问题

k-means算法可以说是机器学习中大家最耳熟能详也是最基础的聚类算法，面试中也常常被问起，由浅至深，没有扎实的理论基础是很难过得了这一关的，因此，我们必须得梳理梳理并且深入理解它。

k-means算法简介

● k-means是一种聚类算法。所谓的聚类，就是指在不知道任何样本的标签的情况下，通过数据之间的内在关系将样本分成若干个类别，使得相同类别样本之间的相似度高，不同类别之间的样本相似度低。因此，k-means算法属于非监督学习的范畴。

● k 是指 k 个簇（cluster），means 是指每个簇内的样本均值，也就是聚类中心。

● 基本思想：通过迭代的方式寻找 k 个簇的划分方案，使得聚类结果对应的代价函数最小。代价函数可以定义为各个样本距离它所属的簇的中心点的误差平方和：

$$J(c,\mu)=\sum_{i=1}^{N}||x_i-\mu_{c_i}||^2 \tag{1}$$

其中，x_i代表第i个样本，c_i是x_i所属的簇，μ_{c_i}代表簇对应的中心点（即均值），N是样本总数.

k-means算法流程

k-means算法采用了**贪心策略**，通过多次迭代来近似求解上面的代价函数，具体算法流程如下：

1. 随机选取 k 个点作为初始聚类（簇）中心，记为 $\mu_1^{(0)},\mu_2^{(0)},\ldots,\mu_k^{(0)}$。
2. 计算每个样本 x_i 到各个簇中心的距离，将其分配到与它距离最近的簇。

$$c_i^{(t)}\leftarrow \underset{k'}{argmin}||x_i-\mu_{k'}^{(t)}||^2 \tag{2}$$

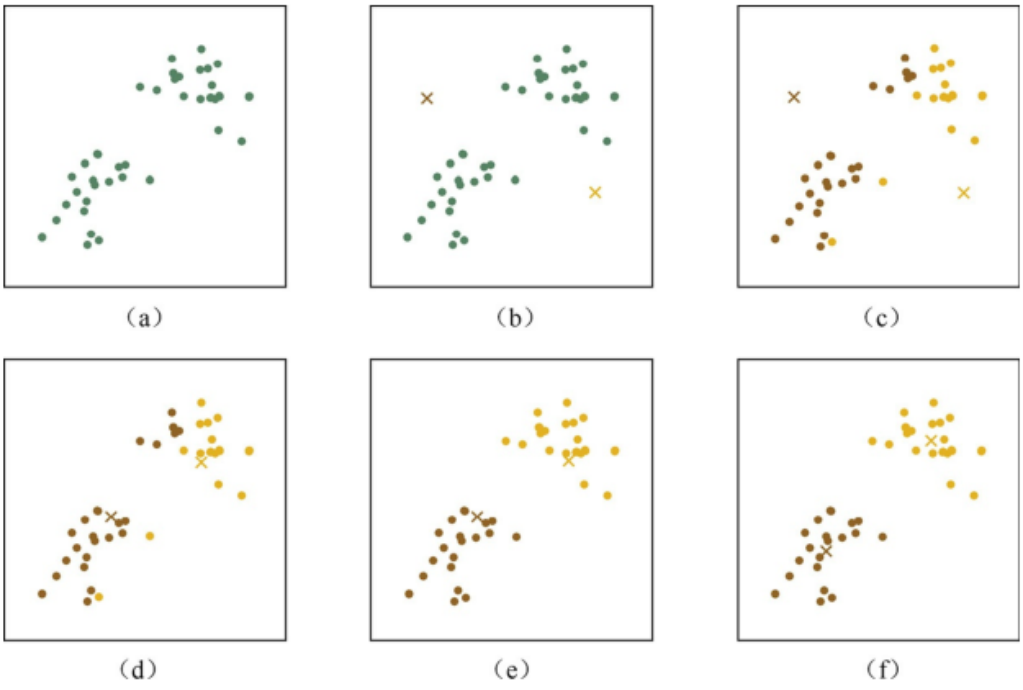
其中,t为当前迭代步数，k'为第k'个簇（类别）$(k'=1,2,..,k)$

3. 对于每个簇，利用该簇中的样本重新计算该簇的中心（即均值向量）：

$$\mu_k^{(t+1)}\leftarrow \underset{\mu}{argmin}\sum_{i:c_i^{(t)}=k'}||x_i-\mu||^2 \tag{3}$$

