机器学习第五次作业
——k 均值聚类

2052902 韩意

一、 问题描述

给定$k=2$，试设计一个能自动确定初始聚类中心的改进k均值算法，要求聚类过程收敛时改进算法迭代的次数不大于随机选择初始聚类中心时的情形。在西瓜数据集 4.0 上编程实现并展示改进效果。

二、 算法实现

本题使用 Python 实现，用到的库包括 NumPy、Matplotlib、time，其中 time 库用于比较算法的运行速度。

k均值聚类属于启发式方法，不能保证收敛到全局最优。在聚类过程中，初始中心的选择会对聚类结果产生重要影响。随机选择初始中心虽然简单，但可能导致算法收敛到局部最优解，增加迭代次数。为了改进这一点，有几种方法可以更好地确定初始中心，从而减少算法的迭代次数，提高聚类结果的稳定性和质量。下面针对 K-means++方法进行分析。

K-means++方法是一种改进的初始化方法，旨在选择更加分散的初始中心，以提高聚类结果的质量。其步骤如下：

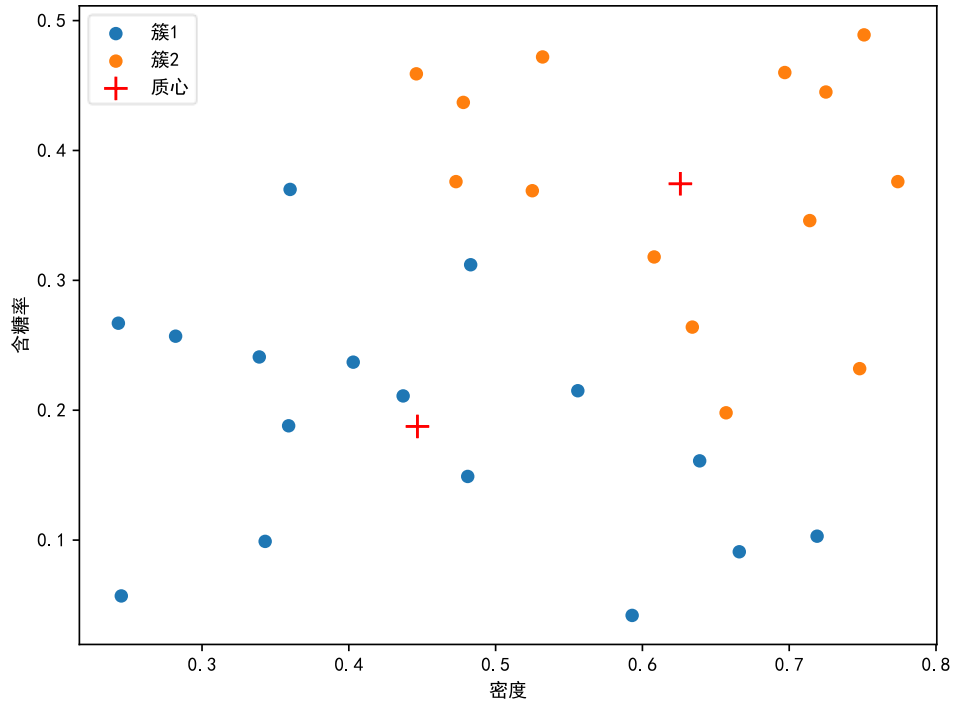
1. 随机选择一个数据点作为第一个初始中心。
2. 对于每个剩余的数据点x，计算它与最近的已选择中心点之间的距离$D(x)$。
3. 以 $D(x)^2$ 作为概率分布，从剩余的数据点中选择下一个初始中心。
4. 重复步骤 2 和 3，直到选择出 k 个初始中心。

