ISSN 1004-9037,CODEN SCYCE4
Journal of Data Acquisition and Processing Vol. 33,No. 6,Nov. 2018,pp. 1077－1085
DOI:10.16337/j.1004-9037.2018.06.016

http://sjcj.nuaa.edu.cn
E-mail:sjcj@nuaa.edu.cn
Tel/Fax:＋86-025-84892742

基于 Spark 的大数据聚类研究及系统实现

王　磊[1,2,3]　邹恩岑[1,2,3]　曾　诚[4]　奚雪峰[1,2,3]　陆　悠[1,2,3]

(1.苏州科技大学电子与信息工程学院,苏州,215009;2. 苏州市虚拟现实智能交互及应用技术重点实验室,苏州,215009;3.苏州科技大学普开大数据重点实验室,苏州,215009;4. 昆山市公安局指挥中心,苏州,215300)

摘　要:传统聚类算法由于单机内存和运算能力的限制已经不能满足当前大数据处理的要求,因而迫切需要寻找新的解决方法。针对单机内存运算问题,结合聚类算法的迭代计算特点,提出并实现了一种基于 Spark 平台的聚类系统。针对稀疏集和密集集两种不同类型的数据集,系统首先采用不同策略实现数据预处理;其次分析比较了不同聚类算法在 Spark 平台下的聚类性能,并给出最佳方案;最后利用数据持久化技术提高了计算速度。实验结果表明,所提系统能够有效满足海量数据聚类分析的任务要求。

关键词:Spark;聚类;大数据

中图分类号: TP391　　**文献标志码**:A

Research and Implementation of Big Data Clustering Based on Spark

Wang Lei[1,2,3], Zou Encen[1,2,3], Zeng Cheng[4], Xi Xuefeng[1,2,3], Lu You[1,2,3]

(1. School of Electronic and Information Engineering, Suzhou University of Science and Technology, Suzhou, 215009, China; 2. Virtual Reality Key Laboratory of Intelligent Interaction and Application Technology of Suzhou, Suzhou, 215009, China; 3. Big Data Key Laboratory of PuKai, Suzhou University of Science and Technology, Suzhou, 215009, China; 4. Kunshan Public Security Bureau Command Center, Suzhou, 215300, China)

Abstract: Traditional clustering algorithms can not meet the requirements of current big data processing because of the limitations of stand-alone memory and computing power. Therefore it is urgent to find new solutions. Aiming at problems occurred in stand-alone memory calculating, combined with iterative computing features of clustering algorithms, a clustering system based on Spark platform is proposed. For the two different types of data sets, which are sparse sets and dense sets, the system firstly uses different strategies to achieve data preprocessing. Secondly, the performance of different clustering algorithms on Spark platform is analyzed and the best solution is given. Finally, the computing speed is improved with data persistence technology. Experimental results show that the proposed system can effectively meet the requirements of massive data clustering analysis.

Key words: Spark; clustering; big data

基金项目:国家自然科学基金(61673290,61750110534,61728205)资助项目;苏州市科技发展计划(SYG201707,SYG201817)资助项目。

收稿日期:2017-06-08;**修订日期**:2017-11-13

引 言

聚类是数据挖掘研究的重要方法之一。大数据聚类能有效支撑如客户群细分、文本主题发现和信息检索等大量实际应用[1]。传统聚类方法的重要假设是数据能够一次性地载入内存,然而大部分聚类算法都是迭代型算法,下一轮计算依赖于上一轮的计算结果。随着数据量的急剧增大,单机的内存和运算能力已经不能满足算法要求,需要人们利用分布式计算系统进行并行处理。Hadoop 平台的 MapReduce 计算框架在迭代时需频繁地读写磁盘,I/O 开销大,对于聚类算法效果并不理想。Spark 作为基于内存的计算框架,可将需要迭代的数据持久化到内存当中,当内存远大于待处理数据时,则无需进行 I/O 操作,在很大程度上加快了算法的执行速度。因而 Spark 基于内存的计算方式显然更加适合分布式的聚类计算。

