文章编号:1671-4598(2021)02-0176-04 DOI:10.16526/j.cnki.11—4762/tp.2021.02.035 中图分类号:TP301.6 文献标识码:A

基于改进 Canopy—K-means 算法的并行化研究

王 林,贾钧琛
(西安理工大学 自动化与信息工程学院,西安 710048)

摘要:随着互联网数据的快速增长,原始的 K-means 算法已经不足以应对大规模数据的聚类需求;为此,提出一种改进的 Canopy—K-means 聚类算法;首先面对 Canopy 算法中心点随机选取的不足,引入“最大最小原则”优化 Canopy 中心点的选取;接着借助三角不等式定理对 K-means 算法进行优化,减少冗余的距离计算,加快算法的收敛速度;最后结合 MapReduce 框架并行化实现改进的 Canopy—K-means 算法;基于构建的微博数据集,对优化后的 Canopy—K-means 算法进行测试;试验结果表明:对不同数据规模的微博数据集,优化后算法的准确率较 K-means 算法提高了约 15%,较原始的 Canopy—K-means 算法提高了约 7%,算法的执行效率和扩展性也有较大提升。

关键词:Canopy—K-means 算法;文本聚类;最大最小原则;三角不等式;MapReduce

Research on Parallelization Based on Improved Canopy—K-means Algorithm

Wang Lin, Jia Junchen
(School of Automation and Information Engineering, Xi'an University of Technology, Xi'an 710048, China)

Abstract: With the rapid growth of Internet data, the original K-means algorithm is no longer sufficient to meet the clustering needs of large—scale data. To this end, an improved Canopy—K-means clustering algorithm is proposed. Faced with the shortcomings of the random selection of the center point of the Canopy algorithm, the "maximum and minimum principle" was introduced to optimize the selection of the Canopy center point; then the K-means algorithm was optimized with the help of the triangle inequality theorem to reduce redundant distance calculations and accelerate the convergence rate of the algorithm; finally combined with MapReduce framework parallelization to achieve improved Canopy—K-means algorithm. Based on the constructed Weibo dataset, the optimized Canopy—K-means algorithm is tested. The test results show that the accuracy of the optimized algorithm is about 15% higher than that of the K-means algorithm and about 7% higher than that of the original Canopy—K-means algorithm. The execution efficiency and scalability of the algorithm are also improved. Greatly improved.

Keywords: Canopy—K-means algorithm; text clustering; maximum and minimum principle; triangle inequality; MapReduce

0 引言

随着互联网普及率的不断提高,网络数据呈几何级增长,面对海量以及快速增长的网络数据,通过聚类分析可以快速准确地从中挖掘出价值信息。但是,传统的聚类算法无论是在聚类精度,还是在执行时间上都已经不能很好地满足当前需求,利用分布式计算框架对其进行并行化改进,不仅可以缩短聚类时间,还可以增强算法的扩展性,更好地满足当下数据挖掘的需要。

K-means 算法作为一种具有代表性的聚类算法,具备较快的收敛速度、可靠的理论以及容易实现等诸多优势,因而被人们广泛应用于各行各业,但是算法也存在聚类中心点的选取具有随机性,需要提前确定聚类个数等不足[1]。对此,许多学者对 K-means 算法进行了改进并取得了一定的成果。

邓海等人[2]结合密度法和“最大最小原则”优化 K-means 初始聚类中心点的选择,算法准确率得到提高,但是改进后算法的时间复杂度较高,运行时间较长。赵庆等人[3]通过 Canopy 算法对数据集进行“粗”聚类,避免了传统 K-means 中心点选取存在的盲目性,极大提升了其准确性,然而在采用 Canopy 算法初始阈值需要人为指定,所以聚类结果不稳定。刘纪伟等人[4]结合密度思想优化了 K-means 初始中心点的选取,同时引入聚类有效性判别函数确定值,提高了算法的准确度,但是也增加了算法的运行时间,执行效率较低。李晓瑜等人[5]结合 MapReduce 分布式框架并行化实现改进的 Canopy—K-means 算法,并行化实现的算法具有良好的准确率和扩展性,但是 Canopy 算法初始阈值人为指定的问题仍然存在。

上述工作均是针对 K-means 算法初始中心点随机选取的不足进行改进,一定程度上提高了算法的聚类准确度,

收稿日期:2020-06-22; **修回日期**:2020-07-07。
基金项目:陕西省科技计划重点项目(2017ZDCXL—GY—05—03)。
作者简介:王 林(1963-),男,江苏东台人,博士,教授,主要从事大数据、数据挖掘、计算机视觉方向的研究;
引用格式:王 林,贾钧琛.基于改进 Canopy—K-means 算法的并行化研究[J].计算机测量与控制,2021,29(2):176-179,186.

