

引用格式:柴变芳,李有熠.基于 Spark 的主动重叠 K-means 聚类算法[J].微电子学与计算机,2021,38(1):70-76. [CHAI B F,LI Y Y. Active overlapping K-means clustering algorithm based on spark[J]. Microelectronics & Computer,2021,38(1):70-76.]

基于 Spark 的主动重叠 K-means 聚类算法

柴变芳,李有熠

(河北地质大学 信息工程学院,河北 石家庄 050031)

摘 要: 别大规模数据的潜在模式.但其存在两个问题:多次迭代 Master 和 Worker 节点间数据交换,导致算法运行效率低;对初始聚类中心敏感,导致聚类结果不稳定且收敛速度慢.为提高算法运行效率和结果稳定性,提出了一种主动重叠 K-means 聚类算法.其在各个分区上执行重叠 K-means 算法获得局部聚类中心,将结果汇总回收得到 Master 节点,在 Master 节点运行重叠 K-means 算法聚合所有聚类中心,作为最终聚类中心;同时采用并行化主动选择策略获得更优的初始簇中心,提高算法准确性、收敛速度.实验结果表明,改进后的主动重叠聚类算法提高了算法准确性,降低了算法运行时间.

关键词: Spark 框架;主动学习;重叠聚类;并行计算

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2021)01-0070-07

Active overlapping K-means clustering algorithm based on spark

CHAI Bian-fang, LI You-yi

(College of Information Engineering, Hebei GEO University, Shijiazhuang 050031, China)

Abstract: Parallel Overlapping K-means clustering algorithm (POKM) based on Spark framework can effectively identify potential pattern of large-scale data. But multiple iterations of data exchange between the Master and the Worker nodes lead to low efficiency of the algorithm, and it is sensitive to the initial clustering center, resulting in unstable clustering results and slow convergence rate. In order to improve the performance and stability of the algorithm, an active overlapping K-means clustering algorithm is proposed. It performs the overlapping K-means algorithm on each worker and obtains the local cluster center, and then collects the centers and runs the overlapping K-means algorithm on the Master node. At the same time, the parallel active selection strategy is adopted to obtain a better initial cluster center to improve the accuracy and convergence speed. Experiment results show that the improved active overlapping clustering algorithm improves the accuracy and reduces the running time.

Key words: Spark framework; active learning; overlapping clustering; parallelization computing

1 引言

大规模数据聚类已经成为数据挖掘和机器学习中的一个挑战性任务,在现实生活中有广泛的应用,比如图像分割、用户细分、文本聚类.聚类分析将相似的数据对象聚到一起,发现他们的内在联系,获取数据背后的隐藏价值^[1].

目前基于大规模数据的聚类主要分为两类,一类基于串行计算环境,另一类基于并行计算环境^[2].基于串行计算环境的聚类方法有基于样例选择的方法^[3],基于增量学习的方法^[4],基于特征选择的方法^[5]和基于降维的方法^[6].串行聚类算法对原始数据集的样本空间进行压缩,使得大规模数据能够在单机环境下执行,此类方法聚类的效果依赖于压缩

后的样本空间. 此外,虽然这些算法可将大规模数据压缩,但算法输入也为大规模数据,很难实现单机存储和计算. 并行计算环境下的聚类算法主要包括:基于 MapReduce 计算框架的方法、基于 Storm 计算框架的方法和基于 Spark 计算框架的方法. 基于 MapReduce 框架的聚类算法可将大规模数据集分布到不同节点,实现 Map 计算和 Reduce 计算,研究者提出了基于 MapReduce 的 K-means、模糊 c-means、DBSCAN 等聚类算法^[7]. 基于 MapReduce 计算框架的方法每执行一次 Map 和 Reduce 计算,需要将结果写入分布式文件系统 (Hadoop distributed file system, HDFS),导致迭代算法反复读写 HDFS,效率较低. Storm 计算框架是一个分布式、高容错的实时大数据处理系统,研究者提出了基于 Storm 计算框架的 DBSCAN 聚类算法、基于 Storm 计算框架的流数据聚类算法 CluStream^[8]. Spark 计算框架是一个基于内存计算的开源集群计算框架,其可将数据分布到不同计算节点,节点计算中间结果可保存在内存中,不需要反复读写 HDFS,更适和机器学习迭代算法.

