Part 2. 综合实训报告

基于 K-means 聚类算法的研究

20121034 胡才郁

(计算机工程与科学学院)

摘　要　K-means 聚类算法是一种广泛使用的聚类算法。本文对 K-means 聚类算法的原理、参数调优进行了分析，并使用 K-means 聚类算法，在二维数据集和图片上进行对比实验，验证了算法的有效性。通过数据可视化得到了明显的聚类效果，给出了具体的代码实现。

关键词　K-means，聚类算法，无监督学习

1 引言

无监督学习是机器学习的一个分支，它从未经标记，分类的测试数据中学习。对于无监督学习而言，聚类问题是其中的重要问题，而 K-means 算法为处理数据聚类问题的一种常用方法。本文分析了 K-means 聚类算法的原理，分析了具体参数对 K-means 聚类效果的影响。该算法综合考虑了边界区域数据样本的位置分布及其邻域数据点的类簇归属信息。

2 K-means 聚类算法

2.1 聚类

聚类是按照某个特定标准(如距离)把一个数据集分割成不同的类或簇，使得同一个簇内的数据对象的相似性尽可能大，同时不在同一个簇中的数据对象的差异性也尽可能地大。也即聚类后同一类的数据尽可能聚集到一起，不同类数据尽量分离。数据聚类为一种无监督式学习，即没有给定事先标记过的训练示例，自动对输入的资料进行分类或分群。

2.1 算法分析

K-means 是一种迭代的，无监督的聚类算法，将类似的实例组合成簇。该算法通过随机确定每个簇的初始聚类中心开始，然后重复将实例分配给最近的簇，并重新计算该簇的聚类中心。

首先，随机确定 k 个初始点的坐标作为簇的聚类中心的初始化坐标。然后将数据集中每个点分配到一个簇中，具体而言，为每个点寻找距离其最近的质心，并将其分配给该质心所对应的簇。这一步完成之后，每个簇的质心更新为该簇所有点的平均值。

如果用数据表达式表示，假设簇划分为$(C_1, C_2, \dots C_k)$，则我们的目标是最小化平方误差E：

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - \mu_i|_2^2$$

其中μ_i是簇C_i的均值向量，也称为质心，表达式为：

$$\mu_i = \frac{1}{|C_i|}\sum_{x \in C_i} x$$

直接求得上式的最小值并不容易，通过迭代更新的方法最小化平方误差E，此算法的迭代更新过程如下图所示：

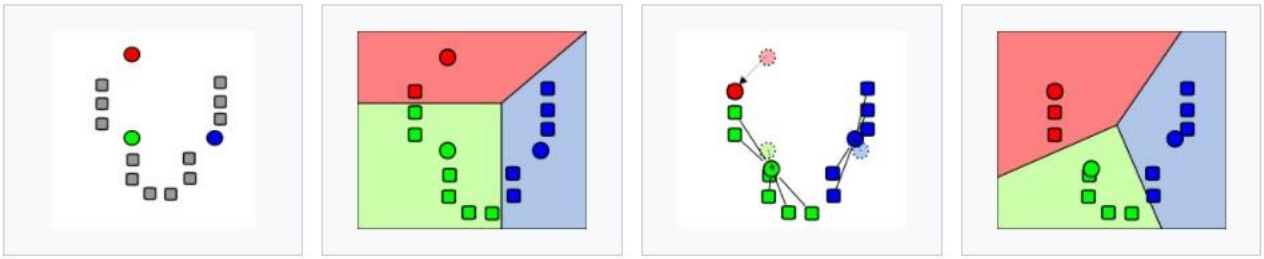

图 1. K-means 聚类算法图示

步骤一：k 个初始“均值”（此处 k=3）实在数据与内随机生成（以颜色区分）

步骤二：通过将每个观察值与最近的平均值相关联来创建 k 个集群。

步骤三：k 个簇中的每一个的质心成为新的均值

步骤四：重复步骤 2，3，直到达到收敛

图 1 中第一幅图片为了初始的数据集，此处假设 k=3，随机选择了 3 个类别的质心，即图中的红色质心、绿色质心和蓝色质心。如第二幅图片所示，分别求样本中所有点到这两个质心的距离，并标记每个样本的类别为和该样本距离最小的质心的类别。如第三幅图片所示，此时对当前标记为红色、绿色和蓝色的点，分别求其新的质心，质心位置发生变动，并进行迭代操作，最终得到四个类别。

