 *K* 以查看对压缩的影响。

# 主成分分析

在本练习中，您将使用主成分分析 （PCA） 来执行降维。您将首先使用示例 2D 数据集 进行试验，以深入了解 PCA 的工作原理，然后将其用于包含 5000 张人脸图像数据集的更大数据集。

提供的脚本 ex7 pca.m 将帮助您逐步完成练习的前半部分。

## 示例数据集

为了帮助您了解PCA的工作原理，您将首先从2D数据集开始，该数据集具有一个较大的变化方向和一个较小的变化方向。脚本 ex7 pca.m 将绘制训练数据（图 4）。在本部分练习中，您将直观地看到使用 PCA 将 d ata从 2D 降低到 1D 时发生的情况。在实践中，您可能希望将数据从256个维度减少到50个维度。但是在此示例中使用较低维度的数据使我们能够更好地可视化算法。

1

2

3

4

5

6

2

3

4

5

6

7

8

图 4：示例数据集 1

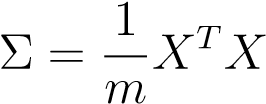
## 实施 PCA

在本部分练习中，您将实现 PCA。PCA 由两个计算步骤组成：首先，计算数据的协方差矩阵。

然后，使用Octave / MATLAB的 SVD 函数来计算eigenvectors *U*1*，U*2*,...，Un*.这些将对应于数据中变异的主成分。

在使用 PCA 之前，重要的是首先通过从数据集中减去每个特征的平均值来规范化数据，然后缩放每个维度，以便它们处于相同的范围内。在提供的脚本 ex7 pca.m 中，已使用功能规范化函数为您执行了此规范化。

对数据进行规范化后，可以运行 PCA 来计算主成分。您的任务是完成 pca.m 中的 e 个代码 来计算数据集的主成分。首先，您应该计算数据的协方差矩阵，该矩阵由下式给出：



其中 *X* 是行中包含示例的数据矩阵， *m* 是示例数。请注意，Σ 是 n × *n* 矩阵，而不是求和运算符。

计算协方差矩阵后，可以对其运行 SVD 以计算主成分。在Octave/MATLAB中，您可以使用以下命令运行SVD： [U，S， V] = svd（Sigma），其中 U 将包含主成分， S 将包含对角矩阵。

1

2

3

4

5

6

2

3

4

5

6

7

8

图 5：数据集的计算特征向量

完成 pca.m 后，ex7 pca.m 脚本将在示例数据集上运行 PCA 并绘制找到的相应主成分（图 5）。该脚本还将输出找到的顶部主成分（特征向量），您应该期望看到大约 [-0.707 -0.7 07] 的输出。（Octave/MATLAB可能会输出这个的负数，因为 *U*1 和 −*U*1 是第一个主成分的同样有效的选择。

*您现在应该提交您的解决方案。*

## 使用 PCA 进行降维

计算主成分后，您可以使用它们通过将每个示例投影到较低维空间*（x*（*i*）→*z*（*i*））上来减小数据集的特征维度（例如，将数据从2D投影到1D）。在本部分练习中，您将使用 PCA 返回的特征向量，并将示例数据集投影到一维空间中。

在实践中，如果您使用的是线性回归或神经网络等学习算法 ，则现在可以使用投影数据而不是原始数据。通过使用投影数据，可以更快地训练模型，因为输入中的维度较少。

### 将数据投影到主成分上

您现在在 projectData.m 中完成代码。具体而言，您将获得一个数据集 X、主成分 U 以及要简化为 K 的所需维数。您应该将 X 中的每个示例投影 到 U 中的前 K 个组件上。请注意， U 中的前 K 分量由 U 的前 K 列给出，即 U reduce = U（：， 1：K）。。

完成 projectData.m 中的代码后，ex7 pca.m 会将第一个示例投影到第一个维度上，您应该看到大约 1.481 的值（或者可能是 -1.481，如果您得到 −*U*1 而不是 *U*1）。

*您现在应该提交您的解决方案。*

### 重建数据的近似值

将数据投影到较低维空间后，可以通过将数据投影回或初始高维空间来近似地恢复数据。您的任务是完成 recoverData.m，将 Z 中的每个示例投影 回原始空间，并在 X rec 中返回恢复的近似值。