目前基于 Spark 计算框架的聚类算法已经有大量的研究^[9],如 K-means、密度聚类和谱聚类算法. 密度聚类和谱聚类算法能够识别任意形状的簇,但是计算复杂度较高. K-means 聚类算法由于其较低的计算复杂度,近年来衍生出了一系列的变体. 为了解决 K-means 只能分配给单个簇的限制,Guillaume 基于 K-means 算法提出了重叠 K-means 算法^[10],简称 OKM (Overlapping K-means Cluster). OKM 假设每个样本隶属于多个簇,通过最小化样本点到所属多个簇的簇均值的距离,实现样本的重叠划分. Abir 等针对大规模的数据集,基于 Spark 计算框架提出了并行重叠的 K-means 聚类算法,简称 POKM 算法^[11].

POKM 算法流程如图 1 所示. 该算法随机初始簇中心,执行两阶段计算:将簇中心下发到各个 Worker 节点,在各个 Worker 节点执行一次样本簇指派和局部簇中心计算;将局部簇中心回收 Master 节点,更新全局簇中心. 迭代执行这两步直至算法收敛. 在 POKM 算法的两阶段计算过程中,Master 和 Worker 节点频繁进行通信传输数据,网络开销巨大,导致算法运行时间较长. 针对此问题,在各个 Worker 节点上运行 OKM 算法,得到局部的簇

中心并将其回收 Master 节点,之后将局部簇中心作为 OKM 算法的输入,再次运行 OKM 算法得到全局的簇中心,流程如图 2 所示. 将此改进算法简称 I_POKM (Improved Parallel Overlapping K-means Cluster) 算法,该算法 Master 节点只需要回收一次局部簇中心,大大减少了网络传输开销. POKM 算法随机选择初始簇中心,聚类结果对初始簇中心比较敏感,导致算法收敛速度慢、准确性低. 为了加快收敛速度、提高聚类质量,利用主动策略改进并行重叠 K-means 聚类算法,简称 AI_POKM (Active Improved Parallel Overlapping K-means Cluster) 算法.

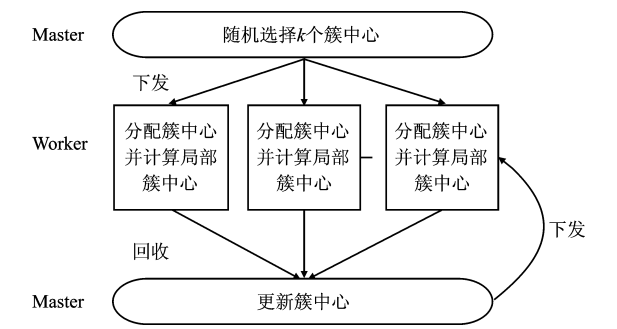


图 1 POKM 算法流程

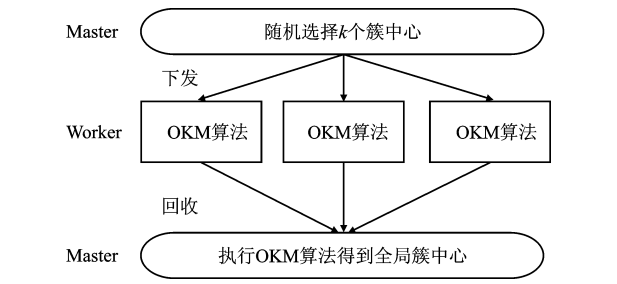


图 2 I_POKM 算法流程

2 重叠 k-means 算法 OKM

给定数据集 X , 形式化为 $[n \times d]$ 维矩阵, n 表示样本个数, d 为样本维度, 每个样本用 x_i 来表示.