4. 重复迭代上面2-3步骤 T 次，若聚类结果保持不变，则结束。

k-means算法迭代示意图如下：



K均值聚类算法的迭代过程示意图

k-means算法优缺点

优点：

① 算法简单易实现。

② 对于大数据集，这种算法相对可伸缩且是高效的，计算复杂度为 $O(TNk)$ 接近于线性（其中T是迭代次数、N是样本总数、k为聚类簇数）。

③ 虽然以局部最优结束，但一般情况下达到的局部最优已经可以满足聚类的需求。

缺点：

① 需要人工预先确定初始K值，该值与实际的类别数可能不吻合。

② **K均值只能收敛到局部最优。**因为求解这个代价函数是个NP问题，采用的是贪心策略，所以只能通过多次迭代收敛到局部最优，而不是全局最优。

③ **K均值的效果受初始值和离群点的影响大。**因为 k 均值本质上是基于距离度量来划分的，均值和方差大的维度将对数据的聚类结果产生决定性的影响，因此需要进行归一化处理；此外，离群点或噪声对均值会产生影响，导致中心偏移，因此需要进行预处理。

④ 对于数据簇分布差别较大的情况聚类效果很差。例如一个类别的样本数是另一类的100倍。

⑤ 样本只能被划分到一个单一的类别中。

> 针对k-means上述缺点，有什么方法改进吗？当然是有的。

k-means++算法

由于 k-means 算法中，初始K值是人为地凭借经验来设置的，聚类的结果可能不符合实际的情况。因此，K值的设置对算法的效果影响其实是很大的，那么，如何设置这个K值才能取得更好的效果呢？

k-means++算法的主要思想：

- 首先随机选取1个初始聚类中心（n=1）。
- 假设已经选取了n个初始聚类中心（0<n<k），那么在选择第n+1个聚类中心时，距离当前已有的n个聚类中心越远的点越有可能被选取为第n+1个聚类中心。
- 可以这样理解上面的思想：聚类中心自然是越互相远离越好，即不同的类别尽可能地分开。

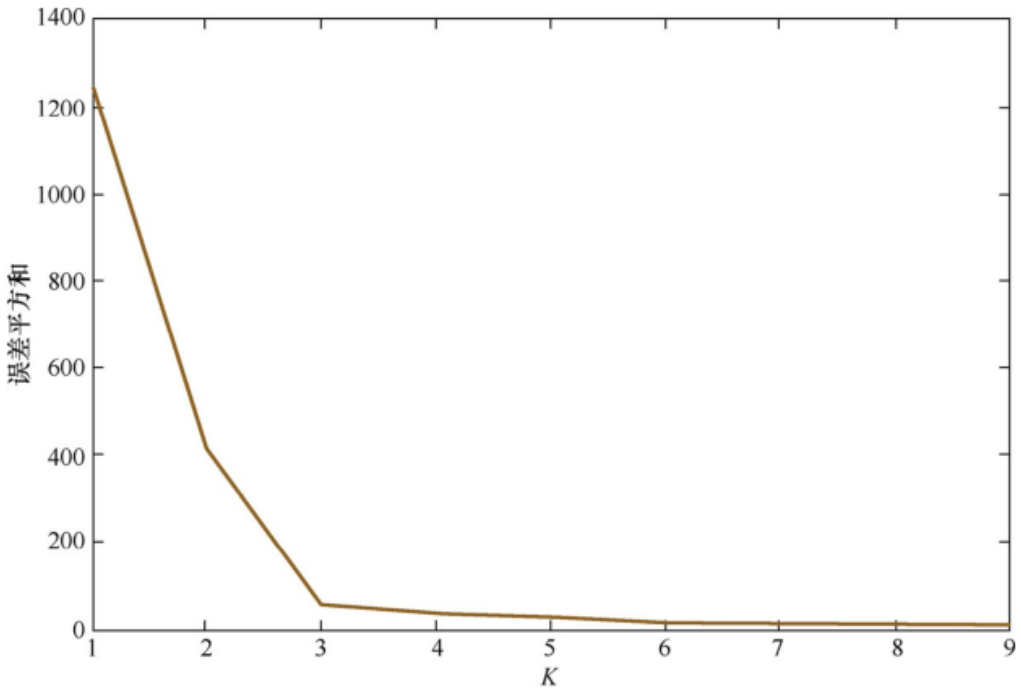
> 面试中可能会被问到这样的问题：k-means算法中的k值是怎么设置的？是一次初始k个值还是一个一个地初始直到k个？有了上面两种算法的对比，这个问题自然就迎刃而解了。
>
> 然而，k-means++算法虽然可以更好地初始k个聚类中心，但还是不能解决一个问题，k 值应该取多少才算合理？