完成 recoverData.m 中的代码后，ex7 pca.m 将恢复第一个示例的 ap 代理，您应该看到大约 [-1.047 -1.047] 的值。

*您现在应该提交您的解决方案。*

### 可视化投影

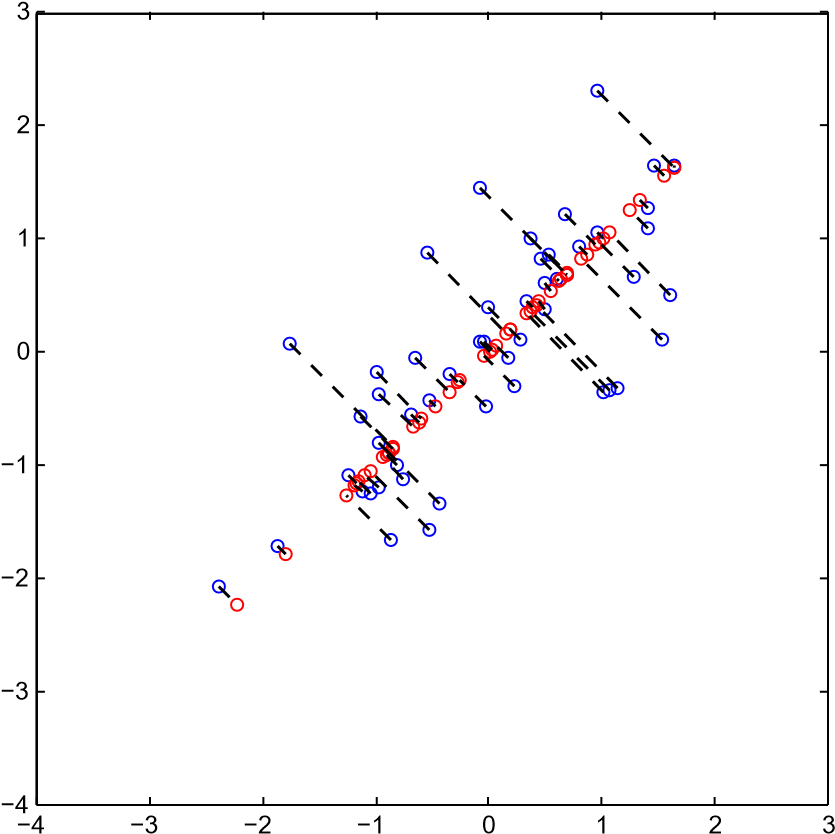


图 6：PCA 之后的归一化和投影数据。

完成 projectData 和 recoverData 后，ex7 pca.m 现在将执行投影和近似重建，以显示投影如何影响数据。在图6中，原始的data点用蓝色圆圈表示，而投影的数据点用红色圆圈表示。投影实际上只保留U 1给出的方向上的信息。

## 人脸图像数据集

在这部分练习中，您将在面部图像上运行PCA，看看如何在实践中使用它来减少尺寸。数据集ex7faces.mat包含一个面部图像数据集[[3]](#footnote-3)X，每个32×32个灰度图像。X 的每行 对应于一个人脸图像（长度为 1024 的行向量）。 ex7 pca.m 中的下一步 将加载并可视化这些人脸图像中的前 100 张（图 7）。



图 7：人脸数据集

### 面部 PCA

为了在人脸数据集上运行 PCA，我们首先通过从数据 matrix X 中减去每个特征的平均值来规范化数据集。脚本 ex7 pca.m 将为您执行此操作，然后运行您的 PCA 代码。运行 PCA 后，您将获得数据集的主成分。请注意，U 中的每个主成分 （每行）都是长度为 n 的向量 （对于人脸数据集，*n* = 1024）。事实证明，我们可以通过将每个主成分重塑为 与原始数据集中的像素相对应的32×32矩阵来可视化这些主成分。脚本 ex7 pca.m 显示描述最大变化的前 36 个主成分（图 8）。如果需要，还可以更改代码以显示更多主成分，以查看它们如何捕获越来越多的细节。

### 降维

现在， 您已经计算了人脸数据集的主成分，您可以使用它来减小人脸数据集的维度。这允许您使用输入大小较小的学习算法（例如，100 个维度），而不是原始的 1024 个维度。这有助于加快学习算法的速度。



图 8：人脸数据集上的主成分



图 9：人脸的原始图像和仅从前 100 个主成分重建的面部图像。

ex7 pca.m 中的下一部分将人脸数据集仅投影到前 100 个主成分上。具体而言，每个人脸图像现在由vector *z*（*i*）∈R100描述。

若要了解降维中丢失的内容，可以仅使用投影数据集恢复数据。在 ex7 pca.m 中，执行数据的近似恢复，并排显示原始和投影的人脸图像（图 9）。从重建中，您可以观察到面部的一般结构和外观被保留，而精细的细节却丢失了。这是数据集大小的显着减少（超过10×），可以帮助显着加快您的算法。例如，如果您正在训练神经网络来执行人名识别（gven a face image，预测 该人的识别），则可以使用仅 100 维的降维输入，而不是origi nal 像素。

