电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

硕士学位论文

MASTER THESIS

论文题目 面向大数据的扁平聚类算法研究

学科专业 软件工程

学　　号 201721220117

作者姓名 任远航

指导老师 罗绪成　副教授

分类号 _____ 密级 _____

UDC 注 1 _____

学　位　论　文

面向大数据的扁平聚类算法研究

（题名和副题名）

任远航

（作者姓名）

指导老师 罗绪成　副教授

电子科技大学　成都

（姓名、职称、单位名称）

申请学位级别 硕士 学科专业 软件工程

提交论文日期 _____ 论文答辩日期 _____

学位授予单位和日期 电子科技大学　年　月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

Flat Clustering Algorithms for Big Data

A Master Thesis Submitted to
University of Electronic Science and Technology of China

Discipline:	**Software Engineering**
Author:	**Yuanhang Ren**
Supervisor:	**A.P. Xucheng Luo**
School:	**School of Information and Software Engineering**

摘　要

为了适应日益增长的宽带信号和非线性系统的工程应用，用于分析瞬态电磁散射问题的时域积分方程方法研究日趋活跃。本文以时域积分方程时间步进算法及其快速算法为研究课题，重点研究了时间步进算法的数值实现技术、后时稳定性问题以及两层平面波算法加速计算等，主要研究内容分为四部分。

……

关键词：时域电磁散射，时域积分方程，时间步进算法，后时不稳定性，时域平面波算法

ABSTRACT

With the widespread engineering applications ranging from broadband signals and non-linear systems, time-domain integral equations (TDIE) methods for analyzing transient electromagnetic scattering problems are becoming widely used nowadays. TDIE-based marching-on-in-time (MOT) scheme and its fast algorithm are researched in this dissertation, including the numerical techniques of MOT scheme, late-time stability of MOT scheme, and two-level PWTD-enhanced MOT scheme. The contents are divided into four parts shown as follows.

Keywords: time-domain electromagnetic scattering, time-domain integral equation (TDIE), marching-on in-time (MOT) scheme, late-time instability, plane wave time-domain (PWTD) algorithm

目 录

第一章 绪 论

1.1 研究工作的背景与意义

在很多研究领域，研究进展的取得往往需要依赖大量数据的分析。比如在天体物理学领域，为了给黑洞“照相”，Bouman 等人需要处理 5PB 的数据[1]。在生物医学领域，一个人体样本的基因组数据会超过 100GB，由于一次实验会收集成百上千的人的数据，因此数据量十分巨大[2]。此外，由于记录数据的设备越来越多其他领域数据量也在不断扩大，比如传感器记录的行动数据、基因数据、照片、语音、金融日志、网络数据等。

海量数据被冠以术语 Big Data，Big Data 至少包含 3 层含义[3]，即数据量大（Volume of data）、数据处理速度快（Velocity of processing the data）与数据多样（Variety of data），合称“3V”。对于这样的数据往往需要使用一些自动化的方法来分析数据中重要的模式和子结构或者对数据进行压缩，这些方法包括聚类和降维。粗略地说，前者即是将数据分为不同的类，使得同类的数据相似，不同类的数据不相似，后者是将数据投影到一个低维空间使得高维空间的数据的结构能够在低维尽可能保留。对于聚类经典的方法有 *k*-means[4]、Spectral Clustering[5] 等，经典降维方法有 PCA[6]、ISOMAP[7]、LLE[8] 等。

与此同时，伴随信息技术的发展，人与人的链接变得日益密切，相关的文本、图像、音频等数据不断增加形成了大数据。为帮助人们从这些数据中梳理出有价值的信息，数据挖掘（Data Mining）技术应运而生。所谓数据挖掘便是从大量无序的数据中发现隐含的、有效的、有价值的、可理解的模式, 进而发现有用的知识, 并得出时间的趋向和关联, 为用户提供问题求解层次的决策支持能力[9]。在这些背景下，聚类作为一种主要的数据挖掘手段收到人们的重视，得到了蓬勃的发展。

故本篇文章聚焦聚类问题[9–11]。这里给出 Everitt[12] 在 1974 年对聚类的定义：一个类簇内的实体是相似的, 不同类簇的实体是不相似的; 一个类簇是测试空间中点的会聚, 同一类簇的任意两个点间的距离小于不同类簇的任意两个点间的距离; 类簇可以描述为一个包含密度相对较高的点集的多维空间中的连通区域, 它们借助包含密度相对较低的点集的区域与其他区域 (类簇) 相分离。形式化地讲，给定数据集 $D = \{x_1, x_2, ..., x_n\}$，以及一个能将任意一个点 x 映射到一个类 id 的划分函数 f，通过 f，数据可以被划分为若干组 $G_1, G_2, ..., G_k$（假定划分的组数是给定的 k），其中 $G_i \subseteq D$。给定一个评价函数 E，我们就能知道聚类的质量，聚类的目标便是

找到那个最优划分f^*，即

$$f^* = \underset{f}{\operatorname{argmax}} E(G_1, G_2, ..., G_k)$$

一个一般的聚类过程包括数据预处理、特征处理、聚类和结果评估这 4 个步骤。

聚类过程：

1. 数据预处理：包括但不限于原始数据清洗，处理缺失值
2. 特征处理：比如选择有利于聚类的特征，特征缩放（scaling），以及通过某些变换得到新的突出特征
3. 聚类：首先选择某种相似度度量，接着依照个人对于聚类的理解（先验知识）制定聚类的目标函数，最后优化该函数
4. 结果评估：选择一个评估方式对聚类结果进行评估，可以是有人工标准参与的评估也可以是没有人工标准的

1.2 常见聚类算法及评价指标介绍

对于聚类来说，没有“万能”算法，即没有任何一种聚类技术 (聚类算法) 可以普遍适用于揭示各种多维数据集所呈现出来的多种多样的结构[13]。因此需要根据不同的数据模式选择不同的聚类算法，有时也会考虑用户的需求而改变算法。下面介绍常见的聚类算法，他们可分为扁平式式聚类（flat clustering）和层次式聚类（hierarchical clustering），示例见图1-1。

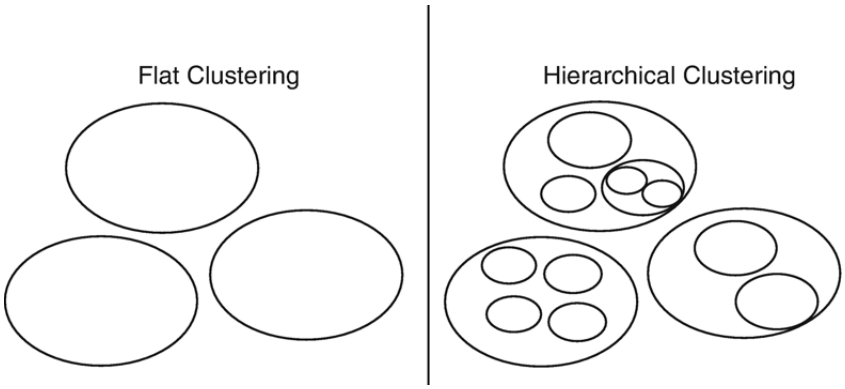


图 1-1 扁平聚类 vs 层次聚类，核心区别是扁平聚类不体现类间关系，层次聚类反映了类的蕴含关系

1.2.1 扁平聚类

扁平聚类指的是得到的类，两两没有交集，所有类的并集构成了所有数据点。这一聚类的代表就是 *k*-means 聚类，准确说，我们先有 *k*-means 问题，然后才有了

相关的解决这一问题的算法。*k*-means 问题指的是给定数据集，寻找 *k* 个点使得所有点到这 *k* 个点的距离平方和最小。在所有解决的算法中最有名的可能就是 Lloyd 算法 [4] 了，该算法分为以下 3 步。第一步，从数据集中均匀采样 *k* 个点作为初始化的中心点；第二步，所有点靠到离自己最近的中心点上去，形成 k 个类；第三步，取 k 个类的中心点作为新的中心点。一般来说，第二、三步会重复 *t* 次，因此算法的时间复杂度是 $O(nkdt)$，其中 n，d 分别指的是待聚类的数据量和数据的维度。作为一个经典聚类算法，该算法被大量应用在自然语言处理、数据挖掘、以及其他领域中。这个算法有一些优点，首先它比较简单，易于实现。其次，它易于并行，在中等数据量下可以很好运行。但是它的一些缺点也非常瞩目，聚类的 *k* 值不好确定，受初始点影响大，仅能对球形数据进行聚类（如果数据非球形则聚类结果和人的直观感觉有较大出入）。

为了解决最后一个问题，谱聚类（spectral clustering）被提了出来。这一聚类算法创新性的从图分割的角度给聚类带来了新的视角，从这一视角下，一些传统上不容易被聚类的数据集也能被攻克。它的基本思想是在数据集上构建一个图，聚类问题转化为大图切割为子图的问题，好的聚类应该是，子图内部节点相似度高，子图之间相似度低。基于这一“好的”聚类的准则，人们制定了称为 Ncut 的切割目标函数，这一目标函数的求解可以转换为图拉普拉斯矩阵的分解问题，通过特征值分解得到新的特征，再在这一特征上聚类从而得到最终结果。过程可总结如下：

1. 根据输入的相似矩阵的生成方式构建样本的相似矩阵 S
2. 根据相似矩阵 S 构建邻接矩阵 W，构建度矩阵 D
3. 计算出拉普拉斯矩阵 L
4. 构建标准化后的拉普拉斯矩阵 $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$
5. 计算 $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ 最小的 k 个特征值所各自对应的特征向量 f
6. 将各自对应的特征向量 f 组成的矩阵按行标准化，最终组成 $n \times k$ 维的特征矩阵 F
7. 对 F 中的每一行作为一个 k 维的样本，共 n 个样本，用 *k*-means 进行聚类，聚类数为 k
8. 得到类划分

实践中谱聚类能够很好处理 *k*-means 不能处理的非球形数据（比如同心圆、半月牙等），对比效果见图1-2，这可以从谱聚类的设计准则解释，因为是做图切割，而相似度又有多种非线性的选择，所以聚类的结果就可以实现 *k*-means 不能做到的非线性划分（*k*-means 用欧式距离所以决策边界是线性的）。由于本文研究的就是

k-means 和谱聚类，所以对于他们的详细剖析见第二、三章。

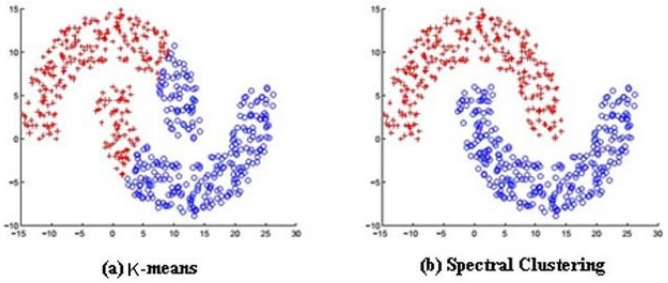


图 1-2 *k*-means vs 谱聚类，可以看到谱聚类能够处理 *k*-means 处理不了的非球形数据

1.2.2 层次聚类

尽管扁平聚类的概念更简单，但我们也看到了他们的一些缺点。首先，现实世界中类和类之间是有关系的，比如说灵长类属于哺乳类，这样的蕴含关系，在扁平聚类中这样的类间关系无法体现。其次，拿 *k*-means 问题来说，我们需要在聚类前指定类的数目 k，这在现实生活中未必好确定。最后，扁平聚类过程是不确定的（nondeterministic），即聚类结果有随机因素，比如 Lloyd 算法初始化不同结果就不同。为了解决这些问题，诞生了层次式聚类。顾名思义，层次式聚类得到的类有层级结构，可以用树状图（tree/dendrogram）来表示，如图1-3所示，他们也不需要指定类的数目，多数层级聚类也是确定的。这些好处的代价就是聚类速度更慢，常用的层级聚类时间复杂度至少是 $O(n^2)$。

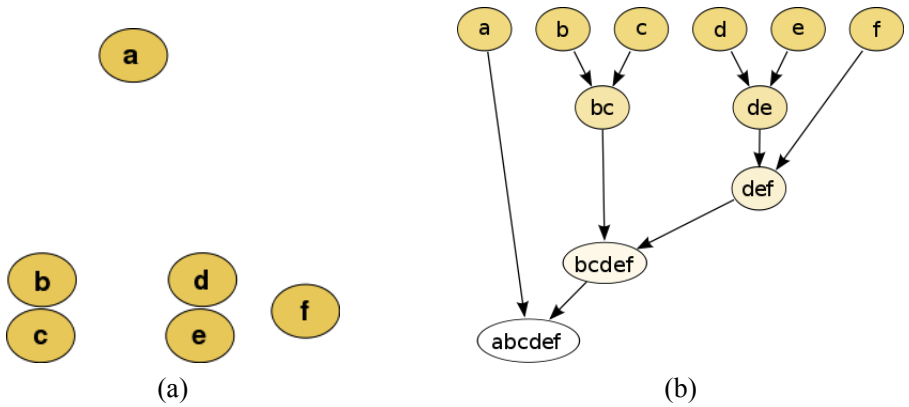


图 1-3 层级聚类图示。（a）示例数据集，每个字母代表一个数据点，距离越近相似度越高；（b）层级聚类后的结果，相似的类归并为一个更大的类从而形成一个二叉树，树的根节点在最下面

层级聚类可以自底向上做，将小的类逐渐归并得到大的类，也可以自顶向

下，将大的类逐渐分割得到小的类，由此得到两种不同的聚类范式。前者称为agglomerative clustering，后者称为 divisive clustering。我们先介绍前者，agglomerative clustering 的一般过程如下所示

1. 初始化：每个点自己作为一个类
2. 直到所有类归并为一个类，否则
   (a) 找出两个最相似的类
   (b) 将他们归并为一个类（父类）
3. 返回类的层级结果（树）

可以看到，这是一个贪心算法，如果有 n 个点，这个合并的过程就会持续 $n-1$ 次，如果运气好，树的深度就是 $\log_2 n$。由于树的结构，我们就可以知道类的蕴含关系，那么如何计算类和类的相似度呢？这里我们需要定义一些常用的计算方式，这些计算方式一般被称为 linkage criteria，见表1-1。

表 1-1 几种常见的 linkage criteria

名字	计算方式
Maximum linkage	$\max\{d(a,b) : a \in A, b \in B\}$
Minimum linkage	$\min\{d(a,b) : a \in A, b \in B\}$
Unweighted average linkage	$\frac{1}{\lvert A\rvert \cdot \lvert B\rvert} \sum_{a\in A}\sum_{b\in B} d(a,b)$
Centroid linkage	$\lVert \mu_A - \mu_B \rVert$
Ward linkage	$\frac{n_a n_b}{n_a + n_b} \lVert \mu_A - \mu_B \rVert^2$

表 1-2 几种常见的距离计算

名字	计算方式
Euclidean distance	$\lVert a-b\rVert_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\lVert a-b\rVert_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\lVert a-b\rVert_1 = \sum_i \lvert a_i - b_i\rvert$
Maximum distance	$\lVert a-b\rVert_\infty = \max_i \lvert a_i - b_i\rvert$

表格1-1中的 d 是点与点间的距离计算方式，这有很多种选择，这里挑了些常用的列在表1-2中，A, B 是两个类，μ_A, μ_B 是这两个类的中心点，n_a, n_b 是两个类中点的数目。可以看到，Maximum/Minimum/Centroid 只考虑单一的点对来计算类间相似度，而 Ward/Unweighted average 基本考虑了全部的。值得一提的是，agglomerative clustering 在聚类的过程会倾向与让大的类更大，从而导致大小不平衡的类，从这个角度讲，Minimum linkage 是最差的 linkage，Ward 会形成差不多大小的 cluster。然而在 Ward 中的距离计算是不可改变的（只有欧式距离），因

此对于非欧的距离度量，average linkage 比较合适。尽管 Minimum linkage 易受到噪声数据的影响，但它计算比较高效（时间复杂度是 $O(n^2)$，一般的其他 linkage 是 $O(n^2 \log n)$），因此这一 linkage 可以用到较大的数据量上，在非球形数据上 Minimum linkage 也有好的表现。

现在我们来看 divisive clustering，前面说过，这是一个自顶向下的层级聚类，它需要挑选一个类，将这个类分为两个子类，并循环这个过程，直到终止条件触发（比如聚到指定数目的类）。从这个过程我们就可以看出该算法主要考虑两件事，如何挑选待分割的类以及如何分割这个类。前者一般通过某种预定义的松散度度量，比如类内最远的两个点的距离，选这个值最大的类进行分割。分割的方法一般是用扁平式聚类法，因为又包括了其他算法，divisive clustering 要更复杂些。如果我们不要求划分太过细致，算法就可以在中间某层结束，如果结合 Lloyd 算法使用，假设我们想得到 s 个类，时间复杂度就是 $O(nsdt)$，d 和 t 分别是数据点维度和迭代次数，这样就会比 agglomerative clustering 快很多。有研究指出 divisive clustering 比 agglomerative clustering 得到的层级结果质量更好[14]，这可能是因为 agglomerative 是自底向上，在每次合并的时候只能看到局部的信息而不考虑全部信息，而合并过程又不可以反悔，所以 divisive clustering 由于是看到了全部信息做的划分，能得到更好的结果。尽管 divisive clustering 有这些优点，它的使用却没有 agglomerative clustering 频繁，可能是处理复杂情况时自底向上更符合人的习惯。

1.2.3 聚类评价

聚类评价方式分为两种，外部评价（external index）和内部评价（internal index）。前者需要有人给出的划分结果作为标准而后者不利用任何参考。我们先看外部评价，假定算法给出的类划分是 $C = \{C_1, C_2, ..., C_k\}$，人给出的划分是 $C^* = \{C_1^*, C_2^*, ..., C_s^*\}$。考虑一对数据对 (x_i, x_j)，假设 x_i, x_j 在 C 中在同一个类里，如果算法给出的划分和人的划分是一致的，那么它们在 C^* 中也应该在同一个类里。令 $\lambda_i = \{1, 2, ..., k\}$ 是数据点 x_i 的类 id，即 $x_i \in C_{\lambda_i}$。那么对于刚刚所说的情况可以定义一个集合 SS 来容纳这些数据对

$$SS = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$

也就是说，如果数据对在 SS 集合中就意味着聚类是成功的。当然也有不成功的情况，我们定义为 SD 和 DS。

$$SD = \{(x_i, x_j) | \lambda_i = \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$
$$DS = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* = \lambda_j^*, i < j\}$$
$$DD = \{(x_i, x_j) | \lambda_i \neq \lambda_j, \lambda_i^* \neq \lambda_j^*, i < j\}$$

