一、 K-means 算法简介

k-平均算法（英文：*k*-means clustering）源于信号处理中的一种向量量化方法，现在则更多地作为一种聚类分析方法流行于数据挖掘领域。*k*-平均聚类的目的是：把 n 个点（可以是样本的一次观察或一个实例）划分到 *k* 个聚类中，使得每个点都属于离他最近的均值（此即聚类中心）对应的聚类，以之作为聚类的标准。这个问题将归结为一个把数据空间划分为 Voronoi cells 的问题。

这个问题在计算上是 NP 困难的，不过存在高效的启发式算法。一般情况下，都使用效率比较高的启发式算法，它们能够快速收敛于一个局部最优解。这些算法通常类似于通过迭代优化方法处理高斯混合分布的最大期望算法（EM 算法）。而且，它们都使用聚类中心来为数据建模；然而 *k*-平均聚类倾向于在可比较的空间范围内寻找聚类，期望-最大化技术却允许聚类有不同的形状。

二、 K-means 算法和 KNN 算法的区别

- K-Means 是无监督学习的聚类算法，没有样本输出；而 KNN 是监督学习的分类算法，有对应的类别输出。
- KNN 基本不需要训练，对测试集里面的点，只需要找到在训练集中最近的 k 个点，用这最近的 k 个点的类别来决定测试点的类别。而 K-Means 则有明显的训练过程，找到 k 个类别的最佳质心，从而决定样本的簇类别。
- 两个算法都包含一个过程，即找出和某一个点最近的点。
- 两者都利用了最近邻(nearest neighbors)的思想。

三、 K-means 算法原理

K-means 算法的思想是对于给定的样本集，按照样本之间的距离大小，将样本集划分为 K 个簇。让簇内的点尽量紧密的连在一起，而让簇间的距离尽量的大。关于 K-means 算法，有一个比较出名的解释，就是牧师-村民模型：

- 有四个牧师去郊区布道，一开始牧师们随意选了几个布道点，并且把这几个布道点的情况公告给了郊区所有的居民，于是每个居民到离自己家最近的布道点去听课。
- 听课之后，大家觉得距离太远了，于是每个牧师统计了一下自己的课上所有的居民的地址，搬到了所有地址的中心地带，并且在海报上更新了自己的布道点的位置。
- 牧师每一次移动不可能离所有人都更近，有的人发现 A 牧师移动以后自己还不如去 B 牧师处听课更近，于是每个居民又去了离自己最近的布道点……
- 就这样，牧师每个礼拜更新自己的位置，居民根据自己的情况选择布道点，最终稳定了下来。

这就是一个 K-means 算法的过程，这个过程可以抽象化描述为：

假设簇划分为$\{C_1, C_2, \dots, C_k\}$，我们的目标是优化公式

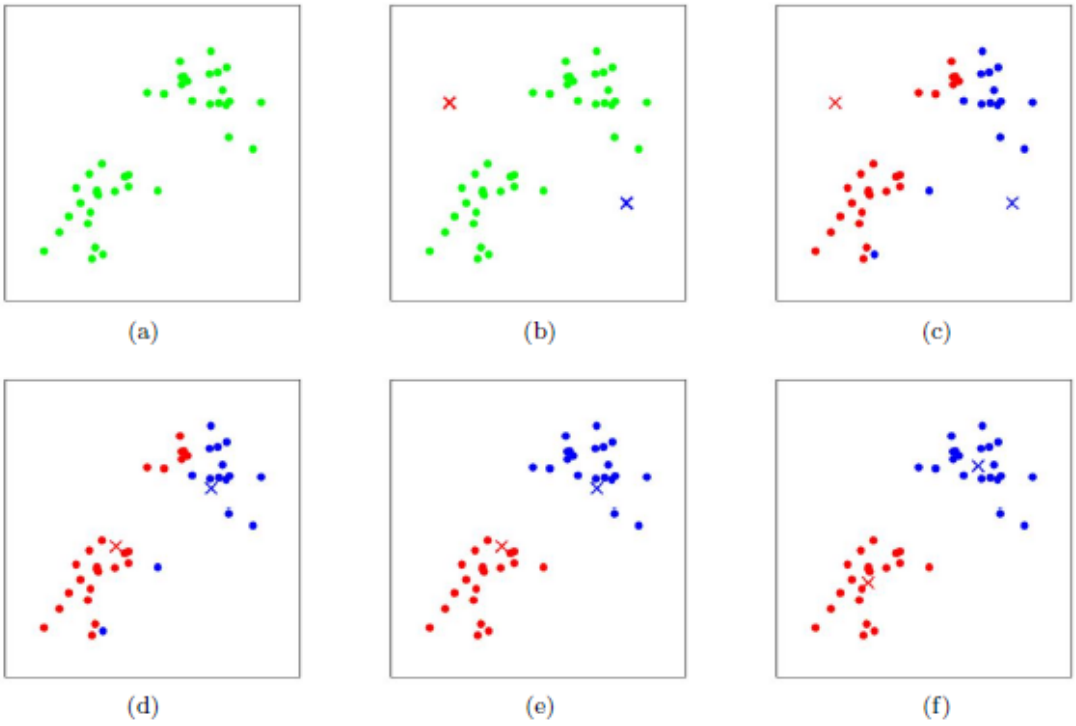
$$E = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \|_2^2$$