## 可选（未评分）练习：用于可视化的 PCA

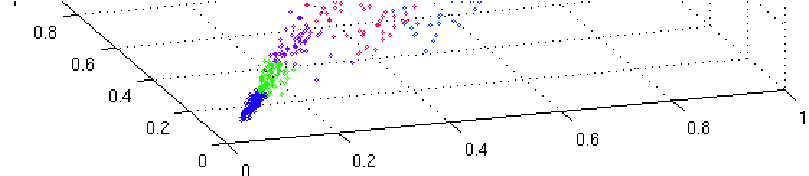
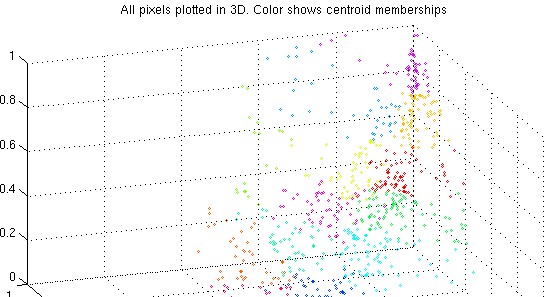


图 10：3D 格式的原始数据

在前面的 *K* 均值图像压缩练习中，您在三维 RGB 空间中使用了 K 均值算法。在 ex7 pca.m 脚本的最后一部分中，我们提供了代码，以使用 scatter3 函数可视化此 3D 空间中的最终像素分配 。每个数据点都根据其分配到的聚类进行着色。您可以在图形上拖动鼠标以旋转并以3维形式检查此数据。

事实证明，在3维或更大的维度中可视化数据集可能很麻烦。因此，通常希望仅以2D形式显示数据，即使以丢失某些信息为代价也是如此。在实践中，PCA通常用于降低数据的维度以进行可视化。在 ex7 pca.m 的下一部分中，脚本会将 PCA 的实现应用于 3 维数据，以将其减少到 2 维，并在 2D 散点图中可视化结果。可以将 PCA 投影视为一种旋转，它选择最大化数据散布的视图，该视图通常对应于“最佳”视图。

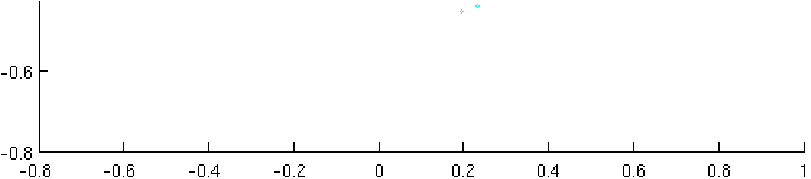
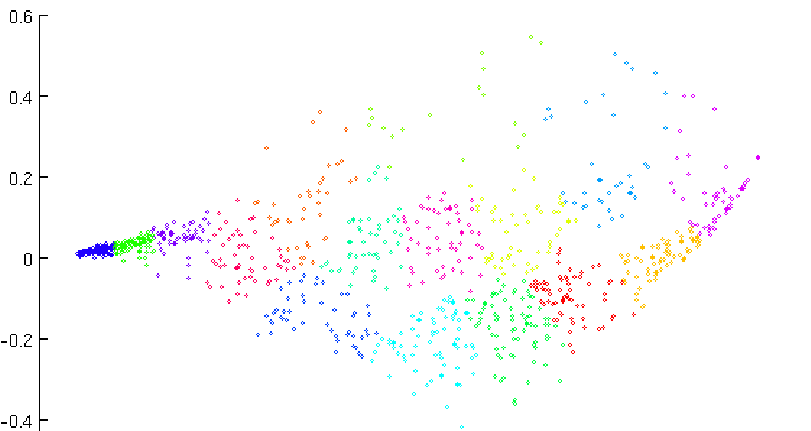


图 11：使用 PCA 生成的 2D 可视化

# 提交和评分

完成作业的各个部分后，请务必使用 提交 功能系统将您的解决方案提交到我们的服务器。以下是本练习每个部分的评分方式的细分。

|  |  |  |
| --- | --- | --- |
| **部分** | **提交的文件** | **点** |
| 查找最近的质心 | findClosestCentroids.m | 30 积分 |
| 计算质心均值 | computeCentroids.m | 30 积分 |
| 断续器 | 断续器 | 20 积分 |
| 项目数据 | projectData.m | 10 积分 |
| 恢复数据 | recoverData.m | 10 积分 |
| 总积分 |  | 100 积分 |

您可以多次提交解决方案，我们将仅考虑最高分。

1. Octave 是 MATLAB 的免费替代品。对于编程练习，您可以自由使用Octave或MATLAB。 [↑](#footnote-ref-1)
2. 本练习中使用的照片属于弗兰克沃特斯并在他的许可下使用。 [↑](#footnote-ref-2)
3. 此数据集基于[裁剪版本](http://itee.uq.edu.au/~conrad/lfwcrop/) 的[野外的标记面孔](http://vis-www.cs.umass.edu/lfw/) 数据。 [↑](#footnote-ref-3)