K-means 算法采用了贪心策略，通过迭代优化来近似求解。算法流程如图 2 所示，其中第 1 行对均值向量进行了初始化，算法分为两个步骤，第一个 for 循环是赋值步骤，即对于每一个样本，计算其应该属于的类。第二个 for 循环是聚类中心的移动，即：对于每一个簇，重新计算该簇的质心。在第 3-7 行与第 8-15 行依次对当前簇划分及均值向量迭代更新，若迭代更行后聚类结果保持不变，则在最终将当前簇划分结果返回。

输入： 样本集$D = \{x_1, x_2, ..., x_m\}$

聚类簇数k

过程：

1: 从D中随机选择k个样本作为初始均值向量$\{\mu_1, \mu_2, ..., \mu_m\}$

2: **repeat**

3: **for** j = 1,2,3,...,m **do**

4: 计算样本x_j与各均值向量μ_i的距离：d_{ji}

5: 根据距离最近的均值向量确定x_j的簇标记：λ_i

6: 将样本x_i划入相应的簇：C_{λ_i}

7: **end for**

8: **for** i = 1,2,3,...,k **do**

9: 计算新均值向量：μ_i'

10: **if** $\mu_i' \neq \mu\_i$ then

```
11:          将当前均值向量μ_i更新为μ_i'
12:      else
13:          保持当前均值向量不变
14:      end if
15:   end for
16:   until 当前均值向量均未更新
输出：簇划分C = {C_1, C_2, ..., C_k}
```

图 2. K-means 聚类算法流程

3 K-means 聚类算法具体应用

3.1 方法实现以及代码分析

本文在研究过程中采用的是 Visual Studio Code 集成开发环境配置 Jupyter notebook，利用了 numpy 科学计算库、matplolib 绘图库等 API 接口，使用 python 语言实现。编写方法，手动实现了该算法。

表 1. 关于项目中各方法的作用的声明

序号	方法名	说明
1	init_centroids	初始化簇的质心坐标
2	find_closest_centroids	对于每个样本，寻找距离其最近的簇
3	compute_centroids	计算并更新簇的坐标
4	run_k_means	启动 K-means 算法

以下给出各个方法的具体实现过程：

步骤一：初始化聚类中心，在此处从样本数据集中选择选择一个随机样本，并将其坐标用作聚类中心的初始坐标。在此函数之中，X 为样本数据，存储在 2 维的 numpy 数组之中；k 为由用户确定的聚类中心的数目。通过 random 模块之中的 randint 函数，随机选取样本点的索引值。并利用数组切片的操作，直接对数据数组进行处理。

```
def init_centroids(X, k):
    m, n = X.shape
    centroids = np.zeros((k, n)) # 构造聚类中心为 k 行 n 列，数据值为 0 的数组
    idx = np.random.randint(0, m, k) # 随机选取样本值点的索引值
    for i in range(k):
        centroids[i,:] = X[idx[i],:] # 利用数组切片
    return centroids
```

图 3. init_centroids 方法定义

步骤二：为每一个样本寻找与其距离最近的簇。此函数接受数据集与簇坐标集通过比较每一个样本与每一个簇质心的位置，将该样本分配给距离最近的簇中。在此处使用的距离度量为欧式距离，两个样本点间的距离是两个样本点相似程度的反映。此函数针对数据值之中的全部样本点，返回所有样本点所对应最近簇的坐标。

```
def find_closest_centroids(X, centroids): # 对于每个样本，寻找距离其最近的簇
    m = X.shape[0]
    k = centroids.shape[0]
    idx = np.zeros(m)
    for i in range(m):
        min_dist = 1000000  # 设置最小距离默认值
        for j in range(k):
            dist = np.sum((X[i,:] - centroids[j,:]) ** 2) # 计算欧式距离
            if dist < min_dist:
                min_dist = dist
                idx[i] = j
    return idx # 返回 size 为 m 的一维数组
```

图 4. find_closest_centroids 方法定义

步骤三：计算并更新簇的质心坐标，在此过程中，X 为 2 为 Numpy 数组，k 为簇的数量。

```
def compute_centroids(X, idx, k):# X 为 2 维数组，k 为簇的数量
    n = X.shape[1]
    centroids = np.zeros((k, n))
    for i in range(k):
        indices = np.where(idx == i) # 返回满足条件的数组索引
        centroids[i,:] = (np.sum(X[indices,:], axis=1)/
                          len(indices[0])).ravel() # 更新簇的质心坐标
     return centroids
```