最后一行是另外一种成功的情况称为 DD。值得注意的是，这几种情况是互不相交的，定义

$$TP = |SS|, FP = |SD|, FN = |DS|, TN = |DD|$$

因此有 $TP + FP + FN + TN = n(n-1)/2$。基于 TP、FP、FN 和 TN，我们可以定义以下外部评价指标（F_1，Rand 系数，Jaccard 系数）

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, F_1 = \frac{2 * P * R}{P + R}$$
$$\text{Rand} = \frac{TP + TN}{TP + FP + FN + TN}$$
$$\text{Jaccard} = \frac{TP}{TP + FP + FN}$$

F_1 和监督学习中的一致，Rand 系数也被称为 Accuracy。除去上面这些还有简单实用的 Purity 以及有信息论背景的 NMI（Normalized Mutual Information），当类的数目比较大的时候，Purity 容易比较大，而 NMI 缓解了此问题。

$$\text{Purity} = \frac{1}{n} \sum_i \max_j n_{ij}$$
$$\text{MI}(C, C^*) = \sum_{i=1}^{|C|} \sum_{j=1}^{|C^*|} P(i,j) \log \left(\frac{P(i,j)}{P(i)P(j)} \right)$$
$$H(C) = -\sum_{i=1}^{|C|} P(i) \log(P(i))$$
$$\text{NMI}(C, C^*) = \frac{\text{MI}(C, C^*)}{\text{mean}(H(C), H(C^*))}$$

其中，$n_{ij} = |C_i \cap C_j^*|$，$n_i = |C_i|$，$P(i) = n_i/n$，$P(i,j) = n_{ij}/n$。$P(i)$ 可以理解为一个数据点落在类 i 中的概率，$P(i,j)$ 可以理解为一个点在既在 C_i 又在 C_j^* 中的概率。MI 称为互信息（mutual information），H 是熵。这里所列的外部评价都是越大越好。

除了外部评价，内部评价在没有人工标注的情况下使用，比如 k-means 的目标

值（objective）可以作为评价指标，该评价指标是聚类问题相关的，评价 *k*-means 算法用，评价其他算法慎用，还有一些聚类问题无关的评价指标，比如 DB 指数（Davies-Bouldin Index）

$$\text{avg} = \frac{2}{|C|(|C|-1)} \sum_{1\le i<j\le |C|} d(x_i, x_j)$$

$$d_{cen}(C_i, C_j) = d(\mu_i, \mu_j)$$

$$\text{DBI} = \frac{1}{k}\sum_{i=1}^{k} \max_{j\neq i}\left(\frac{\text{avg}(C_i)+\text{avg}(C_j)}{d_{cen}(\mu_i,\mu_j)}\right)$$

其中 d 可以是表1-2中的距离函数。

1.3 本文的研究目标及主要贡献和创新

在大数据下聚类，我们应该关心什么问题呢？我们应该注意那些普遍且真实的问题，所以，首先，我们应该考虑的就是聚类速度，传统聚类算法由于其自身较高的时间复杂度，在目前大数据时代越来越难以适用。其次，聚类算法的聚类质量也是一个主要问题，很多算法声称自己有好的聚类质量，并且也在一部分数据上得到了证实，但是可能换一个数据集这个算法效果就不好了，对于这种情况有时很难分析问题在哪里，这种时候我们就会希望这个算法在理论上有质量保证，比如说可以证明该算法不会比最优算法的聚类结果差多少倍。总结一下，在本文中我们将探究以下两个问题：

1. 怎样聚类更加高效？
2. 怎样聚类得出的解有理论保证？

鉴于扁平聚类的普适性，我们将围绕 *k*-means 问题和谱聚类问题探究我们所提出的问题并给出我们的贡献。具体来说，我们的贡献和创新有以下几点：

1.（理论 & 实验贡献）对基于均匀采样加速 *k*-means 的算法给出了更紧的理论界，从 $4(\alpha+\beta)$ 收缩到了 $\alpha+\beta$，同时，我们证明了在温和的假设下，该算法的时间复杂度在多项式对数级别（polylogarithmic time），随后我们在实验上对理论结果进行了验证
2.（理论贡献）将经典的 *k*-means++ 初始化方法扩展到了带权重的 *k*-means 问题上并给出了证明
3.（理论 & 实验贡献）对基于 Weighted kernel *k*-means 的谱聚类算法给出了更紧的理论界，并给出了 Weighted kernel *k*-means 的 MATLAB 实现。

1.4 本论文的结构安排

本文的章节结构安排如下：第一章是绪论，介绍常见聚类算法为后续探讨打下初步基础同时明确指出要研究的问题和我们的创新，第二章探讨 k-means 问题，将会分别深入分析与回答我们所提出的两个问题。第三章探讨谱聚类问题，同样加速和理论保证分别分析。第四章是结论，在总结全文的基础上给出未来可能的研究方向。

第二章 k-means 聚类

本章聚焦 k-means 问题，前面已经提到解决该问题的一个经典算法 Lloyd 算法，由于文献 [15–17] 指出 k-means 问题是 NP 难的，所以该算法准确来说是一个近似算法。前面提到，该算法收到初始点影响大，所以本章将首先探讨有理论保证的 k-means 问题的算法。其次，$O(nkdt)$ 的时间复杂度在海量数据下也难以处理，所以接着我们会探讨如何加速。然后，我们探讨如何将这两点都覆盖，最后我们给出我们的贡献并总结本章。

2.1 k-means 问题引入及背景知识

我们首先定义 k-means 问题，给定点 $x \in \mathbb{R}^d$ 和集合 $C \subseteq \mathbb{R}^d$，d 是数据的维度，我们可以定义该点到集合的距离

$$d(x, C) = \min_{c \in C} \|x - c\|$$

给定 n 个点的数据集 $\mathcal{X} \subseteq \mathbb{R}^d$ 和集合 $C \subseteq \mathbb{R}^d$，我们可以定义如下目标函数

$$\phi(\mathcal{X}, C) = \sum_{x \in \mathcal{X}} d^2(x, C)$$

我们的目标是在给定 n 个点 $\mathcal{X}$ 情况下找到大小为 k 的集合 $C^{\mathcal{X}}_{\mathrm{OPT}_k}$ 使得该目标函数最小，即

$$C^{\mathcal{X}}_{\mathrm{OPT}_k} = \operatorname*{argmin}_{\substack{C \subseteq \mathbb{R}^d \\ |C| = k}} \phi(\mathcal{X}, C)$$

上文谈到解的质量，这里给出如下定义用于定量评价解的质量。

定义 2.1 (k-means **问题解的质量**) 一个算法 $\mathcal{A}$ 被称为 k-means 问题的一个 α 近似算法如果存在一个数字 $\alpha \geq 1$ 使得

$$\phi(\mathcal{X}, C) \leq \alpha \phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

成立，此时称解 C 为一个 α 近似解，这里 α 称为近似系数。

从该定义可以看出，α 越小解质量越好，越接近最优解。

2.2 给 k-means 的解提供质量保证

选择好的初始化方法（seeding）是提供质量保证的一条路线，在 Lloyd 算法中传统初始化方法一般是均匀不放回的抓取 k 个点，这不是一个好的初始化方法，由于 Lloyd 算法对初始点很敏感[18]，这种方法为了取得好的效果经常需要多次初始化取最好的，除去花费不少时间此种方法没有理论上的保证，而 David 等人[19]则在这一方向上做出了巨大的贡献。

首先，从直觉上想，当初始化的点聚在一起的时候该初始化就不是一个好的初始化，因此一种容易相到的就是每次新选择的点都应离已经选择的点尽可能远，但是如果直接选择最远的点的话就可能是异常点，为此可以这样改进该算法，每一个点以一个概率被选中，而该概率与该点到已选择到的点的距离成正比，由此便得到了 k-means++ 算法[19]，算法详细描述如算法2-1

算法 2-1 k-means++ seeding

输入: 数据集 $\mathcal{X}$，类数目 k
输出: k 个点 C

$c_1 \leftarrow$ 从 $\mathcal{X}$ 中均匀采样一个点
$C \leftarrow \{c_1\}$
for $i = 2, 3, \ldots k$ **do**
  for $x \in \mathcal{X}$ **do**
    $p(x) \leftarrow \mathrm{d}(x, C)^2 / \sum_{x' \in \mathcal{X}} \mathrm{d}(x', C)^2$
  end
  $x \leftarrow$ 从 $\mathcal{X}$ 中依分布 $p(x)$ 采样一个点
  $C \leftarrow C \cup \{x\}$
end
返回 C

该算法的时间复杂度为 $O(nkd)$，实验验证在使用 k-means++ seeding 后 Lloyd 算法能更快收敛。另外，对于 k-means++ 以下定理成立

定理 2.2 (k-means++ **解的质量**) 对于任意数据集 $\mathcal{X}$

$$\mathbb{E}[\phi(\mathcal{X}, C)] \leq 8(\ln k + 2)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

其中 C 算法2-1返回的结果

可以看出即使以初始化的 k 个点作为最后的解，解的平均质量已经能够达到 $O(\log k)$，后续 Lloyd 算法的步骤只会让解更好。

那么，能否获得常数近似的解呢？设想现在已经有一个解的集合 $T \subset \mathcal{X}$，为了在 T 基础上得到更好的解，可以采取以下策略，将点 $t \in T$ 替换为 $T' \in \mathcal{X} - T$ 得到

T'，如果 T' 的解质量更好，则用 T' 替换 T，重复该步骤直到解质量不再变化。算法描述见算法2-2，在实际应用中这个替换会持续很久，不过，通过对文献 [20] 和文

算法 2-2 启发式本地搜索算法 (p-swap)

输入: 数据集 $\mathcal{X}$，类数目 k
输出: k 个点 T
1 $T \leftarrow$ 从 $\mathcal{X}$ 中随机选 k 个点
2 **while** $\exists t \in T, t' \in \mathcal{X} - T$ 使得 $\phi(\mathcal{X}, T - \{t\} + \{t'\}) < \phi(\mathcal{X}, T)$ **do**
3 $\quad T \leftarrow T - \{t\} + \{t'\}$
4 **end**
5 **返回** T

献 [21] 的结果进行拓展，该算法可以在多项式时间（polynomial time）内收敛。首先，对于任意 1 次替换不能再减少 k-means 目标函数值的解称其为 1-stable 的解。基于此，Arya 等人[20] 和 Kanungo 等人[22] 对解的质量给出了如下的理论结果。

定理 2.3 (1-stable **的解的质量**) 令 $O \subset \mathcal{X}$，且是所有大小为 k 的 $\mathcal{X}$ 的子集里能取得最小 k-means 目标函数值的那个解，令 T 是 1-stable 的解，则有

$$\phi(\mathcal{X}, T) \leq 25\phi(\mathcal{X}, O)$$

值得注意的是 O 并不是 $C^{\mathcal{X}}_{\mathrm{OPT}_k}$，通过简单的推论，得知近似系数是 50（乘以 2）。那对于任意 p-stable 的解，近似系数是多少呢？Kanungo 等人[22] 对算法2-2的分析做了推广得到以下结论。

定理 2.4 (p-stable **的解的质量**) 令 $O \subset \mathcal{X}$，且是所有大小为 k 的 $\mathcal{X}$ 的子集里能取得最小 k-means 目标函数值的那个解，令 T 是 p-stable 的解，则有

$$\phi(\mathcal{X}, T) \leq (3 + \frac{2}{p})^2 \phi(\mathcal{X}, O)$$

可见对于充分大的 p，系数是 18。实验表明算法2-2收敛相对于 Lloyd 算法收敛慢，但是该算法能够避免陷入局部极小值。所以，实际使用时会和 Lloyd 算法混合使用，即先用一次替换再进行 Lloyd 算法的 2、3 步，这样的混合算法在实际中有很好的表现，文献 [22] 对此做了验证。

k-means++ 虽然实用，但是近似系数只有 $O(\log k)$，本地搜索算法虽然有常数的近似系数，但是不实用，时间复杂度有 $O(n^3)$，能否将这两种算法结合起来得到一个既有常数近似又快速实用的算法呢？Silvio 等人[23] 便带来了一个这样的算法，算法描述见算法2-3。

算法 2-3 k-means++ 和本地搜索算法的混合算法（k-means++&swap）

输入: 数据集 $\mathcal{X}$，类数目 k
输出: k 个点 C
$C \leftarrow$ 将 $(\mathcal{X}, k)$ 应用于算法2-1
for $i = 2,3,...,z$ **do**
  $C = \text{LocalSearch++}(\mathcal{X}, C)$
end
返回 C

算法 2-4 LocalSearch++ 算法

输入: 数据集 $\mathcal{X}$，解集 C
输出: k 个点 C
$p \leftarrow$ 以 $d^2(p,C)/\sum_{q\in\mathcal{X}} d^2(q,C)$ 的概率采样 p
if $\exists q \in C$ 使得 $\phi(\mathcal{X}, C-\{q\}+\{p\}) < \phi(\mathcal{X}, C)$ **then**
  $q \leftarrow$ 选择一个 q 使得 $\phi(\mathcal{X}, C-\{q\}+\{p\})$ 最小
  $C = C-\{q\}+\{p\}$
end
返回 C

定理 2.5 (k-means++ **和本地搜索的混合算法的解的质量**) 令 C 是算法2-3的解，且 $z \geq 100000k\log\log k$，则有

$$\mathbb{E}[\phi(\mathcal{X}, C)] \in O(\phi(\mathcal{X}, C^{\mathcal{X}}_{\text{OPT}_k}))$$

算法2-3的理论保证由定理2.5给出，该定理说明只要迭代大概 $O(k)$ 次 LocalSearch++，解的近似系数就能达到常数，此时算法2-3的时间复杂度是 $O(ndk^2\log\log k)$。实验表明在不使用 Lloyd 算法后续迭代的情况下，混合的算法2-3比 k-means++ 好 8- 35%，而使用 Lloyd 迭代 10 次后，依然能比 k-means++ 好 1-18%

尽管 k-means++ 在线性时间里能得到不错的解，但是其序列化属性使得该算法无法并行，因此在大数据下难以适用，k-means|| 采样便是为了解决这个问题而被提出的，它的想法是基于已经挑选出的点集，下一次不止挑选一个点而是挑选一个集合，其期望大小为 l，将若干次挑选出来的集合汇总之后再在其上做聚类即可，采样算法详细描述见算法2-5，对于算法2-5有如下理论保证

定理 2.6 (k-means|| **采样理论保证** 1) 令 S 是算法2-5 t 次迭代后返回的结果，则

算法 2-5 *k*-means|| 采样算法

输入: 数据集 $\mathcal{X}$，每一轮期望采样数 l
输出: 采样集 S

$S \leftarrow$ 在 $\mathcal{X}$ 中均匀采样一个点
$\xi \leftarrow \phi(\mathcal{X}, S)$
for $i = 1,2,...,\log\xi$ **do**
  $C' \leftarrow \emptyset$
  for $x \in \mathcal{X}$ **do**
    以概率 $\min(1, \frac{ld^2(x,S)}{\phi(\mathcal{X},S)})$ 将 x 加入 C' 中
  end
  $S \leftarrow S \cup C'$
end
返回 S

有

$$\mathbb{E}[\phi(\mathcal{X}, S)] \leq (\frac{1+\alpha}{2})^t\xi + \frac{16}{1-\alpha}\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

其中 $\alpha = exp(-(1-e^{-l/(2k)})) \approx e^{-\frac{l}{2k}}$

上述定理说明算法2-5经过 $O(\log\xi)$ 次迭代之后，只要 l 足够大，所有点到挑选出来的 S 个点的距离和已经降到了 $O(\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k}))$，也就是说和最优的距离和已经在一个级别了。接着在 S 上聚类便可以得到 *k*-means 的解，在这里为了使解有理论上的保证，需要对 S 上的点加权重，首先定义如下带权 *k*-means 问题。

定义 2.7 **(带权 *k*-means 问题)** 给定集合 $\mathcal{X} \subseteq \mathbb{R}^d$ 和每一个 $x \in \mathcal{X}$ 的权重 w_i, 找到一个大小为 k 的集合 $C \subseteq \mathbb{R}^d$ 使得下述目标函数能最小

$$\psi(\mathcal{X}, C) = \sum_{x\in\mathcal{X}} w_i d^2(x, C)$$

接着在带权重的 S 上聚类得到算法2-6，可以证明，如果使用的是 α 近似算法，

算法 2-6 *k*-means|| seeding 算法

输入: 数据集 $\mathcal{X}$，每一轮期望采样数 l，聚类数 k
输出: k 个点 C

$S \leftarrow$ 将 $(\mathcal{X}, l)$ 应用于算法2-5
对点 $s_i \in S$，令 w_i 是 $\mathcal{X}$ 中离 s_i 最近的点的数目
$C \leftarrow$ 令 w_i 是 s_i 的权重，在带权的 S 上运行一个 α 近似算法
返回 C

那算法2-6输出的 k 个点便是原 *k*-means 问题的一个 $O(\alpha)$ 近似解。在实践中发现当

l 较大时，往往不需要 $\log\xi$ 次迭代，O. Bachem [24] 的分析证实了这一点，令迭代次数为 t，则有下述定理。

定理 2.8 (k-means|| **采样理论保证** 2) 令 $k \in \mathbb{N}$，$t \in \mathbb{N}$，且 $l \geq k$，令 $\mathcal{X} \subseteq \mathbb{R}^d$ 且 S 是算法2-5返回的结果，则

$$\mathbb{E}[\phi(\mathcal{X}, S)] \leq 2(\frac{k}{el})^t \mathrm{Var}(\mathcal{X}) + 26\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

