机器学习实验 10. 聚类（一）K-means 聚类

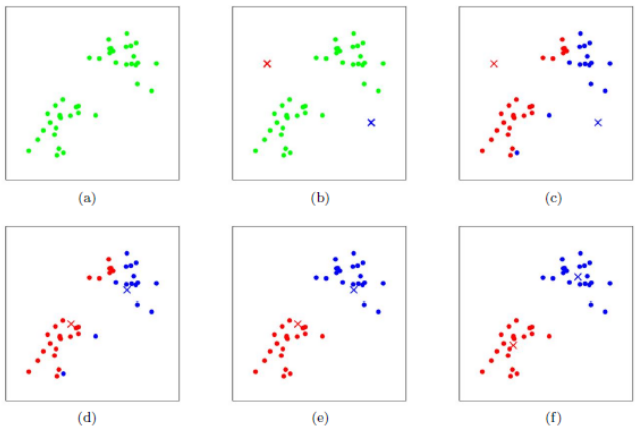
K-means 聚类原理概述

K-means 算法是最经典的无监督学习方法。在 K-means 聚类过程中，首先（随机）选择 K 个类簇中心（表示 K 个不同的类），根据每个待测样本到类簇中心的距离进行类别划分，即将待测样本 x_i 归类于最近类簇中心所表示的类 C_k。然后，对类簇中心进行重新计算（更新）：对应类中所有数据对象的均值，即为更新后该类簇的类簇中心。定义第 k 个类簇的类簇中心为 $Center_k$，则类簇中心更新方式如下：

$$Center_k = \frac{1}{|C_k|}\sum_{x_i \in C_k} x_i \tag{1}$$

其中，C_k 表示第 k 个类簇，$|C_k|$ 表示第 k 个类簇中数据对象的个数，这里的求和是指类簇 C_k 中所有元素在特征向量上的和，因此 $Center_k$ 也是一个含有 D 维特征的向量，表示为 $Center_k=(Center_{k,1},Center_{k,2},...,Center_{k,D})$。依据上述步骤，交替迭代不断划分类簇，并更新类簇中心，直至收敛。

k-means 算法聚类过程示意图，如下：



其中，红色和蓝色的×代表两个类簇中心，红色和蓝色的圆点分别对应每次迭代中依据更新的两个类簇中心得到的类别划分。

K-means 聚类算法描述

1）初始化 K 个类簇中心;

2）计算各个数据对象到聚类中心的距离，把数据对象划分至距离其最近的聚类中心所在类簇中;

3）根据所得类簇，更新类簇中心;

4）继续计算各个数据对象到聚类中心的距离，把数据对象划分至距离其最近的聚类中心所在类簇中;

5）根据所得类簇，继续更新类簇中心;……一直迭代，直到达到最大迭代次数 T，或者两次迭代 J 的差值小于某一阈值时，迭代终止，得到最终聚类结果。

K-means 聚类中的问题

（1）度量距离如何选取?

通常采用欧氏距离来计算数据对象间的距离:

$$dist(x_i, x_j) = \sqrt{\sum_{d=1}^{D} (x_{i,d} - x_{j,d})^2} \tag{2}$$

其中，D 表示数据对象的属性个数。也可以采用其他距离，例如曼哈顿距离等。

（2）迭代终止条件如何设置?

一般情况，有两种方法来终止迭代：一种方法是设定迭代次数 T，当到达第 T 次迭代，则终止迭代，此时所得类簇即为最终聚类结果；另一种方法是采用误差平方和准则函数，函数模型如下:

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} dist(x_i, Center_k) \tag{3}$$

其中，K 表示类簇个数。当两次迭代 J 的差值小于某一阈值时，即$\Delta J < \delta$时，则终止迭代，此时所得类簇即为最终聚类结果。

（3）如何确定类别数目 K?

Kmeans 算法需要实现确定类别的数目 K，也就是初始化的聚类中心数。但在现实中，这个问题很难以确定。这也是大多数聚类方法所面临的一个共同问题。

许多预处理技术尝试事先确定类别数目 K,但更常用的方法是事先选取小部分数据进行交叉检测来确定 K 值。

（4）如何选取初始点?