图 5. compute_centroids 方法定义

步骤四：启动 K-means 算法，将前三个步骤封装整合，依次执行，读取样本数据 X，给定簇初始坐标，最大迭代次数。

```
def run_k_means(X, initial_centroids, max_iters):
    m, n = X.shape
    k = initial_centroids.shape[0]
    idx = np.zeros(m)
    centroids = initial_centroids
    for i in range(max_iters): # 进行迭代
        idx = find_closest_centroids(X, centroids)
        centroids = compute_centroids(X, idx, k)
    return idx, centroids
```

图 6. run_k_means 方法定义

3.2 针对二维数据集的应用

3.2.1 数据预处理

将 K-means 聚类算法实施和应用到一个简单的二维数据集，以获得一些直观的工作原理。本文选取了 300 个二维数据点，即(300 * 2)的二维数据集。利用 python 语言的 Pandas 库中的相关函数，将其封装为 DataFrame 对象，使用 K-means 算法进行聚类处理。如图

	X1	X2
0	1.842080	4.607572
1	5.658583	4.799964
2	6.352579	3.290854
3	2.904017	4.612204
4	3.231979	4.939894
...	...	...
295	7.302787	3.380160
296	6.991984	2.987067
297	4.825534	2.779617
298	6.117681	2.854757
299	0.940489	5.715568

300 rows × 2 columns

图 7. 封装后的二维数据集

3.2.2 数据可视化

人工选择聚类数 k 值为 3，在对二维数据集进行处理后，使用 matplotlib 提供的绘图方法，对数据进行可视化处理。观察数据处理前后的可视化结果可知，在选择簇的数量 k 为 3 时，二维数据较好的聚为 3 类。K-means 算法在选择 k 值合理，迭代次数足够的情况下，对于二维数据的拟合效果较好。

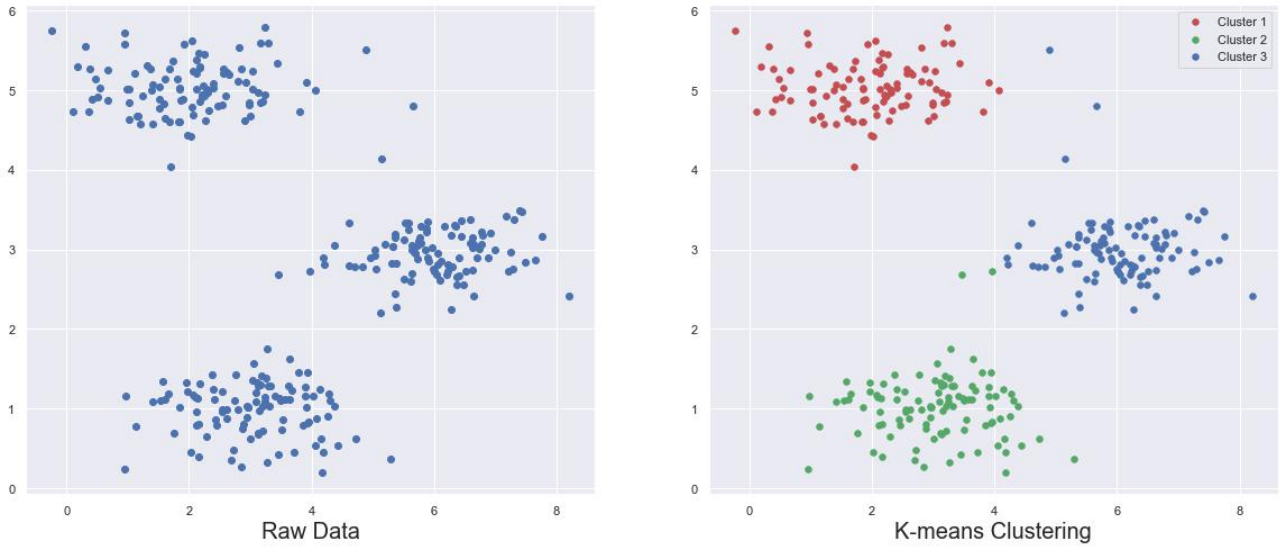

图 8. 二维数据集的聚类结果

3.3 k值的选取

K 值的选择会对 K-means 聚类算法的结果产生重大影响。

如果选择的 k 值较小，就相当于用较小的邻域中的训练实例进行预测，“学习”的近似误差会减小，只有与输入实例较近的（相似的）训练实例才会对预测结果起作用。但缺点是“学习”的估计误差会增大，预测结果会对近邻的实例点非常敏感。如果临近的实例点恰巧是噪声，预测就会出错。换句话说，k 值的减小就意味着整体模型变得复杂，容易发生过拟合。