该定理同定理2.6关键区别是只要 l 足够大，指数衰减的速度可以任意快，而定理2.6的衰减速度最多是 1/2。虽然 k-means|| 时间复杂度是 $O(nld\log n)$（如果 t 次迭代则是 $O(nltd)$），比 Lloyd 算法高，但是它易于并行，因此在能够并行的情况下很多经典大数据处理框架都采用了该算法比如 Spark [25,26]。

除了改善 k-means++ 的序列化属性来加速 seeding 外，还可以用其他方法加速。通过分析即可知道在 k-means++ 方法中最费时间的是找到计算概率所需要的分母，有没有在不知道分母的情况下近似采样分布的方法呢？有，该算法便是 MH 算法 [27]，该算法属于 MCMC 方法，其基本想法是构建一个马尔可夫链使得其平稳分布（stationary distribution）等于要采样的分布，这样就可以通过随机游走的方式来近似采样，基于 MCMC 来近似 k-means++ 的方法称为 K-MC^2 [28]，Bachem 等人通过对数据集生成的分布做出假设使得基于马尔科夫链随机游走的时间复杂度被限制在次线性时间（sub-linear time）内，从而让 K-MC^2 在 $O(k^3 d\log^2 n\log k)$ 的时间复杂度下获得了 k-means++ 的近似系数。K-MC^2 算法见算法2-7，算法中 $Unif(0,1)$ 指的是 0-1 中均匀采样的一个数。为了让游走能尽快收敛到平稳分布，我们定义如下两个数据相关的指标：

$$\zeta(\mathcal{X}) = \max_{x\in\mathcal{X}} \frac{d(x,\mu(\mathcal{X}))^2}{\sum_{x'\in\mathcal{X}} d(x',\mu(\mathcal{X}))^2}, \eta(\mathcal{X}) = \frac{\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_1})}{\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})}$$

假定数据是从生成数据的分布 F 中独立同分布的采样出来的，且该分布满足以下

两个假设，如果满足这些假设的话 $\zeta(\mathcal{X})$ 和 $\eta(\mathcal{X})$ 就可以被限定住，游走就能很快收敛到平稳分布

假设 1 分布 F 的尾部是指数衰减的，即 $\exists c,t$ 使得 $P[d(x,\mu(F)) > a] \leq ce^{-at}$，其中 $x \sim F$

假设 2 F 在一个球面上的最大和最小概率密度被一个常数限制

如果满足假设 1 和 2 分别有以下推论

推论 2.9 如果假设 1 成立，则有

$$\zeta(\mathcal{X}) = O(\log^2 n)$$

算法 2-7 K-MC^2 seeding

输入: 数据集 $\mathcal{X}$，游走次数 m，类数目 k
输出: k 个点 C

$c_1 \leftarrow$ 从 $\mathcal{X}$ 中均匀采样一个点
$C \leftarrow \{c_1\}$
for $i = 2, 3, \ldots k$ **do**
  $x \leftarrow$ 从 $\mathcal{X}$ 中均匀采样一个点
  $d_x \leftarrow \mathrm{d}(x, C)^2$
  for $j = 2, 3, \ldots m$ **do**
    $y \leftarrow$ 从 $\mathcal{X}$ 中均匀采样一个点
    $d_y \leftarrow \mathrm{d}(y, C)^2$
    if $\frac{d_y}{d_x} > Unif(0,1)$ **then**
      $x \leftarrow y, d_x \leftarrow d_y$
    end
  end
  $C \leftarrow C \cup \{x\}$
end
返回 C

推论 2.10 如果假设 2 成立，则有

$$\eta(\mathcal{X}) = O(k)$$

于是我们得出定理2.11。

定理 2.11 在假设 1 和 2 成立的情况下

$$\mathbb{E}[\phi(\mathcal{X}, C)] \leq O(logk)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k}))$$

C为算法2-7的返回结果,此时,$m = O(k \log^2 n \log k)$,算法时间复杂度为$O(k^3 d \log^2 n \log k)$

该方法在次线性时间里完成了 $O(\log k)$ 的近似，因此在数据较大且不能很好并行的情况下应该优先使用该算法。然而，在实践中该方法有以下缺点：

1. 如果存在小的类且该类离其他类比较远，那由于均匀选择的点很难落在小的类里，这样该方法就会产生较大的距离和
2. 不少现实中的数据不满足上述假设，比如经常能观测到重尾分布（heavy tailed distributions）的数据
3. 验证上述假设是困难的，比如计算 $\zeta(\mathcal{X})$ 需要对数据进行两次遍历，计算 $\eta(\mathcal{X})$ 更是 NP 难的
4. 定理2.11并没有描述游走次数 m 和算法解的质量之间的权衡，只有选择特定的游走次数才有效，因此，如果假设不成立，就得不到任何理论上的保证

因此，需要提出新的算法解决这些问题。

AFK-MC^2 [29] 通过使用提议分布（proposal distribution）很好的解决了这些问题，首先通过该提议分布，$\zeta(\mathcal{X})$ 的假设得以废弃；第二，一种新的理论分析使得在不假设 $\eta(\mathcal{X})$ 的情况下也能获得解的质量的理论保证；最后，新的结果表明了游走次数 m 和算法质量之间的权衡。AFK-MC^2 描述在算法2-8中。可以看出点的选

算法 2-8 AFK-MC^2 seeding

输入: 数据集 $\mathcal{X}$, 游走次数 m，类数目 k
输出: k 个点 C
/* 预处理 */
$c_1 \leftarrow$ 从 $\mathcal{X}$ 中均匀采样一个点
for $x \in \mathcal{X}$ **do**
  $q(x) \leftarrow \frac{1}{2}\mathrm{d}(x, c_1)^2 / \sum_{x' \in \mathcal{X}} \mathrm{d}(x', c_1)^2 + \frac{1}{2n}$
end
/* 主循环 */
$C \leftarrow \{c_1\}$
for $i = 2, 3, \ldots k$ **do**
  $x \leftarrow$ 依分布 q 从 $\mathcal{X}$ 中采样一个点
  $d_x \leftarrow \mathrm{d}(x, C)^2$
  for $j = 2, 3, \ldots m$ **do**
    $y \leftarrow$ 依分布 q 从 $\mathcal{X}$ 中采样一个点
    $d_y \leftarrow \mathrm{d}(y, C)^2$
    if $\frac{d_y q(x)}{d_x q(y)} > Unif(0,1)$ **then**
      $x \leftarrow y, d_x \leftarrow d_y$
    end
  end
  $C \leftarrow C \cup \{x\}$
end
返回 C

择依赖提议分布 q，因此小的类也能被选到点，从而解决了 K-MC^2 方法中的问题1。对于 AFK-MC^2，有如下定理

定理 2.12 (AFK-MC^2 **理论结果**) 令 $\varepsilon \in (0, 1)$ 且 $k \in \mathbb{N}$。令数据集 $\mathcal{X} \subseteq \mathbb{R}^d$，$|\mathcal{X}| = n$，$C$ 是算法2-8的输出，$m = 1 + \frac{8}{\varepsilon} log \frac{4k}{\varepsilon}$，则

$$\mathbb{E}[\phi(\mathcal{X}, C)] \leq 8(\log_2 k + 2)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k}) + \varepsilon \mathrm{Var}(\mathcal{X})$$

其中 $\mathrm{Var}(\mathcal{X}) = \sum_{x \in \mathcal{X}} d(x, \mu(\mathcal{X}))^2$，容易分析得，算法中预处理花费时间为 $O(nd)$，主要循环花费时间为 $O(\frac{1}{\varepsilon}k^2 d \log \frac{k}{\varepsilon})$。虽然预处理需要花费线性的时间，时间复杂度比 K-MC^2 高，但是我们有以下理由认为实践中该步骤问题不会很大

1. 对所有数据只需遍历一次
2. 该步骤很容易并行
3. 从定理可以看到，预处理后游走的步数从 $\log^2 n$ 变到了常数

因此该算法和 K-MC^2 一样，在实际中有良好的适用性，另外，定理2.12良好的揭示游走次数同聚类质量间的联系，可以看到随着 ε 的减小，游走步数 m 逐渐增大，聚类质量逐渐向 k-means++ 的质量靠拢。同时，如果对数据的假设 $\eta(\mathcal{X})$ 满足的话，我们可以得到下述推论。

推论 2.13 令 $k \in \mathbb{N}$，数据集 $\mathcal{X} \subseteq \mathbb{R}^d$，$|\mathcal{X}| = n$，且满足 $\beta(\mathcal{X}) = O(k)$，$C$ 是算法2-8的输出，$m = O(k \log k)$，则

$$\mathbb{E}[\phi(\mathcal{X}, C)] \leq 8(\log_2 k + 3)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

此时，算法中预处理花费时间为 $O(nd)$，主要循环花费时间为 $O(k^3 d \log k)$。实验结果表明，对于大数据集，在 0 到 1% 的相对误差下，AFK-MC^2 算法要比 k-means++ 快 200 到 1000 倍。

除去前述的假设 1 和 2，也有些其他假设值得注意。Ben-David [30] 通过引入聚类描述方案（clustering description scheme）并在其上添加性质，获得了常数近似系数的解，且该算法的时间复杂度是常数。文献 [30] 的主要假设叫做 α-m-covering，它要求对领域集合（domain set）中的任何子集 S，通过应用聚类描述方案到 S 的一个 l 元组上，S 的最优聚类能被 α 近似。

至此，本节梳理了一些有理论保证的算法同时在实验上对他们进行了比较，值得注意的是文献 [31, 32] 指出 k-means 问题是 APX 难的，多项式时间内不存在近似系数小于 1.0013 的解，目前最好的近似系数是 6.357 [33]。不过，近似系数越小的算法越慢，越难以实用。另外，k-means 问题可以构造 PTAS（Polynomial Time Approximation Scheme）的解，即在多项式时间内取得 1+ε 的近似，相关文献有 [34–36]。

2.3 更快得到 k-means 的解

本节分析如何加速 k-means 的求解，目前有两种方法加速，一种是减少数据量，还有一种利用三角不等式。

加速聚类的一个自然的想法是将大数据浓缩，得到小数据，在小数据上聚类。用小数据上的解作为大数据上的解。由于用于减少数据量的采样方法有很多，这是一个框架性算法，算法列在2-9中。在上一节提到的理论保证算法都可以扩展后

算法 2-9 基于减少数据量的 k-means 算法

输入: 数据集 $\mathcal{X}$，类数目 k，聚类算法 $\mathcal{A}_c$，采样算法 $\mathcal{A}_s$，采样数 m
输出: k 个点 C

$S \leftarrow \mathcal{A}_s(\mathcal{X}, m)$
$C \leftarrow \mathcal{A}_c(S, k)$
返回 C

当 $\mathcal{A}_s$，比如将 k-means++、K-MC^2、AF-KMC^2 的 seeding 数目从 k 增加到 m，使用 k-means|| 采样，将局部搜索的返回 k 个点增加到返回 m 个点，以及经典的均匀采样。虽然理论保证的算法可以扩展为采样算法，但是他们主要目的还是提供理论保证，而有一个技术是专门压缩数据量的，称为 Coreset。

通俗来讲，Coreset 指的是一些特殊的采样点（可能带权重），给定任意 k 个点，这些采样点到这 k 个点的距离平方和都能够近似所有点到这 k 个点的距离平方和，定义见2.14。

定义 2.14 (**常规** Coreset) 带权重集合 S 是数据集 $\mathcal{X}$ 一个 ε-Coreset，如果对任意集合 $C \subset \mathbb{R}^d$，$|C| = k$ 满足

$$|\phi(\mathcal{X}, C) - \psi(S, C)| \leq \varepsilon\phi(\mathcal{X}, C)$$

这是一个非常强的定义，因为它要求对所有 C 都要符合定义的约束，所以符合定义2.14的 Coreset 也被称为 Strong Coreset。对于 Strong Coreset，它的一个良好的性质是采样点上的最优距离平方和就可以近似所有点上的最优距离平方和。

推论 2.15 (Coreset **性质** 1) 令 $\varepsilon \in (0, \frac{1}{3})$，$S$ 是数据集 $\mathcal{X}$ 的一个 ε-Coreset，则有

$$\phi(\mathcal{X}, C^S_{\mathrm{OPT}_k}) \leq (1 + 3\varepsilon)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

目前效果很好的 ε-Coreset 基于重要性采样 (importance sampling)，算法见文献 [37] 中算法 9，其采样数目为 $O(\frac{k\log k}{\varepsilon^2}(dk\log k + \log\frac{1}{\delta}))$，时间复杂度为 $O(ndk\log\frac{1}{\delta})$，对于采样点来说近似系数是 $1 + O(\varepsilon)$。该方法的近似系数非常好而花费的时间又是线性的，可以说是目前 k-means 问题中相当好的算法。

虽然常规的 Coreset 如 ε-Coreset 效果好，但是花费的时间还是稍微有点多，Lightweight Coreset [38] 即是牺牲了一部分聚类的质量来换取速度的提升，具体来说 Lightweight Coreset 指的是满足定义2.16的带权点集。

定义 2.16 (Lightweight Coreset) 令 $\varepsilon > 0$，$k \in \mathbb{N}$，$\mathcal{X} \subset \mathbb{R}^d$，它中心点是 $\mu(\mathcal{X})$，如果对任意 $Q \in \{Q' | Q' \subset \mathbb{R}^d, |Q'| \leq k\}$ 有

$$|\phi(\mathcal{X}, Q) - \psi(S, Q)| \leq \frac{\varepsilon}{2}\phi(\mathcal{X}, Q) + \frac{\varepsilon}{2}\phi(\mathcal{X}, \mu(\mathcal{X}))$$

则称 S 是一个 (ε, k)-Lightweight Coreset

效仿推论2.15，易得其有以下近似系数的推论。

推论 2.17 (Lightweight Coreset **性质**) 如果 S 是一个 (ε, k)-Lightweight Coreset，有

$$\phi(\mathcal{X}, C^{S}_{\mathrm{OPT}_k}) \leq \phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k}) + 4\varepsilon\phi(\mathcal{X}, \mu(\mathcal{X}))$$

构造该 Coreset 的算法见算法2-10，该算法有如下理论保证

算法 2-10 Lightweight coreset

输入: 数据集 $\mathcal{X}$, 采样点数目 m
输出: m 个点 S

$\mu \leftarrow$ 计算 $\mathcal{X}$ 的均值
for $x \in \mathcal{X}$ **do**
  $q(x) \leftarrow \frac{1}{2}\mathrm{d}(x,\mu)^2 / \sum_{x' \in \mathcal{X}} \mathrm{d}(x',\mu)^2 + \frac{1}{2n}$
end
$S \leftarrow$ 依分布 q 从 $\mathcal{X}$ 中采样 m 个点且每个点 x 的权重是 $\frac{1}{m.q(x)}$
返回 S

定理 2.18 (Lightweight Coreset **理论保证**) 令 $\varepsilon > 0$，$\delta > 0$，且 $k \in \mathbb{N}$。令 $\mathcal{X}$ 是 $\mathbb{R}^d$ 中的数据点，令 S 是算法2-10的输出。其中采样数

$$m \geq c\frac{dk\log k + \log(1/\delta)}{\varepsilon^2}$$

其中 c 是一个常数。这样就可以至少 1-δ 的概率使得集合 S 是一个 (ε, k)-Lightweight Coreset。

该定理说明只要采样常数个点（与 n 无关），算法就可以在 $O(nd)$ 的时间复杂度里取得好的聚类结果（如果 $\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_1})/\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$ 不大的话）。特别的，如果前述假设 2 满足的话，对于采样点来说近似系数就变为了 $1 + O(k\varepsilon)$。

构造 Coreset 的方法除了这里提到的，还有很多，比如还可以用 k-means++ 构造 Coreset[39]，Coreset 作为一个经典的数据压缩范式还被用于其他机器学习问题中，比如 PCA，NNMF（Non Negative Matrix Factorization）等，文献 [40] 和 [41] 是很好的参考资料。

除了减少数据量，基于三角不等式也可以加速，由于本论文工作和该方向重合度不高，感兴趣的读者可参考文献 [42–45]。

2.4 加速 k-means 且有理论保证

很多时候我们希望算法既有理论保证也要速度快，本节我们将介绍这类算法。算法的基本思想是用前文减少数据量的方法减少数据，然后在少量数据上运行有理论保证的算法，接着证明这个解对于全部点也是有保证的。一种简单且有理论

算法 2-11 基于均匀不放回采样的 k-means 算法

输入: 数据集 $\mathcal{X}$，类数目 k，采样数 s
输出: k 个点 C

1 $S \leftarrow$ 从 $\mathcal{X}$ 中均匀采样 s 个点
2 $C \leftarrow$ 在 S 上运行一个 α 近似算法
3 **返回** C

保证的减少数据量的方式是均匀采样 [46]，算法详细描述见算法2-11，可以证明该方法有如下理论保证。

定理 2.19 (**均匀不放回采样的解的质量**) 令 $0 < \delta < 1/2, \alpha \geq 1, \beta > 0$ 是近似的参数。令 C 是由算法 2-11返回的点. 假设我们均匀不放回的采样 s 个点，使得,

$$s \geq \ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{\beta^2 m^2}{2\Delta^2\alpha^2}+\frac{\ln(1/\delta)}{n})$$

则，我们有

$$\phi(\mathcal{X}, C) \leq 4(\alpha+\beta)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

以至少 $1-2\delta$ 的概率, 其中 $\Delta = \max_{i,j} \|x_i - x_j\|^2$ 是数据直径的平方, $m = \phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})/n$ 是点到其对应的中心点的距离的平均值。

该方法优点是简单，但是该方法的采样点数目由数据点的直径决定，如果直径较大，该方法便会失去实用价值。类似的还有基于均匀放回采样的算法 [47]。

为了解决直径的问题，我们可以使用其他有理论保证的采样算法替换算法2-9中的算法 $\mathcal{A}_s$，不过替换之后为了使最终解有理论保证，需要修改算法2-9，即给采样点带上权重，权重是采样点对应的类的大小，修改后的算法见2-12。前面