K-means 算法会产生局部最优解，即如果聚类中心选择不准确，有可能无法迭代到最优聚类的情况就收敛了。可以采用如下方法选择聚类中心。

1，随机选一个聚类中心 a，然后统计所有的样本点到该中心的距离。

2，令距离 a 越远的样本点被选中的概率越大，随机选取第二个聚类中心 b。

3，再计算所有样本点到这两个聚类中心的距离（可计算所有样本点到最近的那个聚类中心的距离），距离越远的选中概率越大，从而随机选取第三个聚类中心 c。。

3，重复上面的步骤，直到选出 K 个聚类中心。

K-means 聚类算法的改进

（1）K-中值聚类

在更新聚类中心的过程中采用当前类别样本的中值而非均值,可以避免数据分布和噪声带来的影响，具有一定聚类鲁棒性。

（2）Fuzzy Kmeans

采用模糊数学理论，计算每个样本的模糊类别，使得同一个样本可以同时处于多个类，以适应多类问题。

K-means 聚类的优缺点

优点：

是解决聚类问题的一种经典算法，简单、快速;

对处理大数据集，该算法保持可伸缩性和高效率;

当簇近似为高斯分布时，它的效果较好。

缺点：

在簇的平均值可被定义的情况下才能使用，可能不适用于某些应用；

必须事先给出 k(要生成的簇的数目)，而且对初值敏感，对于不同的初始值，可能会导致不同结果；

不适合于发现非凸形状的簇或者大小差别很大的簇；

对躁声和孤立点数据敏感；

可作为其他聚类方法的基础算法，如谱聚类。

K-means 聚类算法在模拟数据中的一般实现

1) 我们先创建一个新的 Python 文件，然后导入下面的程序包：

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.cluster import KMeans
import utilities
```

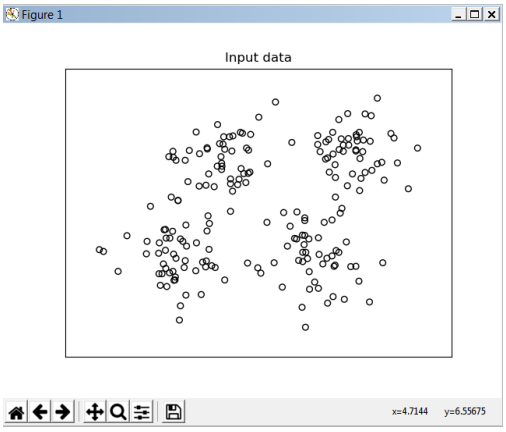
2) 加载输入数据，然后定义集群的数量。我们将使用 data_multivar.txt 数据文件：

```
data = utilities.load_data('data_multivar.txt')
num_clusters = 4
```

3) 我们需要看看输入数据是什么样子的。继续向 Python 文件中加入下面的代码：

```
plt.figure()
plt.scatter(data[:,0], data[:,1], marker='o',
        facecolors='none', edgecolors='k', s=30)
x_min, x_max = min(data[:, 0]) - 1, max(data[:, 0]) + 1
y_min, y_max = min(data[:, 1]) - 1, max(data[:, 1]) + 1
plt.title('Input data')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
```

运行代码，可以看到如下图所示的图形。



4）现在可以训练模型了。先初始化一个 k-means 对象，然后训练它：

```
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)
kmeans.fit(data)
```

5）数据训练之后，我们需要可视化边界。继续向 Python 文件中加入下面的代码：

```
# Step size of the mesh
step_size = 0.01

# Plot the boundaries
x_min, x_max = min(data[:, 0]) - 1, max(data[:, 0]) + 1
y_min, y_max = min(data[:, 1]) - 1, max(data[:, 1]) + 1
x_values, y_values = np.meshgrid(np.arange(x_min, x_max, step_size), np.arange(y_min, y_max, step_size))

# Predict labels for all points in the mesh
predicted_labels = kmeans.predict(np.c_[x_values.ravel(), y_values.ravel()])
```

6）我们已经通过网格数据评估了模型。接下来把这些结果都画出来，看看边界 线的布局：

```
# Plot the results
predicted_labels = predicted_labels.reshape(x_values.shape)
plt.figure()
plt.clf()
plt.imshow(predicted_labels, interpolation='nearest',
           extent=(x_values.min(), x_values.max(), y_values.min(), y_values.max()),
           cmap=plt.cm.Paired,
           aspect='auto', origin='lower')

plt.scatter(data[:,0], data[:,1], marker='o',
        facecolors='none', edgecolors='k', s=30)
```

7）把中心点画在图形上：

```
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:,0], centroids[:,1], marker='o', s=200, linewidths=3,
        color='k', zorder=10, facecolors='black')
x_min, x_max = min(data[:, 0]) - 1, max(data[:, 0]) + 1
y_min, y_max = min(data[:, 1]) - 1, max(data[:, 1]) + 1
plt.title('Centoids and boundaries obtained using KMeans')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

运行代码，可以看到如下图所示的图形。