OKM 算法的任务是将数据集聚类, 每个样本可隶属多个簇. 假设数据集有 k 个簇中心, 形式化为矩阵 $\mathbf{P} = [k \times d]$, 所有样本隶属各个簇的隶属度用矩阵 $\mathbf{A} = [n \times k]$ 表示, a_{ij} 表示样本 i 属于簇 j 的隶属度, 值为 0 或 1.

OKM 算法优化目标是为每个样本寻找最佳的簇指派, 通过最小化各样本到所属簇的距离的均值实现. 目标函数为:

$$j_{okm}(A, P) = \sum_{i=1}^n \|x_i - \bar{x}_i\|^2 \quad (1)$$

$$\bar{x}_i = \frac{1}{|A_i|} \sum_{P_c \in A_i} P_c \quad (2)$$

$$p_c = \frac{\sum_{x_i \in A_c} x_i}{|A_c|} \quad (3)$$

式中, $\bar{x}_i$ 表示 x_i 对应的簇均值如公式(2)所示, A_i 表示样本 x_i 所属簇集合, P_c 表示第 c 个簇中心。

OKM 算法基于迭代框架实现重叠聚类, 每次迭代包括两步计算: 为样本点分配簇中心和更新簇中心。其迭代至满足收敛条件。该算法随机初始化簇中心, 即为每个样本分配簇中心, 每个样本的分配情况有 2^k 种, 指派时间复杂度较高。为了降低指派复杂度, 采用启发式的指派策略, 具体流程如下:

(1) 随机初始化簇中心 $P = \{p_c\}_{c=1}^k$, 初始化分配矩阵 $A = \{a_i\}_{i=1}^n = 0$ 。

(2) 计算每个样本 $x_i \in \{x_i\}_{i=1}^n$ 到各个簇中心的距离, 并按从小到大的顺序进行排序。

(3) 更新分配矩阵 $A_{\text{new}} = \{a_i'\}_{i=1}^n$: 首先将每个样本分配给最近的簇, 按照公式(2) 计算簇均值, 计算样本到簇均值的距离。继续分配下一个近的簇, 并计算样本到簇均值的距离。若距离较上次小, 接着分配下一个近的簇, 直到样本所有的簇中心已经分配完成; 否则停止分配。

(4) 按照公式(3) 更新簇中心。

(5) 按照公式(1) 计算 $j_{okm'} = j_{okm}(A_{\text{new}}, P)$, 若 $j_{okm'} < j_{okm}$, 令 $A = A_{\text{new}}$, $j_{okm} = j_{okm'}$, 返回步骤(2) 继续, 否则结束算法。

OKM 算法详细设计如算法 1 所示, 为了便于描述, 采用函数式编程的方式描述, 但执行过程为串行。

算法 1 OKM(X, k)

输入: 数据 X, k 聚类中心的个数。

输出: 聚类中心 P 和隶属度矩阵 A 。

(1) 随机初始化簇中心 $P, itr_{\text{MAX}} = 1\ 000, \text{converge} = 0, itr = 0, A = \{a_i\}_{i=1}^n = 0$;

(2) $j_{okm} = \infty$;

(3) while ($itr < itr_{\text{MAX}}$) && (!converge)

(4) $A_{\text{new}} = X. \text{map}(x \Rightarrow \text{Assign}(x, P))$;

// 为每个样本分配簇中心。

(5) 按照公式(3) 更新簇中心 P ;

(6) 按照公式(1) 计算 $j_{okm'} = j_{okm}(A_{\text{new}}, P)$;

(7) if $j_{okm'} < j_{okm}$ $A = A_{\text{new}}, j_{okm} = j_{okm'}$;

(8) else converge = 1;

(9) $itr++ = 1$;