说过 k-means++ 可以扩展到 seed s 个点（$s > k$），把扩展后的算法称为 k-means++ overseeding，对于 k-means++ overseeding 算法可以证明如下结论 [48]

定理 2.20 (k-means++ overseeding **解的理论保证** 1) 令 S 是 k-means++ overseed

算法 2-12 基于减少数据量的 k-means 算法 2

输入: 数据集 $\mathcal{X}$，类数目 k，聚类算法 $\mathcal{A}_c$，有理论保证的采样算法 $\mathcal{A}_s$，采样数 s

输出: k 个点 C

1 $S \leftarrow \mathcal{A}_s(\mathcal{X}, s)$

2 对点 $s_i \in S$，令 w_i 是 $\mathcal{X}$ 中离 s_i 最近的点的数目

3 $C \leftarrow$ 令 w_i 是 s_i 的权重，在带权的 S 上运行一个 α 近似算法 $\mathcal{A}_c$

4 **返回** C

出来的点集，且 $|S| = \lceil 16(k+\sqrt{k})\rceil = O(k)$，则有

$$\phi(\mathcal{X}, S) \leq 20\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

以至少 0.03 的概率成立（通过重复实验 $\log(1/\delta)$ 次取最好结果，概率可以被增加到 $1-\delta$）

该定理说的是用 k-means++ overseeding 采样 $O(k)$ 个点，所有点到采样点的距离平方和就能到常数 20，注意算法2-12的目标是常数近似系数，不然使用 k-means++ 可以直接有 $O(\log k)$ 的近似，使用 k-means++ overseeding 就会失去意义。另外，Dennis [49] 对 k-means++ overseeding 给出了一个更紧的理论界。

定理 2.21 (k-means++ overseeding **解的理论保证** 2) 令 S 是 k-means++ overseed 出来的点集，且 $|S| = \beta k$，$\beta \geq 1$，则有

$$\frac{\mathbb{E}[\phi(\mathcal{X}, S)]}{\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})} \leq 8(1 + \min\{\frac{\phi(k-2)}{(\beta-1)k+\phi}, H_{k-1}\}) - \Theta(\frac{1}{n})$$

其中 $\phi = (1+\sqrt{5})/2$ 是黄金比例，$H_k = 1 + \frac{1}{2} + ... + \frac{1}{k} \sim \log k$ 是第 k 个调和数。

通过该定理，我们容易推得在采样数目是 $O(k)$ 时，近似系数将从 20 变为 9.137。用 k-means++ overseeding 算法作为算法2-12中的 $\mathcal{A}_s$，结合定理2.20或者2.21可以证明下面的结论

推论 2.22 (**算法**2-12**理论保证** 1) 使用 k-means++ overseeding 作为算法2-12中的算法 $\mathcal{A}_s$，且采样数 $s = O(k)$，则有

$$\phi(\mathcal{X}, C) \leq O(\alpha)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

其中 C 为算法2-12返回的解，α 是算法2-12第 3 步中 α 近似算法的近似系数。

证明: 利用三角不等式，详细见文献 [48] ■

研究发现 k-means++ overseeding 也能形成 Coreset[39]，由于 Coreset 符合推论2.15的良好性质，我们可以得出如果在 Coreset 运行一个 α 近似的聚类算法，则

能获得一个在全部点上是 $\alpha(1+O(\varepsilon))$ 近似的解。

定理 2.23 (k-means++ overseeding **形成** Coreset) 令算法2-12中采样数为 s，如果 $s=\Theta\left(\frac{k\log n}{\delta^{d/2}\epsilon^d}\log^{d/2}\left(\frac{k\log n}{\delta^{d/2}\epsilon^d}\right)\right)$，且采样方法是 k-means++ overseeding，则算法2-12中带权重的 S 有至少 $1-\delta$ 的概率是 $\mathcal{X}$ 的一个 $(k,6\varepsilon)$ 的 Coreset。

根据 Coreset 的性质，我们有以下推论。

推论 2.24 (**算法**2-12**理论保证** 2) 使用 k-means++ overseeding 作为算法2-12中的算法 $\mathcal{A}_s$，且采样数 $s=\Theta\left(\frac{k\log n}{\delta^{d/2}\epsilon^d}\log^{d/2}\left(\frac{k\log n}{\delta^{d/2}\epsilon^d}\right)\right)$，则有

$$\phi(\mathcal{X},C)\leq\alpha(1+O(\varepsilon))\phi(\mathcal{X},C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

其中 C 为算法2-12返回的解，α 是算法2-12第 3 步中 α 近似算法的近似系数。

需要注意的是形成 Coreset 需要采样更多的点，采样数从 $O(k)$ 增加到了 $O(\log n)$，所以时间复杂度也增加了，但是相应的常数的系数也变小了，得到了收获。由于已经有常数近似的算法，我们自然希望能在保持这一近似的情况下让算法尽量的快。那么，是否存在次线性时间的常数近似的算法呢？遗憾的是，文献 [50] 告诉我们一般情况下不存在这样的算法，除非给数据加一些假设。在下一节，我们将展示如何通过一些数据假设让算法2-11能在次线性时间，具体来说是多项式对数时间内，获得常数近似系数的解。

2.5 基于均匀采样算法的新理论结果

在上一节，我们看到基于均匀采样的算法2-11能够有常数近似系数，然而该算法的采样点可能由于数据直径大而很多。本节我们首先证明算法2-11可以取得更加紧的理论界，同时利用数据假设让采样点数目在多项式对数级别，从而在多项式对数时间内，获得常数近似系数的解。

2.5.1 基于均匀采样的更紧理论界

我们首先给出结果再证明

定理 2.25 (**均匀不放回采样的解的质量** 2) 令 $0<\delta<1/2,\alpha\geq1,\beta>0$ 是近似的参数。令 C 是由算法 2-11返回的点. 假设我们均匀不放回的采样 s 个点，使得,

$$s\geq\ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{\beta^2m^2}{2\Delta^2\alpha^2}+\frac{\ln(1/\delta)}{n})$$

则，我们有

$$\phi(\mathcal{X},C)\leq(\alpha+\beta)\phi(\mathcal{X},C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

以至少 $1-2\delta$ 的概率, 其中 $\Delta = \max_{i,j} \|x_i - x_j\|^2$ 是数据直径的平方, $m = \phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})/n$ 是点到其对应的中心点的距离的平均值。

为了证明定理2.25，我们需要一些工具，首先介绍一个 concentration bound，名叫 Serfling bound，concentration bound 描述的是随机变量同其均值的联系，更多细节在附录中描述。

引理 2.26 (Serfling bound) 记均匀不放回采样得到的随机变量是 $X_1 \ldots X_s$，其中 $0 \le X_i \le \Delta$。则有

$$P[\sum_{i=1}^{s} \frac{X_i - \mathbb{E}[X_i]}{s} \ge \gamma] \le \exp(-\frac{2s\gamma^2}{(1-(s-1)/n)\Delta^2})$$

成立，其中 γ, Δ 是任意大于 0 的数字。

接着引入下面的定义，该定义是定义2.1的延伸。可以这样理解这个定义，当我们用减少数据的方法加速聚类的时候，误差来自与两方面，一方面是聚类算法在采样点上聚类产生，由 α 量化，另一方面由采样算法产生，由 β 量化。

定义 2.27 (k-means **问题解的质量** 2) 令 $\alpha \ge 1$，$\beta > 0$，一个大小为 k 的集合 C 称为 k-means 的一个 β-bad α-approximation 的解，如果有

$$\phi(\mathcal{X}, C) > (\alpha + \beta)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

否则，C 称为一个 β-good α-approximation 的解。

有了这些工具后，我们证明两个关键引理

引理 2.28 令 S 是算法2-11的采样点，且

$$s \ge \ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{2\beta^2 m^2}{\Delta^2\alpha^2} + \frac{\ln(1/\delta)}{n})$$

记 C 是算法2-11返回的解，则我们有

$$\phi(S, C) \le (\alpha + \beta)sm$$

以至少 1-δ 的概率成立，其中 Δ 和 m 的定义同定理2.19中的一样。

证明: 记 $X_i = \mathrm{d}(v_i, C^{V}_{\mathrm{OPT}_k})^2$，其中 $v_i \in S$。则，$\phi(S, C^{\mathcal{X}}_{\mathrm{OPT}_k}) = \sum_{i=1}^{s} X_i$。注意到

$\mathbb{E}[X_i] = m$ 且 $X_i \leq \Delta$，我们有

$$P[\phi(S, C^{\mathcal{X}}_{\mathrm{OPT}_k}) > (1+\frac{\beta}{\alpha})\frac{s}{n}\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})] \tag{2-1}$$

$$= P[\sum X_i > (1+\frac{\beta}{\alpha})sm)] \tag{2-2}$$

$$\leq \exp(-\frac{2s\cdot\frac{\beta^2}{\alpha^2}\cdot m^2}{(1-s/n)\Delta^2}) \tag{2-3}$$

根据 Serfling bound，最后一步成立，令 $\exp(-\frac{2s\cdot\frac{\beta^2}{\alpha^2}\cdot m^2}{(1-s/n)\Delta^2}) \leq \delta$，得到

$$s \geq \ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{2\beta^2 m^2}{\Delta^2\alpha^2}+\frac{\ln(1/\delta)}{n})$$

由于在 S 上运行一个 α 近似的算法，我们有式 (2-4) 成立

$$\phi(S, C) \leq \alpha\phi(S, C^{S}_{\mathrm{OPT}_k}) \tag{2-4}$$

$$\leq \alpha\phi(S, C^{\mathcal{X}}_{\mathrm{OPT}_k}) \tag{2-5}$$

$$\leq (\alpha+\beta)sm \tag{2-6}$$

式 (2-5) 成立是因为 $C^{S}_{\mathrm{OPT}_k}$ 是带权重的 S 上最优的 k 个点，自然比任意其他解要好，才有了 $\phi(S, C^{S}_{\mathrm{OPT}_k}) \leq \phi(S, C^{\mathcal{X}}_{\mathrm{OPT}_k})$，将式 (2-1) 带入式 (2-5) 得到式 (2-6)。 ■

引理 2.29 令 S 是算法2-11中的采样点，且

$$s \geq \ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{2\beta^2 m^2}{\Delta^2}+\frac{\ln(1/\delta)}{n})$$

记 V 是 k-means 问题 2β-bad α-approximation 的解集，C 是任意 k 个点，则

$$P[\phi(S, C) \leq (\alpha+\beta)sm | C \in V] \leq \delta$$

其中 Δ 和 m 的定义同定理2.19中的一样。

证明: 令 $Y_i = \mathrm{d}(v_i, C)^2$，其中 $v_i \in S$，给定 $C \in V$，我们有 $\mathbb{E}[Y_i] > (\alpha+2\beta)m$。

故,

$$
\begin{aligned}
&P[\phi(S,C) \le (\alpha+\beta)sm | C \in V] \\
&\le P[(\sum_{i=1}^{s} Y_i) \le \frac{\alpha+\beta}{\alpha+2\beta} s\mathbb{E}[Y_i]] \\
&\le P[(\sum_{i=1}^{s} Y_i) \le (1-\frac{\beta}{\alpha+2\beta}) s\mathbb{E}[Y_i]] \\
&\le P[(\sum_{i=1}^{s} \frac{Y_i - \mathbb{E}[Y_i]}{s}) \le (-\frac{\beta}{\alpha+2\beta})\mathbb{E}[Y_i]] \\
&\le \exp(-\frac{2s(\frac{\beta}{\alpha+2\beta}\mathbb{E}[Y_i])^2}{(1-(s-1)/n)\Delta^2}) \\
&\le \exp(-\frac{2s\beta^2 m^2}{(1-(s-1)/n)\Delta^2})
\end{aligned}
$$

令 $\exp(-\frac{2s\beta^2m^2}{(1-(s-1)/n)\Delta^2}) \le \delta$，我们有

$$
s \ge \ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{2\beta^2m^2}{\Delta^2} + \frac{\ln(1/\delta)}{n})
$$

引理得证。 ■

有了引理2.28和2.29，定理2.25证明如下。

证明: 令 C 是算法2-11的返回值，且采样数目

$$
s \ge \ln(\frac{1}{\delta})(1+\frac{1}{n})/(\frac{2\beta^2m^2}{\Delta^2\alpha^2} + \frac{\ln(1/\delta)}{n})
$$

根据引理2.28，我们有

$$
\begin{aligned}
&P[\phi(S,C) \le (\alpha+\beta)sm] \\
&= P[\phi(S,C) \le (\alpha+\beta)sm \quad \text{and} \quad C \in V] && \text{(2-7)} \\
&\quad + P[\phi(S,C) \le (\alpha+\beta)sm \quad \text{and} \quad C \in \bar{V}] && \text{(2-8)} \\
&\ge 1-\delta && \text{(2-9)}
\end{aligned}
$$

其中 $\bar{V}$ 是 k-means 问题的 2β-good α-approximation 的解集。根据引理2.29，我们知

道

$$\begin{aligned}
&P[\phi(S,C) \le (\alpha+\beta)sm \quad \text{and} \quad C \in V] \\
&= P[\phi(S,C) \le (\alpha+\beta)sm | C \in V] \\
&\qquad \cdot P[C \in V] \\
&\le \delta
\end{aligned}$$

因此，

$$\begin{aligned}
P[C \in \bar{V}] &\ge P[\phi(S,C) \le (\alpha+\beta)sm \quad \text{and} \quad C \in \bar{V}] \\
&\ge 1 - 2\delta
\end{aligned}$$

用 $\frac{1}{2}\beta$ 替换 β，定理得证。 ■

2.5.2 加速均匀采样

上面我们证明了算法2-11可以有更紧的理论界，这里我们在一定的数据假设下，具体来说就是前述的假设 1 和 2，证明这个算法可以很快，我们先给出下面的结论。

定理 2.30 令 $0 < \delta < 1/2$，$\alpha \ge 1$，$\beta > 0$。假定假设 1 和 2 成立，令 C 是算法2-11返回的解，我们有

$$\phi(\mathcal{X}, C) \le (\alpha+\beta)\phi(\mathcal{X}, C^{\mathcal{X}}_{\mathrm{OPT}_k})$$

以至少 $1-2\delta$ 的概率成立，如果采样数 $s = O(\ln(\frac{1}{\delta})\frac{\alpha^2}{\beta^2}k^2 \log^4 n)$

这个定理说明，如果在采样点上运行一个多项式时间复杂度常数近似系数的算法，我们就能在多项式对数时间复杂度下获得对于所有点来说常数近似系数的解。为了证明定理2.30，我们先证明一个引理

引理 2.31 如果假设 1 和 2 成立，可以知道

$$\frac{\Delta}{m} = O(k \log^2 n)$$

其中 Δ 和 m 的定义同定理2.19中的一样。

证明: 显然，

$$\max_{x \in \mathcal{X}} \mathrm{d}(x, \mu(\mathcal{X}))^2 \ge \frac{1}{4}\Delta$$

因此，

$$\frac{\max_{x\in\mathcal{X}} \mathrm{d}(x,\mu(\mathcal{X}))^2}{\frac{1}{n}\sum_{x'\in\mathcal{X}} \mathrm{d}(x',\mu(\mathcal{X}))^2}\frac{\phi(\mathcal{X},C^{\mathcal{X}}_{\mathrm{OPT}_1})}{\phi(\mathcal{X},C^{\mathcal{X}}_{\mathrm{OPT}_k})} \geq \frac{1}{4}\frac{\Delta}{m}$$

根据推论2.9和2.10

$$\zeta(\mathcal{X})\eta(\mathcal{X}) = O(k\log^2 n)$$

引理得证。 ■

现在我们证明定理2.30

证明: 由定理2.25，当 n 趋向无穷大时,

$$s \geq \ln(\frac{1}{\delta})\frac{2\Delta^2\alpha^2}{\beta^2 m^2}. \tag{2-10}$$

根据引理2.31，

$$\frac{\Delta}{m} = O(k\log^2 n) \tag{2-11}$$

结合定理2.25，式 (2-10) 和 (2-11)，命题得证。 ■

2.6 Weighted k-means++

在前面的算法2-6和2-12中，都需要解决带权 k-means 问题，本小节我们将 k-means++ 算法扩展到了带权重 k-means 问题上，并证明了这一算法有 $O(\log k)$ 的近似系数。首先，带权 k-means++ 算法见算法2-13。其次，由于推导复杂，本节我

算法 2-13 weighted k-means++ seeding

输入: 数据集 $\mathcal{X}$，数据点的权重 w，类数目 k
输出: k 个点 C

$c_1 \leftarrow$ 以概率 $\frac{w_x}{\sum_{i\in\mathcal{X}} w_i}$ 从 $\mathcal{X}$ 中采样一个点 x
$C \leftarrow \{c_1\}$
for $i = 2, 3, \ldots k$ **do**
  for $x \in \mathcal{X}$ **do**
    $p(x) \leftarrow w_x \mathrm{d}(x,C)^2 / \sum_{x'\in\mathcal{X}} w'_x \mathrm{d}(x',C)^2$
  end
  $x \leftarrow$ 从 $\mathcal{X}$ 中依分布 $p(x)$ 采样一个点
  $C \leftarrow C \cup \{x\}$
end
返回 C

们简化数学符号，简记 $\psi(\mathcal{X}, C)$ 为 $\psi_C(\mathcal{X})$，简记 $\psi(\mathcal{X}, C_{\mathrm{OPT}_k}^{\mathcal{X},\psi})$ 为 $\psi_{\mathrm{OPT}}(\mathcal{X})$，$C_{\mathrm{OPT}_k}^{\mathcal{X},\psi}$ 是使得 $\psi(\mathcal{X}, C)$ 最小的最优解，ψ 的含义参见定义2.7。给定这些记号，对于算法2-13，我们可以推出如下定理。

定理 2.32 对于任意数据集 $\mathcal{X}$，及对应的权重 w

$$\mathbb{E}[\psi_C(\mathcal{X})] \leq 8(\ln k + 2)\psi_{\mathrm{OPT}}(\mathcal{X})$$