然而仍旧存在不足。本文首先针对 Canopy－K－means 算法中 Canopy 中心点随机选取的不足，引入“最大最小原则”进行优化，此外，定义深度指标计算公式，确定 Canopy 中心点的最优个数及区域半径；接着借助三角不等式定理对 K－means 算法进行优化，减少冗余的距离计算，加快收敛速度；最后结合 MapReduce 分布式框架将改进后的算法并行化实现。在构建的微博文本数据集上进行实验，结果表明改进算法的准确率和扩展性都得到提升。

1　MapReduce 并行框架

MapReduce[6]是一种用于处理大规模数据的分布式编程模型，可以将大型任务进行拆分处理，从而加快数据的处理效率。

MapReduce 主要包括 Map 和 Reduce 两个函数，在数据处理过程中，数据均以键/值对形式保存。其中，Map 函数根据用户输入的键/值对生成中间结果，而 Reduce 函数对中间结果进行归并处理，得到的最终结果同样以键/值对形式输出。除了 Map 和 Reduce 两个核心函数外，还提供了 Combine 函数，它在 Map 后调用，相当于本地的 Reduce，主要是为了减少从 Map 到 Reduce 的数据量。

2　Canpoy－K－means 聚类算法研究与改进

2.1　Canpoy－K－means 算法研究

K－means 算法由于算法简单、易于实现等优点而被广泛使用。其基本思想是：从数据集中随机选取 K 个数据对象作为初始聚类中心点；将剩余数据对象和簇中心进行间距计算，并且把它划至间距最短的簇中，持续该过程，直到数据集为空集；然后根据簇中的数据对象计算新的聚类中心点，继续上述过程，直到簇的中心点不再发生变化或者符合停止条件，迭代才会停止，完成聚类划分。

Canpoy－K－means 算法是一种借助 Canpoy 算法改进的 K－means 算法。在 Canpoy－K－means 算法中，通过 Canpoy 算法对数据集进行“粗”聚类，得到个 Canpoy 子集，随后再以个 Canopy 子集的中心点作为 K－means 算法的初始中心点进行“细”聚类，生成聚类结果。Canpoy－K－means 算法执行步骤如下：

1）将待聚类数据集构成 *List* 集合，然后指定两个距离阈值 T_1 和 T_2（$T_1 > T_2$）；

2）随机选取 *List* 合中的一个数据对象 $\boldsymbol{P}$，构成一个新的 Canpoy，并将对象 $\boldsymbol{P}$ 从集合 *List* 中移除；

3）对于 *List* 中剩余的数据对象，计算与对象 $\boldsymbol{P}$ 之间的距离。如果间距小于 T_1，就把它分配到对象 $\boldsymbol{P}$ 所在的 Canpoy 中；如果与对象 $\boldsymbol{P}$ 的间距小于 T_2，则将它从 *List* 中删除；

4）重复步骤 2）和 3），直到 *List* 为空；

5）将形成的 Canpoy 子集数目作为 K 值，Canpoy 子集的中心点作为初始的聚类中心点进行 K－means 聚类，得到较为准确的聚类结果。

Canpoy－K－means 算法虽然解决了 K－means 算法人为指定值和初始中心点随机选取的不足，然而其也存在不足：Canopy 的初始聚类中心点随机选取和初始阈值人为指定，具有盲目性，初始阈值对聚类所得的最终结果具有显著影响，一定程度上降低了聚类结果的稳定性；另外，由于其具备较高的时间复杂度，串行执行过程时所需时间较长，算法串行执行效率较低。

2.2　Canpoy 算法改进

为了改善 Canopy 算法初始阈值人为指定以及初始中心点随机选取的不足，本文引入“最大最小原则”对其进行优化，提高算法的准确率以及聚类结果的稳定性。

基于“最大最小原则”的中心点选取方法基本思想如下：在将数据集划分为若干个 Canopy 的过程中，任意两个 Canopy 中心点之间的距离应尽可能远，即假设目前已生成个 Canopy 中心点，则处于第 $n+1$ 位的 Canopy 中心点应为其它数据点和前 n 个中心点间最短间距的最大者[7]，其公式如下：

$$\begin{cases} DistList = \min\{d_1, d_2, d_3, \ldots d_n\} \\ DistMin(n+1) = \max\{DistList\} \end{cases} \tag{1}$$