(10) Return P ;

算法 2 Assign(x, P)

输入: x 样本点, P 簇中心。

输出: a 分配数组。

(1) 初始化 $a = 0, j_{okm} = \infty, itr = 0, \text{stop} = \text{false}$;

(2) while ($itr < k$) && (!stop)

(3) 查找下一个近的簇中心 $c, a[c] = 1$;

(4) if ($j_{okm}(a', p) < j_{okm}$)

(5) $j_{okm} = j_{okm}(a', p)$;

(6) $a = a', itr++ = 1$;

(7) else stop = true;

(8) end while

(9) Return a ;

3 主动并行重叠的 K-means 算法

OKM 算法不能处理大规模数据, POKM 算法基于 Spark 框架对其进行并行计算, 如图 1 所示。但 POKM 算法需要多次迭代, 每两次迭代之间需要汇总局部簇中心和计算全局簇中心并进行下发, Master 和 Worker 节点频繁进行通信传输数据, 网络开销巨大并且聚类效果易受到随机初始化簇中心的影响。为此设计了两种改进策略:

(1) 局部聚类集成策略。在各个 Worker 节点上运行 OKM 算法, 得到局部的簇中心并将其回收到 Master 节点; 将局部簇中心作为 Master 节点 OKM 算法的输入初始簇中心, 再次运行 OKM 算法得到全局的簇中心。该策略不需要在 Worker 节点和 Master 节点间频繁交换数据, 仅交互一次数据, 提高算法运行效率。此外, 每个 Worker 节点上运行局部 OKM 算法, 最后通过 Master 节点集成聚类结果, 可保证算法性能。

(2) 初始簇中心主动选择策略。初始簇中心对 OKM 聚类算法有较大的影响, 进而会导致集成聚类结果差。在各个 Worker 节点上运行 OKM 算法前, 基于最远距离优先策略构建初始聚类中心。

3.1 基于局部聚类集成策略的 I_POKM 算法

I_POKM 算法采用分而治之的策略, 为保证各个分区上的样本分布与总体样本分布趋于一致, 引入统计感知的概念, 在 hdfs(分布式文件系统)中基于随机样本划分^[12]的大数据管理方式划分分区。I_POKM 算法首先在各个分区上运行 OKM 算法。然

后将各个分区上的局部簇中心回收到 Master 节点,将局部簇中心作为 OKM 算法的输入,再次运行 OKM 算法得到全局的簇中心. 具体流程如算法 3:

算法 3 I_POKM(path,k)
输入:数据集的路径 path,聚类中心个数 k.
输出:聚类中心 P.
(1) data = sc. textfile(path) // 读入数据集
(2) $\{P_i\}_{i=1}^{|B|} = \text{data.mapPrtition}(b \Rightarrow \text{OKM}(b,k))$.
collect() // 在各个分区上运行 OKM 算法
(3) $P = \text{OKM}(\{P_i\}_{i=1}^{|B|})$ // 再次运行 OKM 算法

3.2 基于主动选择策略的 AI_POKM 算法

为了提高聚类的效果和收敛速度,引入主动选择策略,在各个分区上基于最远距离优先策略^[13]构建初始簇中心,将加入主动选择簇中心的算法称 AI_POKM 算法,如算法 4 所示. 具体流程如下:

- (1) 假设数据集有 k 个中心,将数据集分成 m 个分区,在各个分区上独立进行(2) ~ (5) 步骤.
- (2) 初始化主动选择的结果集 $N = \{n_{i=1}^k$ 为空,首先随机选择一个样本点作为第一个簇中心并查询其标签为 t ,放入结果集 n_t 中.
- (3) 接着选择距离结果集最远的样本并查询其标签 t ,放入 n_t 中.
- (4) 重复步骤(3) 直到每个 n_t 中至少有一个样本或者达到指定的查询次数,若达到指定的查询次数, n_t 中还有为空的情况,则随机选择样本放入其中.
- (5) 对 nt 求均值得到初始聚类中心.