其中 C 是算法2-13的返回结果。

为了证明该定理，我们引入以下引理。

引理 2.33 令 $X \in \mathbb{R}^d$ 是一个随机变量，对于任意 $z \in \mathbb{R}^d$，有

$$\mathbb{E}[\|X - z\|^2] = \mathbb{E}[\|X - \mathbb{E}X\|^2] + \|z - \mathbb{E}X\|^2 \tag{2-12}$$

证明: 见文献 [51] 中的引理 2。 ■

记 $\psi_C(\mathcal{X})$ 最优解对应的 k 个类为 $A_1, A_2, ..., A_k$，我们把这些类称为最优类。

推论 2.34 对于任意一个最优类 A，给定任意一个点 $z \in \mathbb{R}^d$，令 μ 是 A 的带权中心点，可得

$$\psi_z(A) = \psi_{\mathrm{OPT}}(A) + (\sum_{i \in A} w_i) \|z - \mu\|^2$$

证明: 令 X 是在类 A 中依照概率 $P[X = a] = \frac{w_a}{\sum_{i \in A} w_i}$ 采样的点，a 是 X 可能取的值。

$$\mathbb{E}[\|X - z\|^2] = \sum_{a \in A} \frac{w_a}{\sum_{i \in A} w_i} \|a - z\|^2 = \frac{1}{\sum_{i \in A} w_i} \psi_z(A) \tag{2-13}$$

$$\mathbb{E}[\|X - \mathbb{E}X\|^2] = \sum_{a \in A} \frac{w_a}{\sum_{i \in A} w_i} \|a - \mu\|^2 = \frac{1}{\sum_{i \in A} w_i} \psi_{\mathrm{OPT}}(A) \tag{2-14}$$

$$\|z - \mathbb{E}X\|^2 = \|z - \mu\|^2 \tag{2-15}$$

在等式2-12左右两边同乘 $\sum_{i \in A} w_i$，结合式2-13，2-14和2-15可得。 ■

证明的基本思路是首先考察均匀采样一个点，这个点所在的类到这个点产生的距离平方和同这个类自身最优距离平方和的关系。然后考察基于 d^2 weighting 采样产生的一个点，这个点所在的类到这个点产生的距离平方和同这个类自身最优距离平方和的关系。给定 k-means++ 的中间结果 C，基于 C 任选 u 个没有被覆盖的最优类记为 $\mathcal{X}_u$，其他的最优类记为 $\mathcal{X}_c$。基于 C 用 d^2 weighting 采样 t 个点，把这些点和 C 放在一起记为 C'。最后，我们考察 $\psi_{C'}(\mathcal{X})$ 同 $\psi_C(\mathcal{X}_c)$，$\psi_C(\mathcal{X}_u)$，$\psi_{\mathrm{OPT}}(\mathcal{X}_c)$，$\psi_{\mathrm{OPT}}(\mathcal{X}_u)$，$t$，$u$ 的关系。然后，利用这一关系证明定理2.32。

引理 2.35 令 A 是任意一个最优类，记 Z 是从 A 中依照 $P[Z=z]=\frac{w_z}{\sum_{i\in A}w_i}$ 概率分布采样的点，z 是 Z 可能取的值，我们有

$$\mathbb{E}[\psi_Z(A)]=2\psi_{\mathrm{OPT}}(A)$$

证明: 根据期望的计算方式有，将上面的推论2.34带入得到

$$\begin{aligned}\mathbb{E}[\psi_Z(A)]&=\sum_{z\in A}\frac{w_z}{\sum_{i\in A}w_i}\psi_z(A)=\sum_{z\in A}\frac{w_z}{\sum_{i\in A}w_i}(\psi_{\mathrm{OPT}}(A)+(\sum_{i\in A}w_i)\left\|z-\mu\right\|^2)\\&=2\psi_{\mathrm{OPT}}(A)\end{aligned}$$

引理得证 ■

这个引理说明均匀采样一个点，点所在的类到这个点的距离平方和不会比最优的差 2 倍。换句话说，如果每次都能选择到不同的类，在这个类里面均匀采样的话最后的近似系数就是 2，可惜我们没有办法让每次 seed 的点落在不同的类，所以才提出了 k-means++ 算法。那么基于 d^2 weighting 采样产生的点，点所在的类到这个点的距离平方和不会比最优的差几倍呢？我们用下个引理回答这个问题。

引理 2.36 令 C 是 k-means++ seeding 的任意中间结果，$1\leq|C|\leq k$，记 $Z\in A$ 是在给定 C 的情况下依照 d^2 weighting 采样的点，A 是任意一个最优类，记 $C'=C\cup\{Z\}$，则可以得到

$$\mathbb{E}[\psi_{C'}(A)|C,Z\in A]\leq 8\psi_{\mathrm{OPT}}(A)$$

证明: 根据期望计算方式有

$$\mathbb{E}[\psi_{C'}(A)|C,Z\in A]=\sum_{z\in A}P[Z=z|C,Z\in A]\psi_{C\cup\{z\}}(A)=\sum_{z\in A}\frac{\psi_C(z)}{\psi_C(A)}\psi_{C\cup\{z\}}(A)$$

对任意数据点 z 和 i，根据三角不等式，我们有下面第一行

$$\begin{aligned}d(z,C)&\leq d(i,C)+\left\|i-z\right\|\\d(z,C)^2&\leq 2d(i,C)^2+2\left\|i-z\right\|^2\\(\sum_{i\in A}w_i)d(z,C)^2&\leq 2\psi_C(A)+2\psi_z(A)\\\psi_C(z)&\leq 2\frac{w_z}{\sum_{i\in A}w_i}\psi_C(A)+2\frac{w_z}{\sum_{i\in A}w_i}\psi_z(A)\end{aligned}$$

上面第二行是根据算术平均数小于等于平方平均数得到，对第二行两边同乘 w_i，把所有 $i\in A$ 里的情况加到一起得到第三行，两边同乘 w_z，再除以 $\sum_{i\in A}w_i$ 得到最

后一行。将最后一行的 $\psi_C(z)$ 带入一开始的期望计算中，得到

$$\mathbb{E}[\psi_{C'}(A)|C, Z \in A] \leq \sum_{z\in A} \frac{2\frac{w_z}{\sum_{i\in A} w_i}\psi_C(A)}{\psi_C(A)}\psi_{C\cup\{z\}}(A) + \sum_{z\in A} \frac{2\frac{w_z}{\sum_{i\in A} w_i}\psi_z(A)}{\psi_C(A)}\psi_{C\cup\{z\}}(A)$$

注意到 $\psi_{C\cup\{z\}}(A) \leq \min[\psi_C(A), \psi_z(A)]$，对于上式第一项取 $\psi_{C\cup\{z\}}(A) \leq \psi_z(A)$，对于第二项取 $\psi_{C\cup\{z\}}(A) \leq \psi_C(A)$，带入上式，得

$$\begin{aligned}\mathbb{E}[\psi_{C'}(A)|C, Z \in A] &\leq \sum_{z\in A} 2\frac{w_z}{\sum_{i\in A} w_i}\psi_z(A) + \sum_{z\in A} 2\frac{w_z}{\sum_{i\in A} w_i}\psi_z(A) \\ &\leq 4\mathbb{E}[\psi_Z(A)] \\ &\leq 8\psi_{\mathrm{OPT}}(A)\end{aligned}$$

上式第二、三行是将引理2.35带入得到，引理得证。 ■

这个引理说明如果我们每次用 d^2 weighting 选到的点都在不同的最优类，那最后的近似系数就是 8，但是我们没办法保证这一点。如果一个最优类里至少一个点被 d^2 weighting 选到了，我们称这个类“被覆盖”了。如果一个类已经被覆盖了，它有可能会再次被覆盖，这样就会浪费 d^2 weighting 的机会，从而导致近似系数大于 8。这里，我们考虑一个一般的命题，给定 k-means++ seeding 的任意中间结果 C，基于 C 我们任意选择 u 个没有被覆盖的最优类记为 $\mathcal{X}_u$，其余的类记为 $\mathcal{X}_c$，如果在 C 的基础上用 d^2 weighting 采样 t 个点，记 C' 是在 C 上加入这 t 个点后的结果，那么所有点到 C' 的距离平方和同 $\psi_C(\mathcal{X}_c)$ 及 $\psi_C(\mathcal{X}_u)$ 有什么关系吗？接下来我们给出一种可能的关系，然后用数学归纳法证明它。

引理 2.37 令 C 是 k-means++ seeding 的任意中间结果，$1 \leq |C| \leq k$，给定 C，任选 $u > 0$ 个没有被覆盖的最优类记为 $\mathcal{X}_u$，其余的类记为 $\mathcal{X}_c$，在 C 的基础上用 d^2 weighting 采样 t 个点，令 $0 \leq t \leq u$，记 C' 是在 C 上加入这 t 个点后的结果，则

$$\mathbb{E}[\psi_{C'}(\mathcal{X})|C] \leq [\psi_C(\mathcal{X}_c) + 8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1 + H_t) + \frac{u-t}{u}\psi_C(\mathcal{X}_u) \tag{2-16}$$

其中 $H_t = 1 + \frac{1}{2} + ... + \frac{1}{t}$ 是调和数。

证明: 考虑数学归纳法，首先证明当 $t = 0, u > 0$ 和 $t = 1, u = 1$ 时，式2-16成立，然后证明如果 $(t-1, u)$ 和 $(t-1, u-1)$ 有式2-16成立，(t, u) 时式2-16也能成立，这样命题得证。考虑 $t = 0, u > 0$ 的情况，由于 $1 + H_t = \frac{u-t}{u} = 1$，$\psi_{C'}(\mathcal{X}) = \psi_C(\mathcal{X}) < \psi_C(\mathcal{X}) + 8\psi_{\mathrm{OPT}}(\mathcal{X}_u)$ 式2-16成立，当 $t = 1, u = 1$ 时，记选择的点为 Z，如果 $Z \in \mathcal{X}_c$，$\mathbb{E}[\psi_{C'}(\mathcal{X})|Z \in \mathcal{X}_c] \leq \psi_C(\mathcal{X})$，如果 $Z \in \mathcal{X}_u$，根据引理2.36，

$\mathbb{E}[\psi_{C'}(\mathcal{X})|Z\in\mathcal{X}_u]\le\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)$，那么

$$\begin{aligned}\mathbb{E}[\psi_{C'}(\mathcal{X})] &= P[Z\in\mathcal{X}_c]\mathbb{E}[\psi_{C'}(\mathcal{X})|Z\in\mathcal{X}_c]+P[Z\in\mathcal{X}_u]\mathbb{E}[\psi_{C'}(\mathcal{X})|Z\in\mathcal{X}_u]\\ &\le \frac{\psi_C(\mathcal{X}_c)}{\psi_C(\mathcal{X})}\psi_C(\mathcal{X})+\frac{\psi_C(\mathcal{X}_u)}{\psi_C(\mathcal{X})}[\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)]\\ &\le 2\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)\end{aligned}$$

由于此时 $1+H_t=2$，式2-16成立。考虑 (t,u) 时的情况，记选择的第一个点叫 Z（一共要选 t 个点），$C''=C\cup\{Z\}$。第一种情况，$Z\in\mathcal{X}_c$，此时，任选 u 个没有被覆盖的类，记为 $\mathcal{X}_u''$，取 $\mathcal{X}_u''=\mathcal{X}_u$，则其他类 $\mathcal{X}_c''=\mathcal{X}_c$，还需要选 $t-1$ 个点，由于式2-16对 $(t-1,u)$ 成立，我们有

$$\begin{aligned}\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_c] &\le [\psi_{C''}(\mathcal{X}_c'')+8\psi_{\mathrm{OPT}}(\mathcal{X}_u'')](1+H_{t-1})+\frac{u-t+1}{u}\psi_{C''}(\mathcal{X}_u'')\\ &\le [\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1+H_{t-1})+\frac{u-t+1}{u}\psi_C(\mathcal{X}_u)\end{aligned}$$

第二种情况，$Z\in\mathcal{X}_u$，此时，假设被选中的类是 $A\in\mathcal{X}_u$，并令 Z 取到的值是 a。任选 $u-1$ 个没有被覆盖的类，记为 $\mathcal{X}_u''$，取 $\mathcal{X}_u''=\mathcal{X}_u-A$，对应的其他类 $\mathcal{X}_c''=\mathcal{X}_c+A$，还需要选 $t-1$ 个点，由于式2-16对 $(t-1,u-1)$ 成立，且注意到 $\psi_{C''}(\mathcal{X}_c'')=\psi_{C''}(\mathcal{X}_c+A)=\psi_{C''}(\mathcal{X}_c)+\psi_{C''}(A)\le\psi_C(\mathcal{X}_c)+\psi_a(A)$ 和 $\psi_{C''}(\mathcal{X}_u'')\le\psi_C(\mathcal{X}_u-A)$ 我们有

$$\begin{aligned}&\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u,Z\in A,Z=a]\le[\psi_{C''}(\mathcal{X}_c'')+8\psi_{\mathrm{OPT}}(\mathcal{X}_u'')](1+H_{t-1})+\frac{u-t}{u-1}\psi_{C''}(\mathcal{X}_u'')\\ &\le[\psi_C(\mathcal{X}_c)+\psi_a(A)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)-8\psi_{\mathrm{OPT}}(A)](1+H_{t-1})+\frac{u-t}{u-1}[\psi_C(\mathcal{X}_u)-\psi_C(A)]\end{aligned}$$

根据期望计算方式 (全期望公式)，有

$$\begin{aligned}\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u,Z\in A] &= \sum_{a\in A}P[Z=a|C,Z\in A]\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u,Z\in A,Z=a]\\ &\le[\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1+H_{t-1})+\frac{u-t}{u-1}[\psi_C(\mathcal{X}_u)-\psi_C(A)]\end{aligned}$$

上式用到了引理2.36，接着考虑 $\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u]$

$$\begin{aligned}\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u] &= \sum_{A\in\mathcal{X}_u}P[Z\in A|C,Z\in\mathcal{X}_u]\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u,Z\in A]\\ &= \sum_{A\in\mathcal{X}_u}\frac{\psi_C(A)}{\psi_C(\mathcal{X}_u)}\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u,Z\in A]\\ &\le[\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1+H_{t-1})+\frac{u-t}{u}\psi_C(\mathcal{X}_u)\end{aligned}$$

上式最后一行用到了算术平均数小于等于平方平均数的结论[①]，最后计算 $\mathbb{E}[\psi_{C'}(\mathcal{X})|C]$

$$\begin{aligned}\mathbb{E}[\psi_{C'}(\mathcal{X})|C] &= P[Z\in\mathcal{X}_c|C]\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_c]+P[Z\in\mathcal{X}_u|C]\mathbb{E}[\psi_{C'}(\mathcal{X})|C,Z\in\mathcal{X}_u]\\ &\le [\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1+H_{t-1})+\frac{u-t}{u}\psi_C(\mathcal{X}_u)+\frac{1}{u}\frac{\psi_C(\mathcal{X}_c)}{\psi_C(\mathcal{X})}\psi_C(\mathcal{X}_u)\\ &\le [\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1+H_{t-1}+\frac{1}{u})+\frac{u-t}{u}\psi_C(\mathcal{X}_u)\\ &\le [\psi_C(\mathcal{X}_c)+8\psi_{\mathrm{OPT}}(\mathcal{X}_u)](1+H_t)+\frac{u-t}{u}\psi_C(\mathcal{X}_u)\end{aligned}$$

最后一行成立是因为 $\frac{1}{u}\le\frac{1}{t}$，根据归纳性，引理得证。 ■

现在，我们可以证明定理2.32了，证明如下

证明: 考虑引理2.37，取第一个带权均匀采样点的集合作为 C，记这个采样点是 Z，令 A 是任意最优类，$a\in A$ 是 Z 可能取的值。取 $t=u=k-1$，则 $\mathcal{X}_u=X-A$，$\mathcal{X}_c=A$。根据引理2.37，我们有

$$\mathbb{E}[\psi_{C'}(\mathcal{X})|Z=a]\le[\psi_a(A)+8\psi_{\mathrm{OPT}}(\mathcal{X})-8\psi_{\mathrm{OPT}}(A)](1+H_{k-1})$$

根据引理2.35和 $H_{k-1}\le 1+\ln k$，可得

$$\begin{aligned}\mathbb{E}[\psi_{C'}(\mathcal{X})] &= \sum_{a\in A}P[Z=a]\mathbb{E}[\psi_{C'}(\mathcal{X})|Z=a]\\ &\le 8(\ln k+2)\psi_{\mathrm{OPT}}(\mathcal{X})\end{aligned}$$

定理得证。 ■

2.7 实验

在“基于均匀采样算法的新理论结果”一节中，我们证明了基于均匀不放回采样的算法2-11有更紧的理论界且在一定的数据假设下可以在多项式对数时间复杂度下取得常数近似的解。在本节，我们设计实验验证这一点，接受测试的基准算法分别是 K-MC^2(算法2-7) 和我们提出的一个新加速算法 Double-K-MC^2，这一新算法描述见算法2-14，它的定位是在聚类质量和时间复杂度上位于均匀不放回采样和 K-MC^2 采样之间。这一算法提出的初衷是为了估算算法2-12中点 s_i 上的权重 w_i。容易知道计算准确权重需要 $O(nsd)$ 的时间复杂度，大数据下时间复杂度太高，因此，新想法是利用一种采样算法，比如 K-MC^2 来估算权重，减少时间开销，w_i+1 是为了解决有的 w_i 可能是 0 的问题，Double-K-MC^2 中采样部分的时间复杂

① 对任意实数 $a_1,a_2,...,a_m$，有 $\sum a_i^2\ge\frac{1}{m}(\sum a_i)^2$

度是 $O(us^2d)$，而用 K-MC^2 做基准是因为 K-MC^2 也用到了前述的假设 1 和 2。这

算法 2-14 Double-K-MC^2

输入: 数据集 $\mathcal{X}$，采样数目 s，游走次数 u，聚类算法 $\mathcal{A}_c$，类数目 k
输出: k 个点 C

$S_1 \leftarrow$ 从 $\mathcal{X}$ 中用 K-MC^2 采样 s 个点
$\mathcal{X}' \leftarrow$ 从 $\mathcal{X}$ 中移除 S_1
$S_2 \leftarrow$ 从 $\mathcal{X}'$ 中用 K-MC^2 采样 s 个点
对 S_1 中任意点 s_i，令 w_i 是 S_2 中离 s_i 最近的点的数目
$C \leftarrow$ 令 $w_i + 1$ 是 s_i 的权重，在带权的 S_1 上运行一个 α 近似的算法 $\mathcal{A}_c$
返回 C