式中，d_n 表示第 n 个中心点与候选数据点的最小距离；*DistList* 表示前 n 个中心点与候选数据点最小距离的集合；*DistMin*（$n+1$）则表示集合 *DistList* 中最小距离的最大者，即 Canopy 集合 $n+1$ 的第个中心点。

基于“最大最小原则”的 Canopy 中心点选择方法，在实际应用中符合下述情况：如果中心点数量和最佳中心点的数量较为接近，此时 *DistMin*（$n+1$）具备最大的变化幅度。所以，为了确定最优的 Canopy 中心点个数及区域半径 *Depth*（i），根据参考文献[8]提出的边界思想，采用深度指标 T_1，描述 Canopy 中心点的变化幅度，如公式（2）所示：

$$Depth(i) = |DistMin(i) - DistMin(i-1)| + |DistMin(i+1) - DistMin(i)| \tag{2}$$

当 i 接近真实聚类簇数时，*Depth*（i）取得最大值，此时设置 $T_1 = DistMin$（i）使得聚类结果最优。

2.3　K－means 算法改进

传统 K－means 算法需要迭代计算数据对象与中心点的间距，完成数据对象的划分，然而在该过程中存在许多不必要的距离计算，为了减少 K－means 算法的计算量，加快算法的收敛速度，本文引入三角不等式定理对其进行优化改进[9]。

定理 1：任意一个三角形，两边之和大于第三边，两边之差小于第三边。由于欧式距离也满足三角不等式的特性，因此将其扩展到多维的欧几里得空间可知：对于欧式空间的任意向量 x、b、c，满足：$d(x, b) + d(b, c) \geqslant d(x, c)$ 和 $d(x, b) - d(b, c) \leqslant d(x, c)$ 成立。

对于任意一个向量 x 和两个聚类中心 b、c，根据三角不等式定理可得：$d(x, b) + d(b, c) \geqslant d(x, c)$，但是为了避免计算距离 $d(x, b)$，需要得到 $d(x, b) \leqslant d(x, c)$ 这个不等式关系，给出引理及其证明过程如下：

引理 1：假设 x_p 是数据集中的任意一个向量，c_i 是向量 x_p 当前的簇中心，$d(x_p, c_i)$ 已知且 c_j 是除 c_i 外的任意一个簇中心，如果 $2d(x_p, c_i) \leqslant d(c_i, c_j)$，则有 $d(x_p, c_i) \leqslant d(x_p, c_j)$。

证明：假设有 $2d(x_p, c_i) \leqslant d(c_i, c_j)$，两边同时减去，得 $d(x_p, c_i) \leqslant d(c_i, c_j) - d(x_p, c_i)$，由定理 1 可得 $d(c_i, c_j) - d(x_p, c_i) \leqslant d(x_p, c_j)$：因此可以得到结论 $d(x_p, c_i) \leqslant d(x_p, c_j)$，即向量 x_p 属于簇中心 c_i。

根据引理的推导过程可知，基于三角不等式的改进方法可以有效减少 K－means 冗余的距离计算，应用如下：已知是数据集中任意一个向量，c_i 是向量的当前簇中心，$d(x_p, c_i)$ 已知且 c_j 是另外的任一簇中心，根据引理 1 可知，如果 $2d(x_p, c_j) \leqslant d(c_i, c_j)$，则可以确定数据向量 x_p 属于簇中心 c_i，此时就不再需要计算 $d(x_p, c_j)$。

2.4 改进算法的 MapReduce 并行化实现

本文主要从两方面对 Canopy－K－means 算法进行改进，首先引入“最大最小原则”优化 Canopy 中心点的选取；接着利用三角不等式对 K－means 算法进行优化，减少冗余的距离计算，加快算法的收敛速度。改进后的算法主要分为两个阶段，其流程如图 1 所示。

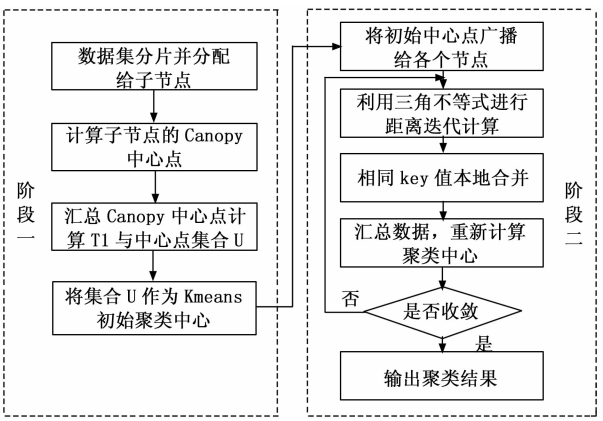

图 1 改进 Canopy－K－means 算法流程图

阶段一：基于“最大最小原则”改进的 Canopy 算法在 MapReduce 框架上的并行化实现，用来选取初始聚类中心点及 K 值。该阶段由 Map 函数和 Reduce 函数两部分完成。算法的伪代码如下：