算法 4 active(B,k,q)
输入:单个分区上的数据 $B = \{b_i\}_{i=1}^{|B|}$,聚类中心个数 k ,查询次数 q .
输出:初始聚类中心 P .
(1) 初始化结果集 $N = \{n_t\}_{t=1}^k = \Phi, \lambda = 1$;
(2) 随机选择样本 b_i , 样本向量 $\text{vector} = \text{extractVector}(b_i)$,样本标签 $t = \text{extractLabel}(b_i)$,将 vector 加入到结果集 n_t 中;
(3) While $\lambda < q \ \&\& \ |N| < k$ do
(4) $b_i = B.\text{map}(x \Rightarrow \text{calculateDistance}(x,N).\text{min}).\text{max}$ // 选择距离最远的样本点
(5) 样本向量 $\text{vector} = \text{extractVector}(b_i)$,样本标签 $t = \text{extractLabel}(b_i)$,将 vector 加入到结果集 n_t ;
(6) $\lambda += 1$
(7) End while;
(8) if $|N| < k$ 随机选择 b_i 加入到结果集 n_t ;

(9) Return $P = \{p_t\}_{t=1}^k = \left\{ \frac{\sum x}{\frac{x \in n_t}{|n_t|}} \right\}$

算法 5 ASCenter(path,k,q)
输入:数据集的路径 path,聚类中心个数 k,查询次数 q.
输出:各个分区上初始聚类中心 P.
(1) data = sc. textfile(path); // 读入数据集
(2) $P = \text{data.mapPrtition}(b \Rightarrow \text{active}(b,k,q))$; // 在各个分区上进行主动选择
(3) Return P;

4 实验

4.1 相关说明

本部分对实验配置、数据集、对比算法进行说明. 本文实验的集群由一个 Master 节点和 4 个 Slaves 节点组成,每个节点分配 2core 和 4G 内存,集群的配置信息参见表 1. 实验所用的数据集包括模拟数据集和真实数据集,模拟数据集模拟高斯分布生成;真实数据集为 Emotions¹ 和 Scene¹,数据集详细说明见表 2.

表 1 计算节点的配置信息

项目	配置说明
Memory	32 GB
Disk	500 GB
JVM Version	Java 1. 8. 0
Hadoop Version	Hadoop 2. 7. 2
Spark Version	Saprk 2. 3. 3
Centos Version	Centos 6. 5

表 2 实验所使用的数据集

	名称	样本个数	特征数
模拟数据集	Data10M	10 000 000	11
	Data20M	20 000 000	11
	Data40M	40 000 000	11
	Data80M	80 000 000	11
真实数据集	Emotions	593	22
	Scene	2 407	294

为了验证改进后算法的效果,分别对 POKM、I_POKM 和 AI_POKM 算法在运行时间、加速比、可扩展性、规模增长性、平方误差和(SSE)、折叠比(overlap)等 6 个方面进行对比验证.

4.2 聚类性能测试

为了验证算法对大规模数据集的适应能力,使用表 2 中的 4 种模拟数据集分别对运行时间、加速比、可扩展性、规模增长性进行测试. 其中 cpu 的核心数通过 spark-submit 命令指定.

图 3~6 表示分别在 10 M,20 M,40 M 和 80 M

数据规模下,随着 cpu 核心数的变化,三种算法的运行时间比较结果.改进后的 I_POKM 算法主从节点的通讯开销降低,较原 POKM 算法运行时间大大减少,加入主动选择后的 AI_POKM 算法,由于选择了较好的初始聚类中心,导致迭代次数减少,速度更快,效率更高.