三个算法将会在八个传统聚类数据集上测试，验证他们的聚类质量和效率。另外，我们也用一个重要的实际应用-图像分割-来测试他们。在这两个任务上，结果都显示基于均匀不放回采样的算法和 Double-K-MC^2 比 K-MC^2 有更好的聚类质量而花费的时间相比 K-MC^2 又不会多太多。

2.7.1 传统聚类

表 2-1 数据量 n, 类数目 k, 维度 d

数据集	n	k	d
a2	5250	35	2
a3	7500	50	2
b2-random-10	10000	100	2
b2-random-15	15000	100	2
b2-random-20	20000	100	2
KDD	145751	200	74
RNA	488565	200	8
Poker Hand	1000000	200	10

在这个任务中，我们使用五个高斯分布的合成数据集和三个大型的真实数据集（最大的数据量有 100 万），合成数据集[①]是 **a2**, **a3** ([52]), **b2-random-10**, **b2-random-15**, **b2-random-20** ([53])，真实数据集是 **KDD**[②]([54]), **RNA**[③]([55]), **Poker Hand**[④]([56]) 数据集的描述参见表2-1。聚类质量用 k-means 目标函数值表示，由于代码实现会影响程序性能，所以效率/时间开销不用多少秒来评价，用计算了多

① 下载地址 http://cs.uef.fi/sipu/datasets/
② 下载地址 http://osmot.cs.cornell.edu/kddcup/datasets.html
③ 下载地址 https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#cod-rna
④ 下载地址 https://archive.ics.uci.edu/ml/datasets/Poker+Hand

少次距离来表示。

算法 K-MC^2, Double-K-MC^2 和算法2-11的参数如下：Double-K-MC^2 和 K-MC^2 的游走次数 $u = 200$，Double-K-MC^2 采样量 $s = 1.5 * \log^2 n$，算法2-11的采样量 $s' = 0.7 * \log^4 n$，α 近似算法是 Weighted k-means++ 接上 Lloyd（在算法2-11中 α 近似算法的权重设为 1）。所有数据集的特征都做了标准化（standardize），由于这些算法都有随机性，实验将在每个数据集上重复 40 次然后汇报平均值。 结果见

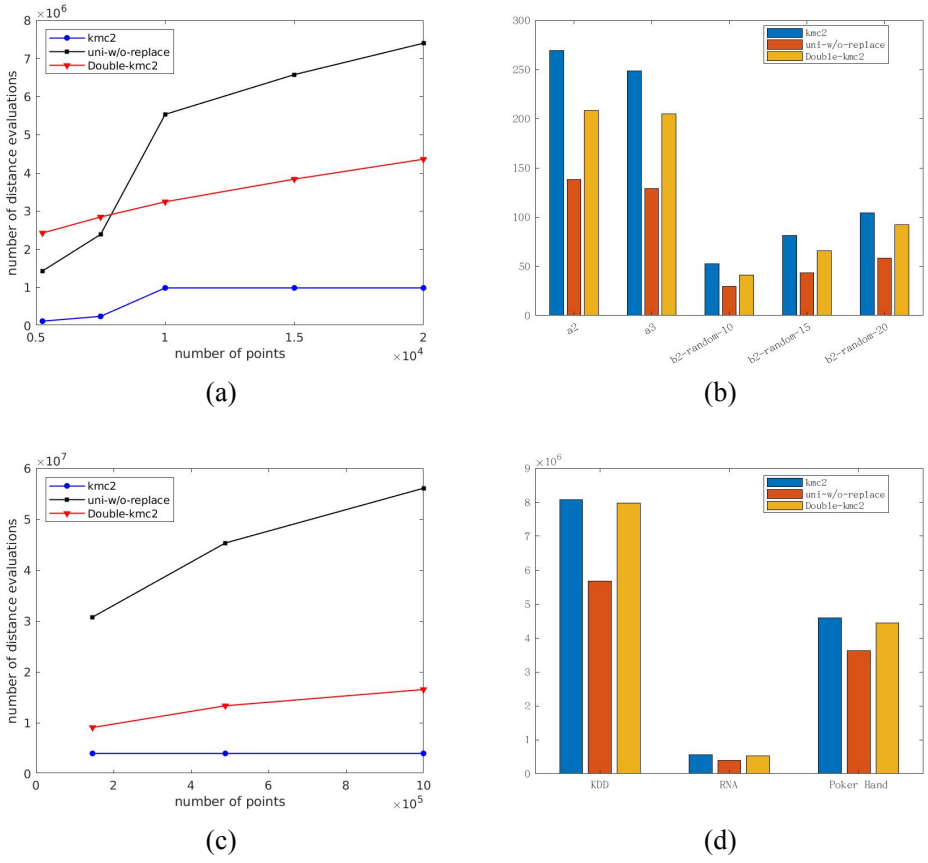


图 2-1 传统聚类任务的结果。（a）合成数据集上的距离计算次数；（b）合成数据集上的 k-means 目标函数值；（c）真实数据集上的距离计算次数；（d）真实数据集上的 k-means 目标函数值

图2-1，在三个算法中基于均匀不放回采样的算法有最好的聚类效果而它的时间开销也最大，差不多是 K-MC^2 时间开销的 10 倍，不过随着数据量的增加，时间增加缓慢。基于均匀不放回采样的算法的 k-means 目标函数值差不多是 K-MC^2 的 60%。有意思的是，Double-K-MC^2 的结果说明，通过 K-MC^2 采样一部分点，相比于单纯 K-MC^2，我们能获得更好的聚类质量，而 Double-K-MC^2 的时间花费比均匀不放回要小很多。这说明，如果你倾向于一个好的聚类质量且时间开销也不想

太大的话，一个好的选择是 Double-K-MC^2，如果追求聚类质量就使用均匀不放回采样。受到特征数目和类数目的影响，*k*-means 目标函数值不会随着数据量的增大而增大。图2-1中的数据列在表2-2中，表中真实数据集上 *k*-means 目标函数值的数量级是 10^6，合成数据集和真实数据集的时间开销分别在 10^6 和 10^7 数量级。

表 2-2 传统聚类任务上 *k*-means 目标函数值和距离计算次数（在括号中）

数据集	K-MC^2	Double-K-MC^2	基于均匀不放回采样的算法
a2	269.331(0.119)	208.395(2.428)	138.449(1.434)
a3	248.691(0.245)	204.705(2.847)	129.201(2.391)
b2-random-10	52.358(0.990)	41.002(3.245)	29.519(5.536)
b2-random-15	81.324(0.990)	65.760(3.839)	43.660(6.576)
b2-random-20	104.458(0.990)	92.411(4.361)	58.267(7.400)
KDD	8.078(0.398)	7.978(0.908)	5.680(3.076)
RNA	0.551(0.398)	0.533(1.334)	0.391(4.532)
Poker Hand	4.596(0.398)	4.446(1.653)	3.624(5.608)

2.7.2 图像分割

聚类的一个应用就是图像分割，这个应用要求对图像的不同的区块进行准确的、光滑的划分，这一应用给聚类算法聚类的质量提供了一个直观的展示。由于图像不同区块的边界不是线性的，所以需要修改 *k*-means 的目标函数（*k*-means 划分的边界是线性的），这里我们将三个算法扩展到他们对应的 kernel 版本。kernel K 由以下方法构建，首先，用文献 [57] 的方法构建相似度矩阵 A，接着，寻找一个离 A 最近的半正定矩阵 K 作为 kernel，这里近与远用矩阵的 Frobenius 范数来度量，即求解

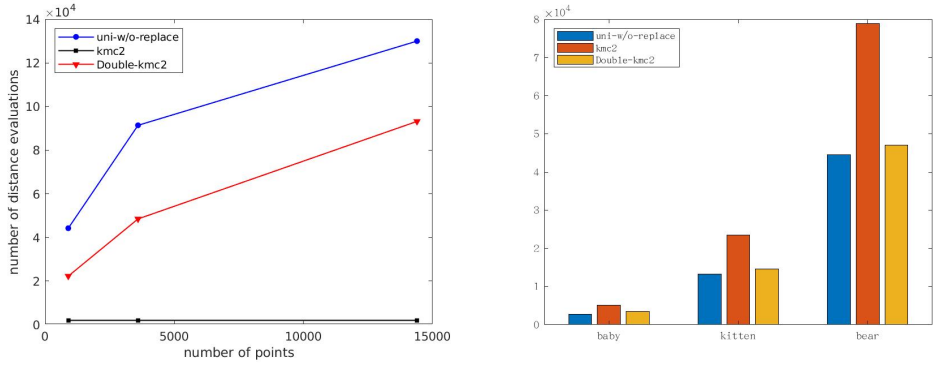
$$\begin{aligned} \min_{K} \quad & \|K-A\|_F \\ s.t. \quad & K \succeq 0 \end{aligned}$$

$K \succeq 0$ 表示 K 是半正定矩阵。由于三个算法只返回 k 个点，k 个划分是通过把所有点靠到离它最近的返回点上计算得到的。测试的图片是“kitten”、“bear”和“baby”[①]，这些图片由文献 [58] 提供。我们调整了这些图片的长和宽，使得聚类的点的数目从 900（30*30）到 14400（120*120）。这个任务中 kernel *k*-means 的目标函数值作为聚类质量的评价指标，效率的评价指标和传统聚类中的一样。

基于均匀不放回采样的 kernel 版、kernel K-MC^2 和 kernel Double-K-MC^2 的参数配置如下：均匀不放回的采样数 $s' = 0.4 * \log^4 n$，kernel *k*-means++ 是均匀不放回

① 下载地址 http://www.cs.utexas.edu/users/dml/Software/graclus.html

的 α 近似算法，kernel K-MC^2 和 kernel Double-K-MC^2 的游走次数都是 200，kernel Double-K-MC^2 的采样量是 $s = 0.25 * \log^2 n$，所有图片的 k 都设为 5，所有算法重复 30 次取平均值。



(a) (b)

图 2-2 图像分割任务上的结果。（a）图片数据集上的距离计算次数；（b）图片数据集上的 kernel k-means 目标函数值

图像分割实验结果见图2-2。实验结论和传统聚类的差不多，kernel 版的均匀不放回采样有最好的聚类质量，而时间增长的趋势并不是很快，kernel Double-K-MC^2 比均匀不放回在聚类质量上稍微差一点但是时间少了很多，kernel K-MC^2 有最快的速度但是聚类质量太差。因此，如果你想要一个折中的平衡算法，kernel Double-K-MC^2 是一个好的选择，如果你更关心效率，选择 kernel K-MC^2。图2-2中的数据列在表2-3中，目标函数值的量级是 10^4，时间开销的量级是 10^4。

表 2-3 图像分割任务上 kernel k-means 的目标函数值和距离计算次数（在括号中）

数据集	kernel K-MC^2	kernel Double-K-MC^2	均匀不放回的 kernel 版
baby(30*30)	0.508(0.200)	0.341(2.228)	0.280(4.424)
kitten(60*60)	2.345(0.200)	1.459(4.848)	1.323(9.140)
bear(120*120)	7.891(0.200)	4.703(9.319)	4.448(13.000)

2.8 本章小结

至此，本章梳理了在 k-means 问题上的加速算法，有理论保证的算法和两者都有的算法，这些算法的相关结论总结在表2-4中，其中 $O(\mathcal{A}_c)$ 是 α 近似算法的时间复杂度。根据表2-4，我们对这些算法分别从速度、理论保证和实用性这三个维度进行评价。这些算法按照时间复杂度从小到大排序如下：K-MC^2、均匀不放回采样 (2)、AFK-MC^2、Lightweight Coreset、k-means++、算法2-12(1)、ε-Coreset、

表 2-4 理论保证及加速相关算法结果汇总

算法	采样量	时间复杂度	聚类质量
k-means++	k	$O(nkd)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq 8(\ln k+2)$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})$
p-swap	k	$O(n^3)$	$\phi_C(\mathcal{X}) \leq 18\phi_{\mathrm{OPT}}(\mathcal{X})$
k-means++ &swap	k	$O(ndk^2 \log\log k)$	$\mathbb{E}[\phi_C(\mathcal{X})] \in O(\phi_{\mathrm{OPT}}(\mathcal{X}))$
k-means|| sampling(1)	$O(kt)$	$O(ndkt)$	$\mathbb{E}[\phi_S(\mathcal{X})] \leq (\frac{1+\alpha}{2})^t\xi$ $+\frac{16}{1-\alpha}\phi_{\mathrm{OPT}}(\mathcal{X})$
k-means|| sampling(2)	$O(lt)$	$O(ndlt)$	$\mathbb{E}[\phi_S(\mathcal{X})] \leq 2(\frac{k}{el})^t\mathrm{Var}(\mathcal{X})$ $+26\phi_{\mathrm{OPT}}(\mathcal{X})$
k-means|| seeding	$O(lt)$	$O(ndlt)+O(ndlt)$ $+O(\mathcal{A}_c)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq O(\alpha)\phi_{\mathrm{OPT}}(\mathcal{X})$
K-MC^2	k	$O(k^3 d\log^2 n\log k)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq O(\log k)$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})$
AFK-MC^2	k	$O(nd)+O(\frac{1}{\varepsilon}k^2 d\log\frac{k}{\varepsilon})$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq 8(\log k+2)$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})+\varepsilon\mathrm{Var}(\mathcal{X})$
ε-Coreset	$O(\frac{k\log k}{\varepsilon^2}(dk\log k+\log\frac{1}{\delta}))$	$O(ndk\log\frac{1}{\delta})$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq \alpha(1+O(\varepsilon))$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})$
Lightweight Coreset	$O(\frac{dk\log k+\log(1/\delta)}{\varepsilon^2})$	$O(nd)+O(\mathcal{A}_c)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq \alpha\phi_{\mathrm{OPT}}(\mathcal{X})$ $+\alpha O(\varepsilon)\mathrm{Var}(\mathcal{X})$
均匀不放回采样 (1)	$s=O(\ln(\frac{1}{\delta})\frac{\Delta^2\alpha^2}{2\beta^2m^2})$	$O(s)+O(\mathcal{A}_c)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq (\alpha+\beta)$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})$
均匀不放回采样 (2)	$s=O(\ln(\frac{1}{\delta})\frac{\alpha^2}{\beta^2}k^2\log^4 n)$	$O(k^2\log^4 n)+O(\mathcal{A}_c)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq (\alpha+\beta)$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})$
算法2-12(1)	$O(k)$	$O(nkd)+O(nkd)$ $+O(\mathcal{A}_c)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq O(\alpha)\phi_{\mathrm{OPT}}(\mathcal{X})$
算法2-12(2)	$s=\Theta\left(\frac{k\log n}{\delta^{d/2}\epsilon^d}\log^{d/2}\left(\frac{k\log n}{\delta^{d/2}\epsilon^d}\right)\right)$	$O(nsd)+O(nsd)$ $+O(\mathcal{A}_c)$	$\mathbb{E}[\phi_C(\mathcal{X})] \leq \alpha(1+O(\varepsilon))$ $\cdot\phi_{\mathrm{OPT}}(\mathcal{X})$

k-means++&swap、k-means|| seeding、算法2-12(2) 和 p-swap。在排序中，1、2 名是次线性时间（需要假设），3、4 名在 $O(nd)$ 量级，5、6、7 名在 $O(nkd)$ 量级，最后 4 名从 $O(ndk^2)$ 逐渐增加到 $O(n^3)$。这些算法中后 6 名有常数近似，k-means++ 是 $O(\log k)$，而前 4 名不是要假设就是要加方差项。最后一个，算法实用与否也非常重要，从实用角度讲，这些算法可以分为 3 类，第一类是 AFK-MC^2、Lightweight Coreset、ε-Coreset、k-means|| seeding，这些算法并行性较好，也无需假设，非常实用；第二类是 K-MC^2、均匀不放回采样 (2)、k-means++、算法2-12(1)、算法2-12(2)、k-means++&swap，这些算法不是要假设就是有强序列属性不利于并行，实用性一般；最后剩下的就是 p-swap，时间复杂度高，不实用。 总结来说，如果追求速度推荐使用 K-MC^2、均匀不放回采样 (2)、AFK-MC^2 和 Lightweight Coreset，如果追

求聚类质量推荐使用 ε-Coreset 和 k-means|| seeding。

梳理完算法后，我们从理论上证明了基于均匀采样的算法2-11能有更好的近似系数，具体来说从 $4(\alpha+\beta)$ 降到了 $\alpha+\beta$，并且在 K-MC^2 所依赖的两个假设，假设 1 和 2 成立的情况下，均匀采样的数目可以在 $O(\log^4 n)$ 数量级，从而在次线性时间内取得了常数近似的解。接着实验表明，通过多采样一些点，确实可以让均匀采样比 K-MC^2 有更好的聚类质量，在传统聚类任务和图像分割任务中这一点均有体现。最后我们指出一些可能的未来研究方向：

1. 在次线性时间内有理论保证且无需假设的算法，比如近似系数是 $O(\log n)$ 的等等
2. 提出新的次线性时间算法，依靠新的易于验证的假设
3. 探索基于距离的自适应采样的新性质，比如进一步缩小定理2.21的界，比如定义 $d'(x,C)=\|x-c_1\|+\|x-c_2\|$，$c_1$ 和 c_2 分别是 C 中离 x 最近的和第二近的点，根据 d' 来挑选类，增加跳转到不同最优类的概率，以改进理论界等等。
4. 估算权重以加速算法2-12
5. 考察 k-means|| sampling 是否可以形成 Coreset，从而改进 k-means|| seeding 的理论界。

随着新的分析技术的发展，k-means 问题历久弥新，新的理论结果依然在不断出现，实用的算法层出不穷，随着大数据时代的到来，该问题必将焕发出新的活力。

第三章 谱聚类

谱聚类是从图论中演化出来的算法，后来在聚类中得到了广泛的应用。它的主要思想是把所有的数据看做空间中的点，这些点之间可以用边连接起来。点和点之间的相似度以边的权重形式体现，相似度越高权重越高。通过对所有数据点组成的图进行切图，让切图后不同的子图间边权重和尽可能的低，而子图内的边权重和尽可能的高，从而达到聚类的目的。本节我们首先介绍该问题的相关背景然后在介绍相关加速算法。

3.1 谱聚类引入及背景

由于谱聚类是基于图论的，因此我们首先温习下图的概念。

3.1.1 图的概念

对于一个图 G，我们一般用点的集合 V 和边的集合 E 来描述。即为 $G(V,E)$。其中 V 即为我们数据集里面所有的点 $(v_1, v_2, ...v_n)$。对于 V 中的任意两个点 v_i, v_j，我们定义 w_{ij} 为它们之间的边的权重。谱聚类切割的图是无向图，所以 $w_{ij} = w_{ji}$。

任意 $w_{ij} \geq 0$，如果两个点不相似，$w_{ij} = 0$。对于图中的任意一个点 v_i，它的度 d_i 定义为和它相连的所有边的权重之和，即

$$d_i = \sum_{j=1}^{n} w_{ij}$$

利用每个点度的定义，我们可以得到一个 $n \times n$ 的度矩阵 D, 它是一个对角矩阵，只有主对角线有值，对应第 i 行的第 i 个点的度数，定义如下：

$$D = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix}$$

根据所有点间的权重值，我们可以得到图的邻接矩阵 W，因为 W 反映了两个点的相似度，我们也把 W 称为相似度矩阵。它也是一个 $n \times n$ 的矩阵，第 i 行的第 j 个值对应权重 w_{ij}。除此之外，对于点集 V 的的一个子集 $A \subset V$，我们定义：

$$|A| := \text{子集 A 中点的个数}$$
$$vol(A) := \sum_{i \in A} d_i$$

3.1.2 相似度矩阵的构建

在上一节我们讲到了邻接矩阵 W，它是由任意两点之间的权重值 w_{ij} 组成的矩阵。通常我们可以自己输入权重，但是在谱聚类中，我们只有数据点的定义，并没有直接给出这个邻接矩阵，那么怎么得到这个邻接矩阵呢?