这种方法可以显著提高 K-means 的聚类效果和收敛速度。

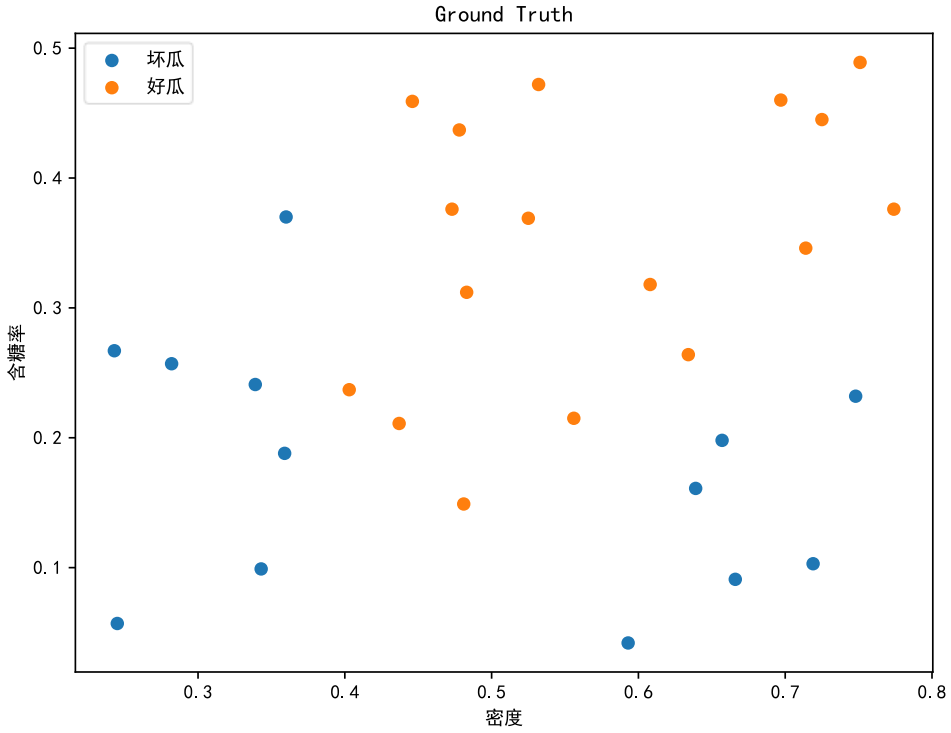
停止规则设定为两次迭代间，均值向量更新的变化小于一定的容限ε。

为了说明这一方法的有效性，注意剩余数据点x与最近的已选择中心点之间的距离$D(x)$表征了簇间相似度，为了使簇间相似度尽可能低，就要使$D(x)$尽可能大，这对应了更加分散的初始中心。而以$D(x)^2$作为概率分布，进一步增大了作出这种选择的可能性。至于按概率分布进行选取而非直接选择$\underset{x}{\operatorname{argmax}} D(x)$作为下一个聚类中心，是因为$k$均值算法是启发式算法，随机选取有助于算法跳出局部最优。实际进行聚类时，通常运行多次（例如 10 次），然后选择 SSE 最小的作为最终结果。或者可以考虑使用模拟退火算法、遗传算法等以进一步跳出局部最优。

使用 K-means++初始化的改进算法进行聚类，其中一次进行 3 次迭代后收敛，结果如下：



“西瓜书”中提到，样本 9~21 的类别是“好瓜=否”，其他样本的类别是“好瓜=是”，根据这一信息，绘制出西瓜数据集的真值（Ground Truth）图如下：



可以发现聚类结果与真值有一定的差异，在无监督情况下，并不能自发找出区分好瓜与坏瓜的内在规律，这是因为k均值聚类的簇之间的分割边界是线性的，不能处理非凸形状的数据分布聚类问题。为了解决这一问题，可以采用核方法，将输入空间中的数据点映射到某个高维特征空间中，从而提取簇之间的非线性边界。

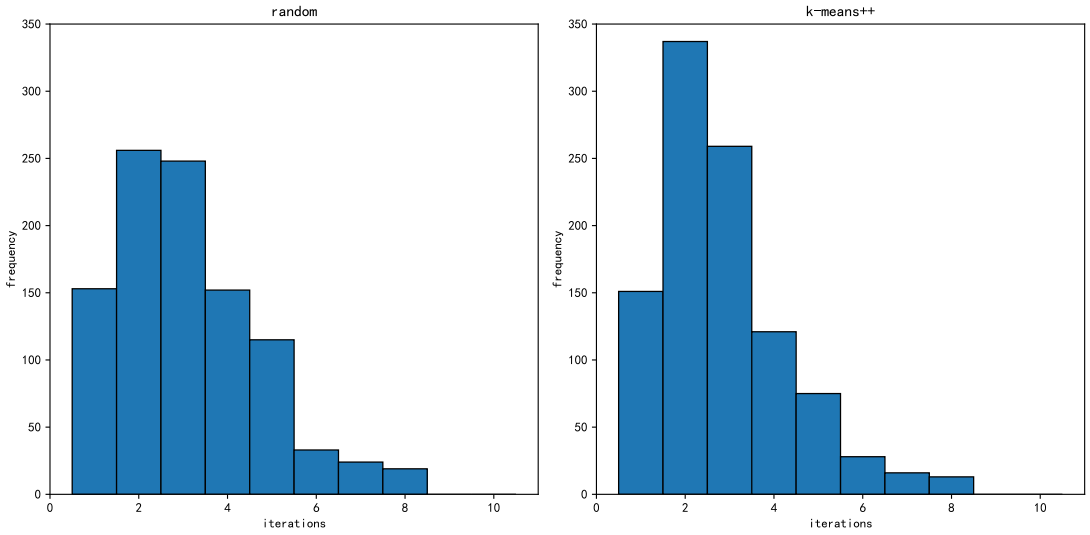
接下来比较改进算法与随机选择初始聚类中心的结果。由于两种算法每一次随机选取的初始中心都不同，聚类过程收敛时算法迭代的次数也各不相同。为了比较算法的优劣，运行两个算法各 1000 次，求出迭代次数的均值近似作为迭代次数的期望，并画出迭代次数分布的直方图以比较算法的性能。