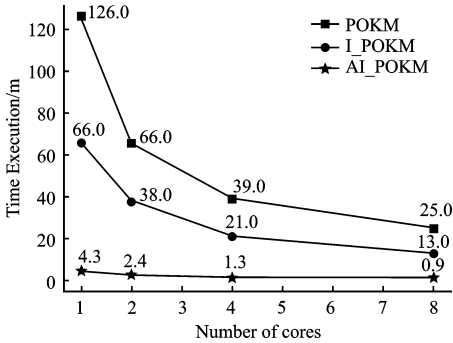


图 3 10 M 数据规模下运行时间图

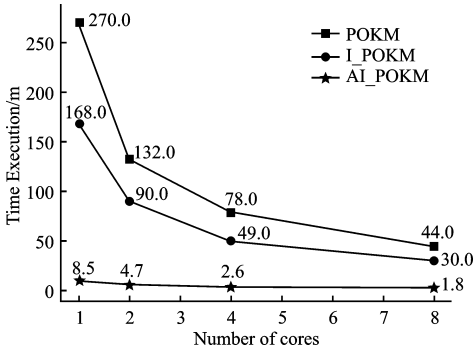


图 4 20M 数据规模下运行时间图

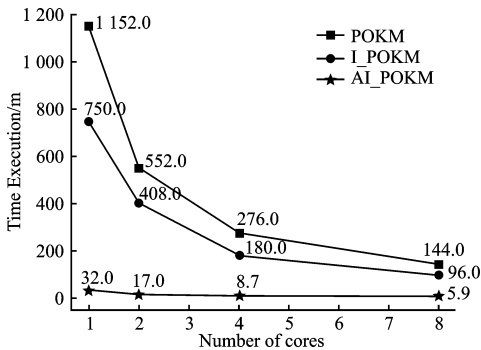


图 5 40 M 数据规模下运行时间图

图 7~9 分别在加速比、可扩展性、规模增长性方面三种算法的对比结果,原 POKM 算法优于 I_POKM 算法,加入主动学习后的 AI_POKM 算法有了一定的提升,获得了较好的加速比、可扩展性和规模增长性.