基本思想是，距离较远的两个点之间的边权重值较低，而距离较近的两个点之间的边权重值较高，不过这仅仅是定性，我们需要定量的计算。一般来说，构建邻接矩阵 W 的方法有三类。ε-邻近法，k 邻近法和全连接法。

对于 ε-邻近法，它设置了一个距离阈值 ε，然后用欧式距离度量任意两点 x_i 和 x_j 的距离。即令 $s_{ij} = ||x_i - x_j||_2^2$, 然后根据 s_{ij} 和 ε 的大小关系，来定义邻接矩阵 W 如下：

$$w_{ij} = \begin{cases} 0 & s_{ij} > \varepsilon \\ \varepsilon & s_{ij} \leq \varepsilon \end{cases}$$

从上式可见，两点间的权重要不就是 ε, 要不就是 0，没有其他的信息了。距离远近度量很不精确，因此在实际应用中，我们很少使用 ε-邻近法。

第二种定义邻接矩阵 W 的方法是 k 邻近法，利用 KNN 算法遍历所有的样本点，取每个样本最近的 k 个点作为近邻，只有和样本距离最近的 k 个点之间的 $w_{ij} > 0$。但是这种方法会造成重构之后的邻接矩阵 W 非对称，我们后面的算法需要对称邻接矩阵。为了解决这种问题，一般采取下面两种方法之一，第一种 k 邻近法是只要一个点在另一个点的 k 近邻中，则计算 w_{ij}

$$w_{ij} = w_{ji} = \begin{cases} 0 & x_i \notin KNN(x_j) \ and \ x_j \notin KNN(x_i) \\ exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2}) & x_i \in KNN(x_j) \ or \ x_j \in KNN(x_i) \end{cases}$$

第二种 k 邻近法是必须两个点互为 k 近邻中，才计算 w_{ij}

$$w_{ij} = w_{ji} = \begin{cases} 0 & x_i \notin KNN(x_j) \ or \ x_j \notin KNN(x_i) \\ exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2}) & x_i \in KNN(x_j) \ and \ x_j \in KNN(x_i) \end{cases}$$

第三种定义邻接矩阵 W 的方法是全连接法，相比前两种方法，第三种方法所有的点之间的权重值都大于 0，因此称之为全连接法。可以选择不同的核函数来定义边权重，常用的有多项式核函数，高斯核函数和 Sigmoid 核函数。最常用的是高

斯核函数 RBF：

$$w_{ij} = exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2})$$

在实际的应用中，使用第三种全连接法来建立邻接矩阵是最普遍的。

3.1.3 拉普拉斯矩阵

单独把拉普拉斯矩阵 (Graph Laplacians) 拿出来介绍是因为后面的算法和这个矩阵的性质息息相关。它的定义很简单，拉普拉斯矩阵 $L = D - W$。D 即为我们第二节讲的度矩阵，它是一个对角矩阵。而 W 即为我们第二节讲的邻接矩阵，它可以由我们第三节的方法构建出。

拉普拉斯矩阵有一些很好的性质如下：

1. 拉普拉斯矩阵是对称矩阵，这可以由 D 和 W 都是对称矩阵而得。
2. 由于拉普拉斯矩阵是对称矩阵，则它的所有的特征值都是实数。
3. 对于任意的向量 f, 我们有

$$f^T Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

证明如下

$$f^T Lf = f^T Df - f^T Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j$$

$$= \frac{1}{2}(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{j=1}^{n} d_j f_j^2) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

4. 拉普拉斯矩阵是半正定的，且对应的 n 个实数特征值都大于等于 0，即 $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$，且最小的特征值为 0，这个由性质 3 很容易得出。

3.1.4 图切割

对于无向图 G 的切图，我们的目标是将图 $G(V, E)$ 切成相互没有连接的 k 个子图，每个子图点的集合为：$A_1, A_2, ..A_k$，它们满足 $A_i \cap A_j = \emptyset$, 且 $A_1 \cup A_2 \cup ... \cup A_k = V$.

对于任意两个子图点的集合 $A, B \subset V, A \cap B = \emptyset$, 我们定义 A 和 B 之间的切图权重为：

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

那么对于我们 k 个子图点的集合：$A_1, A_2, ..A_k$，我们定义切图 cut 为：

$$cut(A_1, A_2, ...A_k) = \frac{1}{2}\sum_{i=1}^{k} W(A_i, \overline{A}_i)$$

其中 $\bar{A}_i$ 为 A_i 的补集，意为 $V - A_i$。

那么如何切图可以让子图内的点权重和高，子图间的点权重和低呢？一个自然的想法就是最小化 $cut(A_1, A_2, ...A_k)$, 但是可以发现，这种极小化的切图存在问题，如图 1 所示， 我们选择一个权重最小的边缘的点，比如 C 和 H 之间进行 cut，这

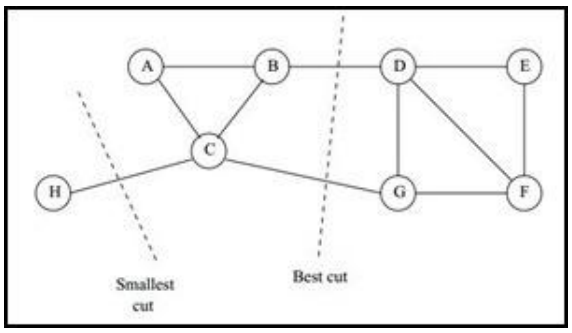


图 3-1 可能出现的不好的切割

样可以最小化 $cut(A_1, A_2, ...A_k)$, 但是却不是最优的切图，如何避免这种切图，并且找到类似图中”Best Cut” 这样的最优切图呢？我们下一节就来看看谱聚类中常用的切图方法。

3.1.5 Normalized Cut

为了避免最小切图导致的切图效果不佳，我们需要对每个子图的规模做出限定，一般来说，有两种切图方式，第一种是 RatioCut，第二种是 Ncut。一般我们使用 Ncut，故在此只介绍 Ncut。

Normalized cut 切图为了避免上一节中的不好的切图，对每个切图，不光考虑最小化 $cut(A1, A2, ...Ak)$，它还同时考虑每个类中点的数目尽可能平衡，即

$$NCut(A_1, A_2, ...A_k) = \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \overline{A}_i)}{vol(A_i)}$$

为了优化该目标函数，我们定义如下指示向量。

$$h_{ij} = \begin{cases} 0 & v_i \notin A_j \\ \frac{1}{\sqrt{vol(A_j)}} & v_i \in A_j \end{cases}$$

则对于 $h_i^T L h_i$, 有：

$$h_i^T L h_i = \frac{1}{2}\sum_{m=1}\sum_{n=1} w_{mn}(h_{im}-h_{in})^2 \tag{3-1}$$

$$= \frac{1}{2}\Big(\sum_{m\in A_i, n\notin A_i} w_{mn}\Big(\frac{1}{\sqrt{vol(A_i)}}-0\Big)^2 + \sum_{m\notin A_i, n\in A_i} w_{mn}\Big(0-\frac{1}{\sqrt{vol(A_i)}}\Big)^2 \tag{3-2}$$

$$= \frac{1}{2}\Big(\sum_{m\in A_i, n\notin A_i} w_{mn}\frac{1}{vol(A_i)} + \sum_{m\notin A_i, n\in A_i} w_{mn}\frac{1}{vol(A_i)} \tag{3-3}$$

$$= \frac{1}{2}\Big(cut(A_i,\overline{A}_i)\frac{1}{vol(A_i)} + cut(\overline{A}_i,A_i)\frac{1}{vol(A_i)}\Big) \tag{3-4}$$

$$= \frac{cut(A_i,\overline{A}_i)}{vol(A_i)} \tag{3-5}$$

也就是说

$$NCut(A_1,A_2,...A_k) = \sum_{i=1}^{k} h_i^T L h_i = \sum_{i=1}^{k}(H^T LH)_{ii} = tr(H^T LH)$$

我们将 Ncut 的优化问题转化为了一个二次型优化问题，并注意到该问题有如下限制

$$h_i^T D h_i = \sum_{j=1}^{n} h_{ij}^2 d_j = \frac{1}{vol(A_i)}\sum_{j\in A_i} d_j = \frac{1}{vol(A_i)} vol(A_i) = 1$$

此时，我们得到如下优化问题

$$\underset{H}{\text{argmin}}\ \ tr(H^T LH)\ \ s.t.\ H^T DH = I$$

注意到 H 是离散的，此时该问题是 NP 难的，为了求解我们允许 H 可以取连续值，该步骤称为 relaxation，使用拉格朗日乘子法可以求解，解为 $D^{-1/2}LD^{-1/2}$ 的最小的前 k 个特征值，对应的 H 为 $D^{-1/2}LD^{-1/2}$ 的前 k 个特征向量，对 H 的每一行再做聚类，比如 k-means 即可得到最后的聚类结果。

3.2 加速谱聚类

3.2.1 Nyström 方法

Nyström 方法是一种半正定矩阵近似方法，可以有效近似半正定矩阵，该方法由文献 [59] 引入到机器学习领域。其核心思想是：通过在原矩阵中采样部分列或者行，用这部分信息对原矩阵进行低秩近似。它也可以近似特征值和特征向量，通过分解大矩阵中小的子矩阵来近似大矩阵的特征值和特征向量，方法详解如下。

假设我们有一个半正定矩阵 $K \in \mathbb{R}^{n\times n}$，因为它是半正定，所以有

$$K = X^T X$$

可将 X 分块，$X = [R \quad S]$，$X \in \mathbb{R}^{d\times n}$，其中 $R \in \mathbb{R}^{d\times m}$，所以

$$K = \begin{bmatrix} R^T R & R^T S \\ S^T R & S^T S \end{bmatrix}$$

另一方面，我们也可以通过某种采样方法（比如均匀采样）选择 K 的 m 行（或者列），一般来说 K 表示的是数据点之间的关系，所以可以调整选中行或者列在 K 中的顺序而不会影响原始信息，不妨将选中的行调整到 K 的前排，没被选中的都放到后排，从而将 K 分块表示为如下形式

$$K = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

其中 $[A \quad B]$ 是我们采样的行，此时，我们令 $A = R^T R$，$B = R^T S$（通常情况，等式不会自然成立，这里我们令他们成立，目的是用 A, B 来表示 C）。首先，我们对 A 做谱分解求得 R，

$$A = R^T R = U\Sigma U^T$$

即，$R = \Sigma^{\frac{1}{2}} U^T$，又因为

$$B = R^T S = U\Sigma^{\frac{1}{2}} S$$

所以，$S = \Sigma^{-\frac{1}{2}} U^T B$，此时 $C \approx B^T A^{-1} B$，记用 A, B 近似的 K 为 $\hat{K}$，容易知道它是 K 的一个低秩近似，秩最多是 m，将 $B^T A^{-1} B$ 带入 $\hat{K}$ 有

$$\hat{K} = \begin{bmatrix} A & B \\ B^T & B^T A^{-1} B \end{bmatrix} = \begin{bmatrix} U\Sigma U^T & B \\ B^T & B^T U\Sigma^{-1} U^T B \end{bmatrix} = \begin{bmatrix} U \\ B^T U\Sigma^{-1} \end{bmatrix} \Sigma \begin{bmatrix} U^T & \Sigma^{-1} U^T B \end{bmatrix} = K_{:1} A^{-1} K_{:1}^T$$

也就是说，我们可以用 $\begin{bmatrix} U \\ B^T U\Sigma^{-1} \end{bmatrix} = K_{:1} U\Sigma^{-1}$ 来近似 K 的特征向量，用 A 的特征值 Σ 来近似 K 的特征值，这里 $K_{:1} = [A \quad B]^T$ 是我们采样的行。由于只需要对 A 这个 $\mathbb{R}^{m\times m}$ 的矩阵做分解就可以近似大矩阵的特征值和特征向量，故我们加速了大矩阵的分解。完整 Nyström 算法见3-1，该算法时间复杂度是 $O(nm^2 + m^3)$。对于

算法 3-1 Nyström 方法

输入: 半正定矩阵 K，采样数目 m
输出: 低秩矩阵 $\hat{K}$

1 $S \leftarrow$ 根据某些采样算法得到的列（或者行）的下标
2 $A \leftarrow K(S,S)$
3 $K_{:1} \leftarrow K(:,S)$
4 令 $A = U\Sigma U^T$ 是 A 的谱分解，U,Σ 分别是 A 的特征向量和特征值。
5 **返回**低秩矩阵 $\hat{K} = K_{:1}A^{-1}K_{:1}^T$ 或者近似的特征向量 $\hat{U}_K = K_{:1}U\Sigma^{-1}$

Nyström 方法，我们应该注意什么呢？第一，矩阵 K 应该本身就是低秩的，否则使用 Nyström 方法将没有意义。第二，行的选择很关键，如果我们选到了大量的线性相关的行，则估计误差就会很大。将 A 改成 K

定理 3.1 令 $\varepsilon \in (0,1)$，$\delta \in (0,1)$，考虑用均匀采样 m 列（可以放回也可以不放回），则当 m 满足 $m \geq 2\varepsilon^{-2}\mu k \log(k/\delta)$ 时

$$\|A - \tilde{A}\|_2 \leq \left(1 + \frac{n}{(1-\epsilon)m}\right)\|A - A_k\|_2$$

以至少 $1-3\delta$ 的概率成立，$\|.\|_2$ 是矩阵的谱范数 (spectral norm)，其中

$$\mu = \frac{n}{k}\max_{i=1,\ldots,n}\|V_k(i,:)\|_2^2$$

记 V 是将 A 的特征值从大到小排序后对应的特征向量，V_k 是 V 的前 k 列，A_k 是 A 的最优的秩为 k 的近似矩阵。

常用的的采样（列选择）是均匀采样，基于文献 [60] 的结论，定理3.1给出了均匀采样的话 Nyström 方法的误差，这里 μ 被称为 V_k 的 coherence。可以看到如果 coherence 比较大的话，均匀采样效果不太好。因此，一些更加高级的采样方法被提了出来，比如以概率 $\|V_k(i,:)\|_2^2/k$ 采样第 i 列，有放回的重复 m 次的采样方式，这个概率也被称为 leverage score。

定理 3.2 令 $\varepsilon \in (0,1)$，$\delta \in (0,1)$，考虑按照上面描述的高级采样方式采样 m 列，则当 $m \geq O\left(\varepsilon^{-2}k\log(k/\delta)\right)$ 时有

$$\|A - \tilde{A}\|_2 \leq \|A - A_k\|_2 + \varepsilon^2\|A - A_k\|_*$$

以至少 $0.8-2\delta$ 的概率成立，其中 $\|.\|_*$ 是矩阵的迹范数 (trace norm)

当然，精确计算 leverage score 需要对 A 做 SVD，而这正是我们想避免的。尽管如此，可以在差不多 $O(ek\log e)$ 的时间复杂度下对 leverage score 做近似，如果 A 有 $O(e)$ 个非零元素的话，这里近似的误差是一个乘性误差。到目前为止最快的

利用 leverage score 的 Nyström 方法的时间复杂度接近 n 的线性函数，该算法用了一种复杂的递归的采样方式[61]。

那么怎么将 Nyström 方法用到谱聚类上呢？令 $A \in \mathbb{R}^{n\times n}$ 是谱聚类的相似度矩阵，它对应的归一化的拉普拉斯矩阵是 $L_n = I - D^{-1/2}AD^{-1/2}$，回忆求解谱聚类需要找到 L_n 的最小的 k 个特征值对应的特征向量，这等价于找到 $M = D^{-1/2}AD^{-1/2}$ 最大的 k 个特征值对应的特征向量，所以我们的目的就是要近似这些特征向量，通过前面的分析，我们可以通过对 A 分块拿到 A 的近似特征向量，这里的基本思想是用 A 的分块来表示 M 的分块，然后用 M 的分块利用 Nyström 方法近似 M 的特征向量，该方法由文献 [62] 提出，详解如下。我们将 A 做如下分块

$$A = \left[\begin{array}{c|c} A_{11} & A_{21}^T \\ \hline A_{21} & A_{22} \end{array}\right]$$