如果选择较大的 k 值，就相当于用较大邻域的训练实例进行预测。其优点是可以减少学习的估计误差，但缺点是学习的近似误差会增大。这时与输入实例较远（不相似的）训练实例也会对预测起作用，使预测发生错误。K 值的增大就意味着整体的模型变得简单。

3.4 针对图像的处理

由于 K-means 算法只能处理数值型数据，因此使用了 skimage 库中 io 模块的 imread 的方法，此方法可以将图片转换为 numpy 数组，图像数据是以 RGB 的格式进行存储的，通道值默认范围 0-255。通过对 RGB 值进行 K-means 聚类处理，取得每一个像素点对应的 RGB 数据值，分配这个值所属于的簇。处理后的数据使用 io 模块的 imshow 方法重新绘制图像，观察处理前后图像差别，可直观感受到该算法的、原理与 k 值选择对聚类效果的影响。

```
data = io.imread('data\bird.png') / 255. # data 为 numpy 数组
plt.imshow(data)
```

图 9. 图像预处理

之后进行的操作类似于之前二维数据集的操作，在此部分使用 sklearn 库提供的 API 进行数据的处理，调整参数，分别选取簇数量 k 为 15，5，3 对数据进行 K-means 聚类，将处理后的数据进行绘制子图。部分关键代码如下：

```
from sklearn.cluster import KMeans # 导入 kmeans 库
model = KMeans(n_clusters= 15 / 5 / 3, n_init=100) # 训练模型
model.fit(data) # 拟合数据
centroids = model.cluster_centers_ # 设置聚类中心
C = model.predict(data) # 预测中心位置
compressed_pic = centroids[C].reshape() # 得到处理后的图片的数据
# 以下为绘图部分代码
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 4, figsize = (20 , 8)) # 设置图片布局
ax.imshow(pic) # 绘图
```

图 10. 模型训练部分关键代码

将鸟的图片数据信息转换 RGB 数组，分别选取簇数量 k 为 15，5，3，进行 K-means 聚类处理，得到图像处理结果如下图：

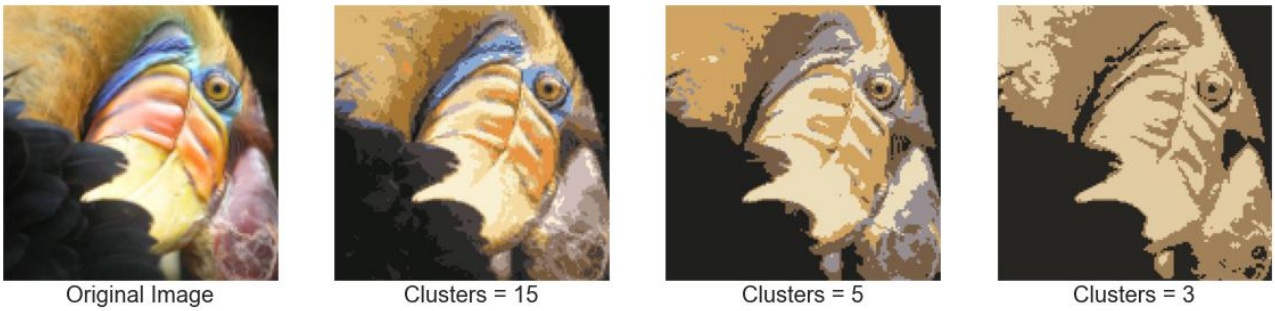

图 11. 簇数量不同时鸟的颜色变化

观察实验结果可以发现，在簇数量 k 值为 15 时，图像已经丢失了部分细节，存在图像的主要特征仍然存在。当簇数量 k 值为 3，鸟的基本轮廓存在。但由于身体部分旁边存在棕色噪声点，身体被“染”为棕色。