其中，μ_i是簇C_i的均值向量，也就是质心。可以表达为

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

开始已经讲过，求这个式子的最小值是一个 NP 难的问题，因此要用启

发式迭代方法解决。



上图 a 表达了初始的数据集，假设 k=2。在图 b 中，我们随机选择了两个 k 类所对应的类别质心，即图中的红色质心和蓝色质心，然后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，如图 c 所示，经过计算样本和红色质心和蓝色质心的距离，我们得到了所有样本点的第一轮迭代后的类别。此时我们对我们当前标记为红色和蓝色的点分别求其新的质心，如图 4 所示，新的红色质心和蓝色质心的位置已经发生了变动。图 e 和图 f 重复了我们在图 c 和图 d 的过程，即将所有点的类别标记为距离最近的质心的类别并求新的质心。最终我们得到的两个类别如图 f。

当然在实际 K-Mean 算法中，我们一般会多次运行图 c 和图 d，才能达到最终的比较优的类别。

四、 K-means 算法流程

输入：样本集$D = \{x_1, x_2, \dots, x_m\}$,聚类的簇数 k，最大迭代次数 N

输出：簇划分$C = \{C_1, C_2, \dots, C_k\}$,

1. 从数据集 D 中随机选择 k 个样本作为初始的 k 个质心向量 :$\{\mu_1, \mu_2, \dots, \mu_k\}$
2. 对于 n=1,2,...,N
   a) 将簇划分 C 初始化为$C_t = \emptyset$，$t = 1,2, \dots, k$
   b) 对于 i=1,2...m,计算样本x_i和各个质心向量$\mu_j (j = 1,2, \dots, k)$的距离：

$$d_{ij} = \|x_i - \mu_j\|_2^2$$

   将x_i标记为最小的d_{ij}对应的类别λ_i，此时更新$C_{\lambda i} = C_{\lambda i} \cup \{x_i\}$

   c) 对于$j = 1,2, \dots, k$,对C_j中所有样本点重新计算新的质心

$$\mu_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$$

   d) 如果所有的 k 个质心向量都没有发生变化，就转到步骤 3
3. 输出簇划分$C = \{C_1, C_2, \dots, C_k\}$

因此，可以总结出 K-means 算法的流程如下：

1. 从数据中选择 k 个对象作为初始聚类中心;
2. 计算每个聚类对象到聚类中心的距离来进行划分；
3. 再次计算每个聚类中心
4. 计算标准测度函数，直到达到最大迭代次数，则停止；否则，继续操作。

五、 初始质心的选取

常见的方法是随机的选取初始质心，但是这样簇的质量常常很差。为此，常见的解决方法有下面三种：

- 多次运行，每次使用一组不同的随机初始质心，然后选取具有最小 SSE（误差的平方和）的簇集。这种策略简单，但是效果可能不好，这取决于数据集和寻找的簇的个数。

- 取一个样本，并使用层次聚类技术对它聚类。从层次聚类中提取 K 个簇，并用这些簇的质心作为初始质心。该方法通常很有效，但仅对下列情况有效：样本相对较小；K 相对于样本大小较小。

- 取所有点的质心作为第一个点。然后，对于每个后继初始质心，选择离已经选取过的初始质心最远的点。使用这种方法，确保了选择的初始质心不仅是随机的，而且是散开的。但是，这种方法可能选中离群点。

六、 K-means 算法优缺点

优点

- 原理简单，实现容易，收敛速度快。
- 聚类效果好。
- 可解释性好。

- 需要调参的参数少，只有簇数 k。

缺点

- K 值选取不好把握
- 不是凸的数据集较难收敛
- 如果各隐含类别的数据不平衡，聚类效果不佳。
- 采用迭代方法，得到的结果只是局部最优。
- 对噪音和异常点较为敏感。