图 10 为了进一步对 AI_POKM 算法的性能进

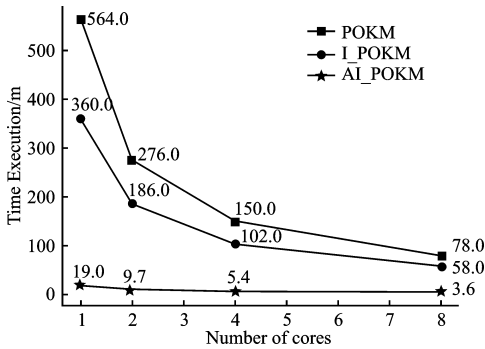


图 6 80 M 数据规模下运行时间图

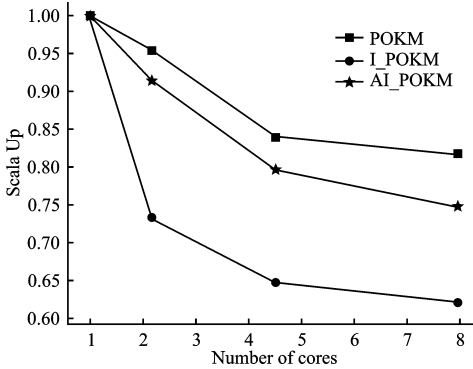


图 7 可扩展性结果图

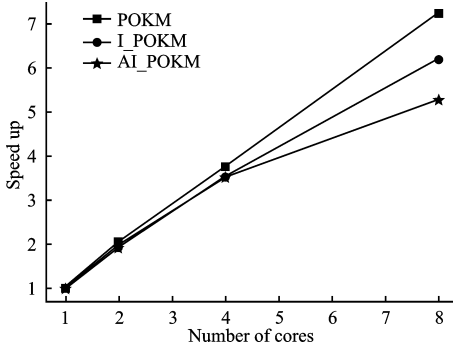


图 8 加速比结果图

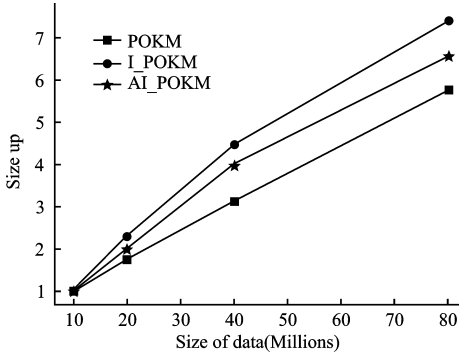


图 9 规模增长性结果图

行测试,分别在 10M,20M,40M 和 80M 数据规模下,随着 slave 节点个数的变化,对其运行时间进行比较,结果表明 AI_POKM 算法表现出良好的性能.

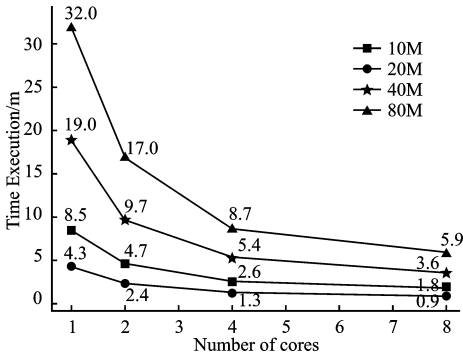


图 10 AI_POKM 随节点数目变化运行时间图

4.3 聚类效果测试

为了对算法的效果进行测试,使用两个真实的数据集对 SSE 和 overlap 进行测试,结果见表 3. 在两个真实数据集上,三种算法在 SSE 表现相当. 在折叠比 overlap 上,I_POKM 和 AI_POKM 算法要优于原 POKM 算法,更接近真实 overlap,获得了较好的聚类效果.

表 3 聚类效果测试结果

数据集	算法	k	SSE	overlap	真实 overlap
Scene	POKM	6	18 343.39	2.45	1.8
	I_POKM	6	18 355.25	1.56	1.8
	AI_POKM	6	18 314.86	1.67	1.8
Emotion	POKM	6	104 008.23	1.77	1.07
	I_POKM	6	109 795.17	1.49	1.07
	AI_POKM	6	97 700.36	1.27	1.07

5 结束语

大数据时代,大规模数据聚类在许多领域具有重要的应用价值. 文中提出一种基于 Spark 框架的主动并行重叠聚类算法 AI_POKM,该算法首先通过随机抽样,将数据分布到不同的集群节点;在各节点 Worker 节点获得其上数据的局部聚类结果;最后将局部聚类中心收集到 Master 节点,以局部聚类的中心作为初始聚类中心重新聚类,获得最终聚类结果. AI_POKM 算法利用主动学习策略和局部聚类集成策略降低算法运行时间、提高聚类准确性和稳定性. 实验中,对比了提出算法的加速比、可扩展

性、规模增长性以及聚类效果度量指标,结果表明该算法聚类性能较优. 下一步将基于此框架,设计模型和算法实现大规模文本聚类.

参考文献:

[1] JAIN A K, MURTY M N, FLYNN P J. Data clustering: a review[J]. ACM Computing Surveys, 1993, 31(3): 264-323. DOI: 10.1145/331499.331504.

[2] 何玉林,黄哲学. 大规模数据集聚类算法的研究进展[J]. 深圳大学学报(理工版), 2019, 36(1): 4-17. DOI: 10.3724/SP.J.1249.2019.01004.

HE Y L, HUANG Z X. A review on clustering algorithms for large-scale data sets[J]. Journal of Shenzhen University (Science & Engineering), 2019, 36(1): 4-17. DOI: 10.3724/SP.J.1249.2019.01004.

[3] CHEN X J, HONG W J, NIE F P, et al. Directly minimizing normalized cut for large scale data[C]//Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM, 2018: 1206-1215.