为了进一步理解不同的 k 值对于 K-means 效果的影响，选取了 RGB 值差异较大，颜色差异较明显的图片。此处选取了不同颜色的上海大学校徽进行实验。

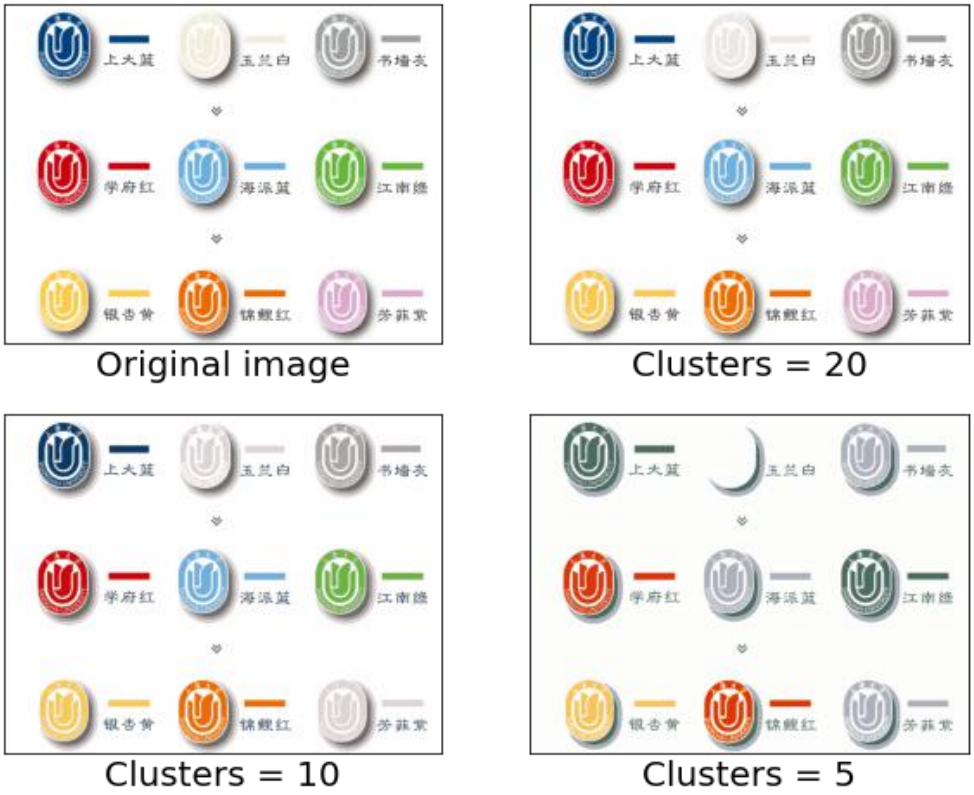

图 12. 簇数量不同时校徽的颜色变化

对于这一张图片而言，校徽共有为 9 种颜色。观察结果可知，当 k 值取 20 时，图像颜色细节丢失较少，效果较好，但“玉兰白”颜色变暗，当 k 值取 10 时，“芳菲紫“褪色，当 k 值取 5 时，“芳菲紫”，“海派蓝”均被染成“书墙灰”色，聚为同一类，对应 k 值较小时，临近的实例点恰巧是噪声图片的底色的噪声，“玉兰白”被染为图片的底色白色，“玉兰白”与图片底色聚为一类，原先为橙红色的“锦鲤红”被染为“学府红”的大红色，聚为一类。

4 结语

通过本次课程的学习，我迈出了机器学习学习的第一步，我对于机器学习领域有了初步的了解。直接调用相应库所提供的 API，通过在这种类似于“黑盒”的环境下进行开发，使我们不用过度注重于数据处理过程中的细节。而动手用代码实现算法原理，更是我对部分机器学习算法有更加深入的认识。

致谢 在本次实训之中，大部分的课程实验内容都是在数学推理与代码实现之中完成的。同时，此外，也需要感谢斯坦福大学吴恩达教授，我主要通过他在 Coursea 平台上的网课进行自学，并完成相关作业；以及南京大学周志华老师所著的《机器学习》，也就是俗称的“西瓜书”。在我对算法无法理解时，为本次项目的完成提供了很多帮助。正是互联网的开源共享精神，有丰富的学习资料可供参考。最后，也十分感

谢在百忙之中抽出时间审阅本文的老师。由于本人的学识和写作的水平有限，在本论文的写作中难免有僻陋，恳请老师多指教。

参 考 文 献

[1] 周志华.机器学习 [M]. 北京：清华大学出版社，2015.

[2] 李航.统计学习方法 [M]. 北京：清华大学出版社，2012.

[3] 机器学习实战[M]. 北京：人民邮电出版社，2013.

[4] 维基百科 https://en.wikipedia.org/wiki/K-means_clustering