我们知道的是 A_{11} 和 A_{21}，根据 Nyström 方法我们可以得到一个低秩近似 $\hat{A}$，根据 $\hat{A}$ 可以得到一个近似的度矩阵 $\hat{D} = \mathrm{diag}(\hat{A}1)$，$\mathrm{diag}(v)$ 是一个以向量 v 为对角元的对角矩阵，1 是一个元素全为 1 的向量，类比于 A 的分块，将 $\hat{D}$ 分块。

$$\hat{A} = \left[\begin{array}{c|c} A_{11} & A_{21}^T \\ \hline A_{21} & A_{21}A_{11}^{-1}A_{21}^T \end{array}\right], \hat{D} = \left[\begin{array}{c|c} D_{11} & \\ \hline & \hat{D}_{22} \end{array}\right]$$

有了 $\hat{A}$ 和 $\hat{D}$ 可以近似 M，令近似矩阵是 $\hat{M}$，则有

$$\hat{M} = \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2} = \left[\begin{array}{c|c} M_{11} & \hat{M}_{21}^T \\ \hline \hat{M}_{21} & \hat{M}_{22} \end{array}\right]$$

其中 $M_{11} = D_{11}^{-1/2}A_{11}D_{11}^{-1/2}$，$\hat{M}_{21} = \hat{D}_{22}A_{21}D_{11}^{-1/2}$，记 $\hat{M}_{:1} = \begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix}$，此时即可用算法3-1来近似 M 的特征向量，只不过此时算法3-1中的 A 是 M_{11}，$K_{:1}$ 是 $\hat{M}_{:1}$。

值得注意的是，这样求出来的的近似特征向量不可直接基于它用 k-means 聚类，原因是这个近似的特征向量列不正交，由于不正交，基于这样的特征向量可能得出的谱聚类的解质量很差，所以我们需要对现有近似特征向量做后处理，使其正交。在文献 [62] 中，作者根据分块矩阵 M_{11} 是否是半正定的提出了两种方法。

（一）M_{11} 是半正定

首先，考虑将 $\hat{M}$ 写成谱分解的样子

$$\begin{aligned}\hat{M} &= \begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} M_{11}^{-1} \begin{bmatrix} M_{11} & \hat{M}_{21}^T \end{bmatrix} \\ &= \left(\begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} M_{11}^{-1/2} U \Sigma^{-1/2} \right) \Sigma \left(\Sigma^{-1/2} U^T M_{11}^{-1/2} \begin{bmatrix} M_{11} & \hat{M}_{21}^T \end{bmatrix} \right) \\ &= V \Sigma V^T \end{aligned}$$

上式中 Σ 是任意对角矩阵，U 是任意正交矩阵，为了求出 U 和 Σ，我们要求 V 列正交，有

$$\begin{aligned} I &= V^T V \\ &= \left(\Sigma^{-1/2} U^T M_{11}^{-1/2} \begin{bmatrix} M_{11} & \hat{M}_{21}^T \end{bmatrix} \right) \left(\begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} M_{11}^{-1/2} U \Sigma^{-1/2} \right) \end{aligned}$$

对上面等式左乘 $U\Sigma^{1/2}$，右乘 $\Sigma^{1/2}U^T$，有

$$\begin{aligned} U \Sigma U^T &= \left(M_{11}^{-1/2} \begin{bmatrix} M_{11} & \hat{M}_{21}^T \end{bmatrix} \right) \left(\begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} M_{11}^{-1/2} \right) \\ &= M_{11} + M_{11}^{-1/2} \hat{M}_{21}^T \hat{M}_{21} M_{11}^{-1/2} \end{aligned}$$

所以说，我们只要对 $S = M_{11} + M_{11}^{-1/2} \hat{M}_{21}^T \hat{M}_{21} M_{11}^{-1/2}$ 做谱分解，得到它的特征向量 U_S 和特征值 Σ_S 就可以得到 M 的近似特征向量 $V = \begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} M_{11}^{-1/2} U_S \Sigma_S^{-1/2}$

（二）M_{11} 不是半正定

如果 M_{11} 不是半正定，$M_{11}^{-1/2}$ 就没有定义，前面方法就不可以正交化，文献 [62] 中将情况 1 的方式称为 one-shot，因为不用求出不正交的特征向量，而这里需要按照以下两步进行。首先，根据前面的方法得出近似的但是不正交的特征向量 $\bar{V} = \hat{M}_{:1} U_{M_{11}} \Sigma_{M_{11}}^{-1}$，其中 $U_{M_{11}}$ 和 $\Sigma_{M_{11}}^{-1}$ 分别是 M_{11} 的特征向量和特征值。接着，取 $Z = \bar{V} \Sigma_{M_{11}}^{1/2}$，对 Z 做 SVD，U_Z 即为所求的正交化处理后的特征向量

$$Z = U_Z \Sigma_Z V_Z^T$$

这个基于 Nyström 方法加速谱聚类的方法总结见算法3-2，这里 $\hat{U}_{:k}$ 是 $\hat{U}$ 最大的 k 个特征值对应的特征向量。

算法 3-2 Nyström 方法加速谱聚类

输入: 近似的相似度矩阵 $\hat{A}$
输出: 归一化拉普拉斯矩阵 L_n 最小的 k 个特征值对应的特征向量

$\hat{D} \leftarrow \mathrm{diag}(\hat{A}1)$
$M_{11} \leftarrow D_{11}^{-1/2} A_{11} D_{11}^{-1/2}$
$\hat{M}_{21} \leftarrow \hat{D}_{22} A_{21} D_{11}^{-1/2}$
if M_{11} 是半正定矩阵 **then**
    $S = M_{11} + M_{11}^{-1/2} \hat{M}_{21}^T \hat{M}_{21} M_{11}^{-1/2}$
    $U_S, \Sigma_S \leftarrow$ 对 S 做谱分解得到它的特征向量和特征值
    $\hat{U} = \begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} M_{11}^{-1/2} U_S \Sigma_S^{-1/2}$
else
    $U_{M_{11}}, \Sigma_{M_{11}} \leftarrow$ 对 M_{11} 做谱分解得到它的特征向量和特征值
    $Z \leftarrow \begin{bmatrix} M_{11} \\ \hat{M}_{21} \end{bmatrix} U_{M_{11}} \Sigma_{M_{11}}^{-1/2}$
    $\hat{U} \leftarrow$ 对 Z 做奇异值分解得到它的左奇异矩阵
end
返回 $\hat{U}_{:k}$

3.3 谱聚类的理论保证

3.4 加速谱聚类且有理论保证

3.5 我的贡献

3.6 实验

3.7 本章小结

第四章 全文总结与展望

4.1 全文总结

本文以时域积分方程方法为研究背景，主要对求解时域积分方程的时间步进算法以及两层平面波快速算法进行了研究。

4.2 后续工作展望

时域积分方程方法的研究近几年发展迅速，在本文研究工作的基础上，仍有以下方向值得进一步研究：

致 谢

在攻读博士学位期间，首先衷心感谢我的导师 XXX 教授

参考文献

[1] K. Akiyama, A. Alberdi, W. Alef, et al. First m87 event horizon telescope results. iv. imaging the central supermassive black hole[J]. The Astrophysical Journal Letters, 2019, 875(1): L4

[2] 宁康, 陈挺. 生物医学大数据的现状与展望 [J]. 科学通报 (中文版), 2015, 60(5/6): 534-546

[3] M. A. Beyer, D. Laney. The importance of ‘big data’: a definition[J]. Stamford, CT: Gartner, 2012, : 2014-2018

[4] J. MacQueen, et al. Some methods for classification and analysis of multivariate observations[C]. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967, 281-297

[5] U. Von Luxburg. A tutorial on spectral clustering[J]. Statistics and computing, 2007, 17(4): 395-416

[6] K. Pearson. Liii. on lines and planes of closest fit to systems of points in space[J]. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1901, 2(11): 559-572

[7] J. B. Tenenbaum, V. De Silva, J. C. Langford. A global geometric framework for nonlinear dimensionality reduction[J]. science, 2000, 290(5500): 2319-2323

[8] S. T. Roweis, L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding[J]. science, 2000, 290(5500): 2323-2326

[9] 贺玲, 吴玲达, 蔡益朝. 数据挖掘中的聚类算法综述 [J]. 计算机应用研究, 2007, 24(1): 10-13

[10] 孙吉贵, 刘杰, 赵连宇. 聚类算法研究 [J]. 软件学报, 2008, 19(1): 48-61

[11] 周涛, 陆惠玲. 数据挖掘中聚类算法研究进展 [J]. 计算机工程与应用, 2012, 48(12): 100-111

[12] A. K. Jain, R. C. Dubes. Algorithms for clustering data[M]. Prentice-Hall, Inc., 1988

[13] S. Sambasivam, N. Theodosopoulos. Advanced data clustering methods of mining web documents.[J]. Issues in Informing Science & Information Technology, 2006, 3:

[14] M. S. G. Karypis, V. Kumar, M. Steinbach. A comparison of document clustering techniques[C]. TextMining Workshop at KDD2000 (May 2000), 2000,

[15] M. Garey, D. Johnson, H. Witsenhausen. The complexity of the generalized lloyd-max problem (corresp.)[J]. IEEE Transactions on Information Theory, 1982, 28(2): 255-256

[16] J. Kleinberg, C. Papadimitriou, P. Raghavan. A microeconomic view of data mining[J]. Data mining and knowledge discovery, 1998, 2(4): 311-324

[17] M. Mahajan, P. Nimbhorkar, K. Varadarajan. The planar k-means problem is np-hard[C]. International Workshop on Algorithms and Computation, 2009, 274-285

[18] F. Cao, J. Liang, G. Jiang. An initialization method for the k-means algorithm using neighborhood model[J]. Computers & Mathematics with Applications, 2009, 58(3): 474-483

[19] D. Arthur, S. Vassilvitskii. k-means++: The advantages of careful seeding[C]. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, 1027-1035

[20] V. Arya, N. Garg, R. Khandekar, et al. Local search heuristics for k-median and facility location problems[J]. SIAM Journal on computing, 2004, 33(3): 544-562

[21] M. Charikar, S. Guha. Improved combinatorial algorithms for the facility location and k-median problems[C]. 40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039), 1999, 378-388

[22] T. Kanungo, D. M. Mount, N. S. Netanyahu, et al. A local search approximation algorithm for k-means clustering[J]. Computational Geometry, 2004, 28(2-3): 89-112

[23] S. Lattanzi, C. Sohler. A better k-means++ algorithm via local search[C]. Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, USA, 2019, 3662-3671

[24] O. Bachem, M. Lucic, A. Krause. Distributed and provably good seedings for k-means in constant rounds[C]. International Conference on Machine Learning, 2017, 292-300

[25] M. Zaharia, M. Chowdhury, M. J. Franklin, et al. Spark: Cluster computing with working sets.[J]. HotCloud, 2010, 10(10-10): 95

[26] M. Zaharia, R. S. Xin, P. Wendell, et al. Apache spark: a unified engine for big data processing[J]. Communications of the ACM, 2016, 59(11): 56-65

[27] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications[J]. , 1970,

[28] O. Bachem, M. Lucic, S. H. Hassani, et al. Approximate k-means++ in sublinear time.[C]. , 2016,

[29] O. Bachem, M. Lucic, H. Hassani, et al. Fast and provably good seedings for k-means[C]. Advances in Neural Information Processing Systems, 2016, 55-63

[30] S. Ben-David. A framework for statistical clustering with a constant time approximation algorithms for k-median clustering[C]. International Conference on Computational Learning Theory, 2004, 415-426

[31] P. Awasthi, M. Charikar, R. Krishnaswamy, et al. The hardness of approximation of euclidean k-means[J]. arXiv preprint arXiv:1502.03316, 2015,

[32] E. Lee, M. Schmidt, J. Wright. Improved and simplified inapproximability for k-means[J]. Information Processing Letters, 2017, 120: 40-43

[33] S. Ahmadian, A. Norouzi-Fard, O. Svensson, et al. Better guarantees for k-means and euclidean k-median by primal-dual algorithms[C]. 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), 2017, 61-72

[34] D. Feldman, M. Monemizadeh, C. Sohler. A ptas for k-means clustering based on weak coresets[C]. Proceedings of the twenty-third annual symposium on Computational geometry, 2007, 11-18

[35] R. Jaiswal, A. Kumar, S. Sen. A simple d 2-sampling based ptas for k-means and other clustering problems[J]. Algorithmica, 2014, 70(1): 22-46

[36] Z. Friggstad, M. Rezapour, M. R. Salavatipour. Local search yields a ptas for k-means in doubling metrics[J]. SIAM Journal on Computing, 2019, 48(2): 452-480

[37] O. F. Bachem. Sampling for large-scale clustering[D]. , 2018,

[38] O. Bachem, M. Lucic, A. Krause. Scalable and distributed clustering via lightweight coresets[J]. arXiv preprint arXiv:1702.08248, 2017,

[39] M. R. Ackermann, M. Märtens, C. Raupach, et al. Streamkm++: A clustering algorithm for data streams[J]. Journal of Experimental Algorithmics (JEA), 2012, 17: 2-4

[40] O. Bachem, M. Lucic, A. Krause. Practical coreset constructions for machine learning[J]. arXiv preprint arXiv:1703.06476, 2017,

[41] D. Feldman, M. Schmidt, C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering[C]. Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms, 2013, 1434-1453

[42] Y. Ding, Y. Zhao, X. Shen, et al. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup[C]. International Conference on Machine Learning, 2015, 579-587

[43] C. Elkan. Using the triangle inequality to accelerate k-means[C]. Proceedings of the 20th international conference on machine learning (ICML-03), 2003, 147-153

[44] J. Drake, G. Hamerly. Accelerated k-means with adaptive distance bounds[C]. 5th NIPS workshop on optimization for machine learning, 2012,

[45] J. P. Newling. Novel algorithms for clustering[R]. EPFL,

[46] M. Mohan, C. Monteleoni. Beyond the nystrom approximation: Speeding up spectral clustering using uniform sampling and weighted kernel k-means[C]. Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, 2494-2500

[47] A. Czumaj, C. Sohler. Sublinear-time approximation for clustering via random sampling[C]. International Colloquium on Automata, Languages, and Programming, 2004, 396-407

[48] A. Aggarwal, A. Deshpande, R. Kannan. Adaptive sampling for k-means clustering[M]. Springer, 2009, 15-28

[49] D. Wei. A constant-factor bi-criteria approximation guarantee for k-means++[C]. Advances in Neural Information Processing Systems, 2016, 604-612

[50] R. R. Mettu, C. G. Plaxton. Optimal time bounds for approximate clustering[J]. Machine Learning, 2004, 56(1-3): 35-60

[51] S. Dasgupta. Lecture 3 - algorithms for k-means clustering[R]. UCSD, 2013

[52] I. Kärkkäinen, P. Fränti. Dynamic local search algorithm for the clustering problem[R]. Joensuu, Finland: Department of Computer Science, University of Joensuu,

[53] T. Zhang, R. Ramakrishnan, M. Livny. Birch: A new data clustering algorithm and its applications[J]. Data Mining and Knowledge Discovery, 1997, 1(2): 141-182

[54] C. Foussette, D. Hakenjos, M. Scholz. Kdd-cup 2004: protein homology task[J]. ACM SIGKDD Explorations Newsletter, 2004, 6(2): 128-131

[55] A. V. Uzilov, J. M. Keegan, D. H. Mathews. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change[J]. BMC bioinformatics, 2006, 7(1): 173

[56] R. Cattral, F. Oppacher, D. Deugo. Evolutionary data mining with automatic rule generalization[J]. Recent Advances in Computers, Computing and Communications, 2002, 1(1): 296-300

[57] X. Y. Stella, J. Shi. Multiclass spectral clustering[C]. null, 2003, 313

[58] I. S. Dhillon, Y. Guan, B. Kulis. Kernel k-means: spectral clustering and normalized cuts[C]. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, 551-556

[59] C. K. Williams, M. Seeger. Using the nyström method to speed up kernel machines[C]. Advances in neural information processing systems, 2001, 682-688

[60] A. Gittens, M. W. Mahoney. Revisiting the nyström method for improved large-scale machine learning[J]. The Journal of Machine Learning Research, 2016, 17(1): 3977-4041

[61] C. Musco, C. Musco. Recursive sampling for the nystrom method[C]. Advances in Neural Information Processing Systems, 2017, 3833-3845

[62] C. Fowlkes, S. Belongie, F. Chung, et al. Spectral grouping using the nystrom method[J]. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(2): 214-225

附录 A 常见的 concentration bound

攻读硕士学位期间取得的成果

[1] J. Y. Li, Y. W. Zhao, Z. P. Nie. New memory method of impedance elements for marching-on-in-time solution of time-domain integral equation[J]. Electromagnetics, 2010, 30(5): 448-462

[2] 张三, 李四. 时间步进算法中阻抗矩阵的高效存储新方法 [J]. 电波科学学报, 2010, 25(4): 624-631

[3] 张三, 李四. 时域磁场积分方程时间步进算法稳定性研究 [J]. 物理学报, 2013, 62(9): 090206-1-090206-6

[4] 张三，李四. 时域磁场积分方程时间步进算法后时稳定性研究. 电子科技大学学报 [J]（已录用，待刊）

[5] S. Zhang. Parameters discussion in two-level plane wave time-domain algorithm[C]. 2012 IEEE International Workshop on Electromagnetics, Chengdu, 2012, 38-39

[6] 张三, 李四. 时域积分方程时间步进算法研究 [C]. 电子科技大学电子科学技术研究院第四届学术交流会, 成都, 2008, 164-168

[7] 张三（4）. 人工介质雷达罩技术研究. 国防科技进步二等奖，2008 年

[8] XXX，XXX，XXX，XXX，王升．XXX 的陶瓷研究．四川省科技进步三等奖，2003 年 12 月