Map 函数

输入：节点数据集合 *List*

输出：节点 Canopy 中心点集合 C_i

1) $C_i = null$

2)While ($List! = null$)

3)If ($C_i = null$)

4)在 *List* 中随机选取一个数据点作为 Canopy 中心点，保存至 C_i 中，并将该数据点从中删除

5)Else if ($C_i! = null$)

6)遍历计算中的数据点到集合 C_i 各个中心点的距离，取距离的最小值 d_n 保存到集合中

7)求出集合 D 中的最大值 $Max(D)$

8)把 $Max(D)$ 对应的数据点作为 Canopy 集合的下一个中心点，存入集合 C_i 中

9)End If

10)End While

11)output(C_i)

Reduce 函数：

输入：各个节点在 Map 阶段产生的局部中心点集合 $C\{C_1, C_2, C_3, \cdots, C_n\}$

输出：Canopy 中心点集合 U；

1)计算集合 C 中的数据总量 $K = Count(C)$ 且令 $j = 0$

2)while($j < K$)

3)计算全局 Canopy 中心点集合 C 中 $Depth(i)$ 的最大值

4)令 $T_1 = Max(Depth(i))$，j ++

5)把集合 C 中的前 i 个中心点赋值给集合 U

6)End While

7) $K = Count(U)$

8)OutPut(U)

阶段二：将阶段一得到的 Canopy 中心点作为初始中心点完成 K－means 聚类。此外，在此阶段引入三角不等式定理，减少迭代过程中不必要的距离计算。该阶段由 Map 函数、Combine 函数和 Reduce 函数三部分组成。算法的伪代码如下：

Map 函数

输入：K 值和 Canopy 中心点集合 U，数据集 $X = \{x_1, x_2, x_3, \cdots, x_n\}$

输出：聚类中心点集合 W

1)While ($W! = U$)

2)计算集合 U 任意两中心点间的距离 $d(c, c')$

3)保存最短距离 $S(c) = \min(d(c, c'))$

4)计算数据集 X 中的数据点到集合 U 中第 i 个中心点的距离 $dist[i]$

5)If ($2dist[i] \leqslant S(c)$)，则标记该数据点属于第 i 个 Canopy 中心点的簇，然后从 X 中删除该数据点；对于不符合条件的数据点，保存其到该中心点的距离

6)If ($X! = null$)

7)计算不符合条件的数据点与中心点的距离，将其划分给距离最小的簇中心并进行标记

8)计算被标记点的新簇中新 W'

9)If ($W = W'$)

10)Break

11)Else 返回 2)重新计算

12)End While

Combine 函数：

输入：X 中数据点所属簇下标 key，key 值所属的键值对列表

输出：X 中数据点所属簇下标 key，各个簇内被标记数据点的各维累加值以及值 key 所属的键值对列表；

在本地解析各维坐标值，求出各维的累加值，并保存到对应列表中。

Reduce 函数：

输入：X 中数据点所对应下标 key，key 值所属的键值对列表

输出：X 中数据点所属簇的下标 key，最终的簇心 W

1）初始化 $Num=0$，记录所属簇内数据点的个数

2）While（$X.hasNext()$）

3）解析 $X.next()$ 中的各维下标值，计算样本个数 num

4）计算各维下标值的累加和并进行存储

5）$Num+num$

6）End While

7）用各维下标的累加和除以 Num，计算新的簇中心 W

Reduce 函数结束后，对比新生成的簇心和之前的簇心是否相同，若簇中心相同，则算法结束，否则继续执行上述过程，直到簇中心不再变化。

3　实验与分析

3.1　实验环境及测试数据集

本文的 Hadoop 集群环境搭建在一台 I7CPU，16 G 内存，2 TB 硬盘服务器之上。集群包括 1 个 Master 节点和 5 个 Slave 节点，每个节点均为 2 GB 内存，200 G 硬盘，操作系统为 CentOs 6.5，jdk 为 jdk1.8.0_181，Hadoop 版本为 2.7.3，程序开发工具为 Eclipse，算法全部由 Java 语言完成。