运行结果如下：

算法	用时/s	平均迭代次数
random	0.35	3.11
k-means++	0.39	2.85

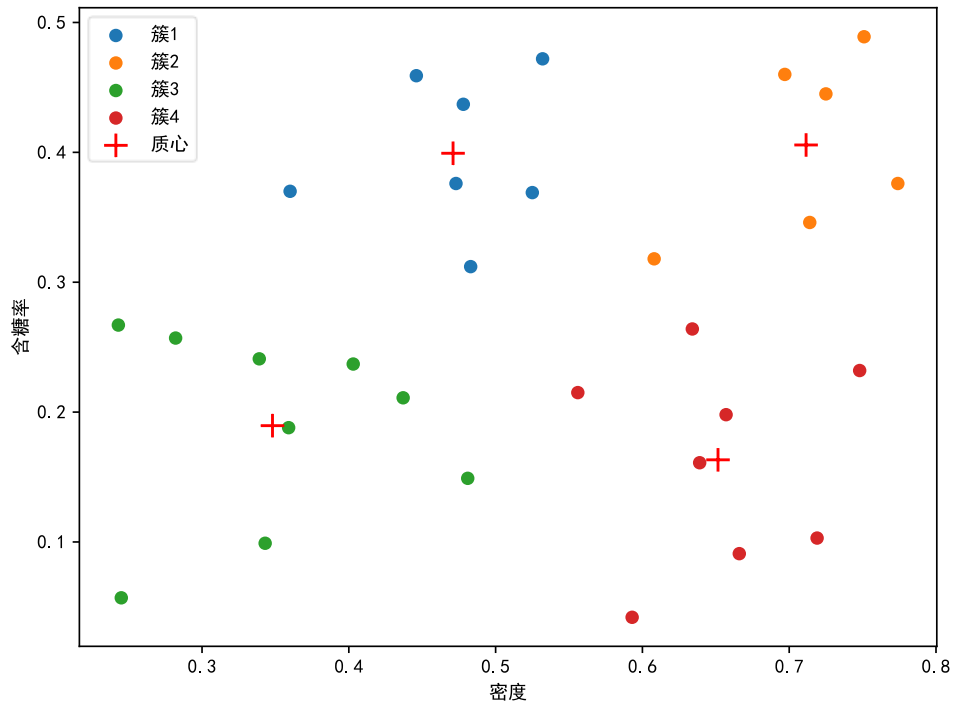
K-means++算法的用时稍长，但平均迭代次数比随机选择初始聚类中心要小。

迭代次数分布的直方图如下：



可以看出，K-means++的迭代次数集中在 2 和 3，整体迭代次数要比随机选择的低。

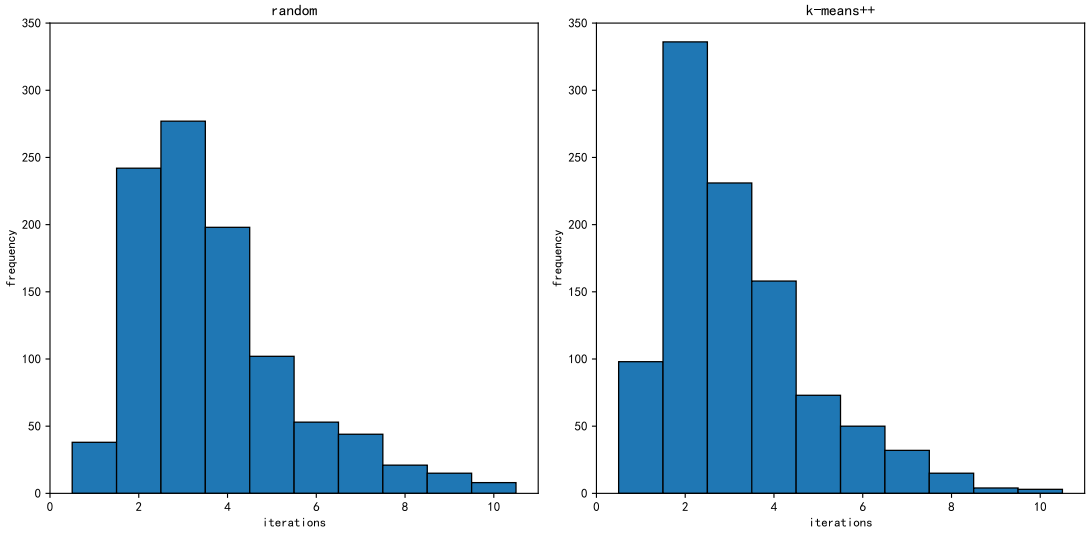
将$k = 2$改为$k = 4$重新比较算法性能，聚类结果如下：



两种算法的差异如下：

算法	用时/s	平均迭代次数
random	0.49	3.69
k-means++	0.59	3.17

迭代次数的分布如下：



由此说明 K-means++针对不同的k值，迭代次数的期望都小于随机选择初始聚类中心的情形，但自动确定初始聚类中心的过程会带来一些额外的计算消耗。

此外，还可以先用层次聚类对数据进行初步聚类，得到k个类时停止，然后从每个类别中选取一个与中心距离最近的点作为 K-means 的初始中心。这种方法利用了其他算法的优势，提供了一个较好的初始解。