如何选取k值？

如何才能合理地选取k值是k-means算法最大的问题之一，一般可以采取手肘法和`Gap Statistic`方法。

手肘法

k值的选择一般基于经验或者多次实验来确定，手肘法便是如此，其主要思想是：通过多次实验分别选取不同的k值，将不同k值的聚类结果对应的最小代价画成折线图，将曲线趋于平稳前的拐点作为最佳k值。如下图所示：



K均值算法中K值的选取：手肘法

> 上图中，k取值在1~3时，曲线急速下降；当k>3时，曲线趋于平稳。因此，在k=3处被视为拐点，所以手肘法认为最佳的k值就是3。
>
> 然而，实际中很多时候曲线并非如同上图一样拐点处那么明显，因此单靠肉眼去分辨是很容易出错的。于是，就又有了一个改进的方法，可以不需要靠肉眼与分辨拐点，而是寻找某个最大值Gap(k)，具体如下。

Gap Statistic 方法

Gap Statistics 定义为：

$$Gap(k) = E(logD_k) - logD_k \tag{4}$$

其中，D_k是第k簇聚类对应的损失值，$E(logD_k)$是$logD_k$的期望。

对于上式的 $E(logD_k)$，一般通过蒙特卡洛模拟产生。具体操作是：在样本所在的区域内，按照均匀分布随机产生和原样本数目一样的随机样本，计算这些随机样本的均值，得到一个 D_k，重复多次即可计算出 $E(logD_k)$ 的近似值。

$Gap(k)$ 可以看做是随机样本的损失与实际样本的损失之差，假设实际样本最佳的簇类数目为 k，那么实际样本的损失应该相对较小，随机样本的损失与实际样本的损失的差值相应地达到最大，即**最大的 $Gap(k)$ 值应该对应最佳的k值。**

因此，我们只需要用不同的k值进行多次实验，找出使得$Gap(k)$最大的k即可。

> 到现在为止我们可以发现，上面的算法中，k值都是通过人为地凭借经验或者多次实验事先确定下来了的，但是当我们遇到高维度、海量的数据集时，可能就很难估计出准确的k值。那么，有没有办法可以帮助我们自动地确定k值呢？有的，下面来看看另一个算法。

ISODATA算法

ISODATA，全称是迭代自组织数据分析法，这种方法是针对传统 k-means 算法需要人为地预先确定k值的问题而改进的，其主要的思想是：

- 当某个类别样本数目过多、分散程度较大时，将该类别分为两个子类别。（分裂操作，即增加聚类中心数）
- 当属于某个类别的样本数目过少时，把该类别去除掉。（合并操作，即减少聚类中心数）

算法优点： 可以自动寻找合适的k值。

算法缺点： 除了要设置一个参考聚类数量 k_0 外，还需要指定额外的3个阈值，来约束上述的分裂和合并操作。具体如下：

1. 预期的聚类数目 k_0 作为参考值，最终的结果在 k_0 的一半到两倍之间。
2. 每个类的最少样本数目 N_{min}，若分裂后样本数目会少于该值，则该簇不会分裂。
3. 最大方差 $Sigma$，用于控制某个簇的样本分散程度，操作该值且满足条件2，则分裂成两个簇。
4. 两个簇最小距离 D_{min}，若两个簇距离小于该值，则合并成一个簇。