实验的数据集是经过中文分词、去停去重和文本特征提取等预处理后的微博数据。本文共构造了 100 M、500 M、1 G 和 2 G 这 4 个数据量依次递增的微博数据集，用于改进 Canopy－K－means 算法的测试。

3.2　实验结果与分析

3.2.1　算法准确率分析

本文以准确率（precision）、召回率（recall）和 F 值作为评判指标[10]。对比传统 K－means 算法（算法 1），Canopy－K－means 算法（算法 2）以及本文改进算法（算法 3）在文本聚类上的优劣，分别在 100 M、500 M、1 G 和 2 G 数据集各聚类 10 次，取各项指标的平均值进行比较，结果如表 1 所示。

表 1　文本聚类测试结果

		100 M	500 M	1 G	2 G
算法 1	准确率	0.658	0.654	0.660	0.651
	召回率	0.645	0.641	0.648	0.639
	F 值	0.651	0.647	0.654	0.645
算法 2	准确率	0.773	0.769	0.758	0.741
	召回率	0.732	0.708	0.712	0.703
	F 值	0.752	0.737	0.734	0.722
算法 3	准确率	0.867	0.852	0.827	0.819
	召回率	0.812	0.798	0.782	0.778
	F 值	0.839	0.824	0.804	0.798

由表 1 中的测试结果可知，与常规 K－means 算法相比，Canopy－K－means 算法的准确率提升了约 10%，而本文改进算法与 Canopy－K－means 算法相比，准确率提升了约 7%。这是由于改进后的 Canopy－K－means 算法，优化了 Canopy 的中心点的选取，根据深度指标计算公式，确定了 Canopy 中心点的最优个数与最佳区域半径，从而使得聚类结果更加稳定，算法的准确率得到提高。

3.2.2　算法扩展性分析

加速比是常用来衡量程序并行化执行效率的重要指标。它的定义如下：$S_p = T_s/T_p$。此处，为在单机条件之下算法运行的具体时长，而 T_p 则是在并行条件之下算法运行的具体时长。加速比 S_p 越大，表示算法的效率越高。考虑到单机环境处理大规模数据时系统容易崩溃，因此本文以 1 个数据节点下算法的执行时长作为。

为了对比改进后算法和未改进算法在扩展性上的差异。使用 K－means 算法、Canopy－K－means 算法以及改进的 Canopy－K－means 算法分别对 1 G 的数据集进行 5 次聚类运算，取其平均运算时长，计算其加速比，测试结果如图 2 所示。

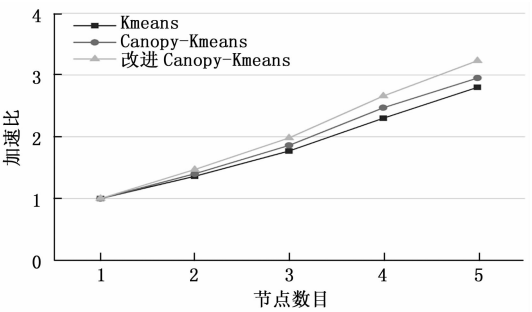

图 2　相同数据集不同算法加速比

根据图 2 可知，在相同规模节点数目下，本文改进算法的执行效率明显优于其它两种算法，这是由于“最大最小原则”的中心点选取方法优化了 Canopy 中心点的选取，减少了算法的迭代次数，并且基于三角不等式定理改进的 K－means 算法，有效减少了迭代过程中存在的冗余距离计算，算法的执行速度得到提高。

为了验证改进后算法在不同数据集上的并行执行效率，分别使用 100 M、500 M、1 G 和 2 G 这 4 个数据集，在节点个数为 1、3、5 的 Hadoop 集群上聚类 5 次，取其平均运算时长，计算加速比。结果如图 3 所示。

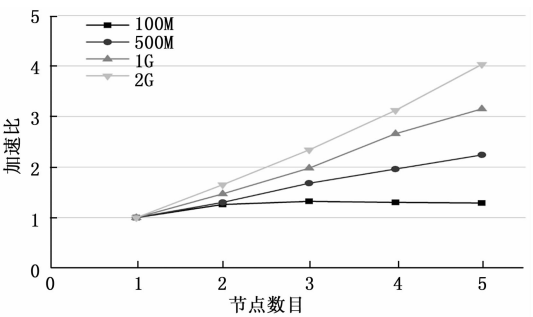

图 3　改进算法在不同数据集下的加速比

根据图 3 可知，由于 100 M 的数据集相对较小，在集群的节点为 2 时，算法的加速比有所提升，此时，数据处理时长超过节点间的通信时长；当集群节点为 3 时，算法的

（下转第 186 页）