[4] ZHAO L, CHEN Z K, YANG L. Incremental CFS clustering on large data [C]//Proceedings of 2017 IEEE Global Conference on Signal and Information Processing. Montreal, Canada: IEEE, 2017: 687-690. DOI: 10.1109/GlobalSIP.2017.8309047.

[5] 杜世强. 基于维数约简的无监督聚类算法研究[D]. 兰州: 兰州大学, 2017.

DU S Q. Unsupervised clustering algorithm based on dimension reduction[D]. Lanzhou: Lanzhou University, 2017.

[6] 马国峻,王水波,裴庆祺,等. 基于主成分分析和 k-means 聚类的平行坐标可视化技术研究[J]. 网络与信息安全学报, 2017, 3(8): 18-27. DOI: 10.11959/j.issn.2096-109x.2017.00189.

MA G J, WANG S B, PEI Q Q, et al. Research on parallel coordinate visualization technology based on principal component analysis and K-means clustering [J]. Chinese Journal of Network and Information Security, 2017, 3(8): 18-27. DOI: 10.11959/j.issn.2096-109x.2017.00189.

[7] 虞倩倩,戴月明. 基于 MapReduce 的并行模糊 C 均值算法[J]. 计算机工程与应用, 2013, 49(14): 133-137. DOI: 10.3778/j.issn.1002-8331.1111-0261.

YU Q Q, DAI Y M. Parallel fuzzy C-means algorithm based on MapReduce[J]. Computer Engineering and Applications, 2013, 49(14): 133-137. DOI: 10.3778/

- j. issn. 1002-8331. 1111-0261.
- [8] 王铭坤, 袁少光, 朱永利, 等. 基于 Storm 的海量数据实时聚类[J]. 计算机应用, 2014, 34(11): 3078-3081. DOI: 10. 11772/j. issn. 1001-9081. 2014. 11. 3078.
- WANG M K, YUAN S G, ZHU Y L, et al. Real-time clustering for massive data using Storm[J]. Journal of Computer Applications, 2014, 34(11): 3078-3081. DOI: 10. 11772/j. issn. 1001-9081. 2014. 11. 3078.
- [9] MENG X R, BRADLEY J, YAVUZ B, et al. MLlib: machine learning in apache Spark[J]. The Journal of Machine Learning Research, 2016, 17(1): 1235-1241.
- [10] CLEUZIOU G. An extended version of the k-means method for overlapping clustering[C]//Proceedings of the 19th International Conference on Pattern Recognition. Tampa: IEEE, 2009. DOI: 10. 1109/ICPR. 2008. 4761079.
- [11] ZAYANI A, N' CIR C E B, ESSOUSSI N. Parallel clustering method for non-disjoint partitioning of large-scale data based on spark framework[C]//Proceedings of 2016 IEEE International Conference on Big Data (Big Data). Washington: IEEE, 2016. DOI: 10. 1109/BigData. 2016. 7840708.
- [12] 魏丞昊, 黄哲学, 何玉林. 基于统计感知的大数据系统计算框架[J]. 深圳大学学报(理工版), 2018, 35(5): 441-443. DOI: 10. 3724/SP. J. 1249. 2018. 05441.
- WEI C H, HUANG Z X, HE Y L. A statistical aware based big data system computing framework[J]. Journal of Shenzhen University (Science & Engineering), 2018, 35(5): 441-443. DOI: 10. 3724/SP. J. 1249. 2018. 05441.
- [13] BASU S, BANERJEE A, MOONEY R J. Active semi-supervision for pairwise constrained clustering [C]//Proceedings of the 4th SIAM International Conference on Data Mining. Lake Buena Vista: Society for Industrial and Applied Mathematics, 2004: 333-344.

作者简介:

柴变芳 女, (1979-), 博士, 副教授. 研究方向为机器学习、复杂网络分析. E-mail: chaibianfang@163. com.

李有熠 男, (1995-), 硕士研究生. 研究方向为机器学习、复杂网络分析.