聚类的算法众多,最为常用的是 K-means 算法[1]。K-means 算法往往能够得到局部最优,但与整体最优却相去甚远。为了提高算法的聚类效果,研究者们针对 K-means 算法提出了很多的优化算法。比如文献[3]提出了二分 K-means 算法;Steinbach 等[4]对二分 K-means 算法进行了评估,发现其聚类质量优于标准 K-means 算法,与层次聚类的聚类质量相当; Vassilvitski 和 Arthur[5]提出了 K-means++算法,其选取初始聚类中心的基本原则是使它们之间的距离尽可能的远,选择聚类中心时优先选择那些远离之前的选择点,因为通常认为好的聚类分析初始聚类中心都是相对分散的;Bahmani 等[6]则针对大数据聚类提出了一种并行化版本的 K-means++算法,称之为 K-means||。张翔等[7]和张玉芳等[8]分别基于马氏距离和取样的方式改进了 K-means 算法,两者在算法的稳定性上均获得了提升。在 K-means 算法的实现方法上,Lloyd 算法是其最为常见的实现方式[9]。文献[10]基于树形结构实现了 K-means,并将原先运行时间缩短了一到两个数量级。然而,随着大数据时代的到来,传统的实现方式在效率上的不足依旧明显。在大数据环境下,需要寻找新的解决方案[11]。张军伟等[12]基于并行的思想实现了二分 K 均值,算法取得了不错的加速比。随着并行框架 Spark[13]的出现,研究者开始依靠 Spark 优良的框架设计并行实现聚类算法。张波等[14]基于 Spark 实现了 Canopy K-means 并行算法,其相比于 K-means 并行算法更为高效,梁鹏等[15]则基于 Spark 实现了并行的模糊 C 均值算法,使得模型高效的同时能适应更多形状的数据。

本文首先给出任务定义,然后面向任务要求,提出基于 Spark 的大数据聚类系统框架,并详细讨论了系统组成模块,根据常用聚类算法的比对选择聚类效果最好的算法,以此组成所提系统中的聚类模块。

1 K-means 的优化算法

1.1 K-means++算法

一直以来,K-means 算法都是最受欢迎的数据挖掘算法之一,随着数据规模的增大,它的热度依然不减。对于 K-means 算法来说,适当的选择初始聚类中心对最终获得良好的聚类效果至关重要。为优化初始聚类中心的选择方式,Arthur 和 Vassilvitski 等[5]提出了 K-means ++算法。K-means ++算法在初始聚类中心点的选择上遵循的原则是使得各初始聚类中心之间离得尽可能的远。首先随机选取一个聚类中心点,之后以 $P(x)=\dfrac{D(x,c)^2}{\sum\limits_{x\in X}D(x,c)^2}$ 作为权重进行取点,其中 $D(x,c)^2$ 为点到离其最近聚类中心点的距离的平方,$\varphi_x=\sum\limits_{x\in X}D(x,c)^2$ 为误差平方和,离上一个被选中初始点越远的点就越容易被选中。这样选出来的点都是相对分散的。K-means++算法初始化聚类中心算法的伪代码如下所示。

算法 1 K-means++(k) 聚类中心初始化算法

1. C ←从数据集 X 中随机选择选择一个点
2. while $|C| < k$ do
3. 按照 $P(x)=\frac{D(x)^2}{\sum_{x\in X} D(x)^2}$ 的几率抽取点 $x \in X$
4. $C \leftarrow C \cup \{x\}$
5. end

1.2 K-means||算法

由于 K-means++也存在不足之处,即选取初始聚类中心点具有内在顺序,必须经过 K 步来选择 K 个合适的初始聚类中心,所以很难适用于大量数据的聚类。Bahmani 等[4]从 K-means++算法中得到灵感,提出了一种并行化版本的 K-means++算法,称为 K-means||算法。该算法先选出一个初始聚类中心并计算误差平方和 φ,之后在 $\log(\varphi)$次的循环当中每次选取 l 个点,最后对选出的 $l \cdot \log(\varphi)$ 个点进行重新聚类得到 K 个初始聚类中心,其初始化算法的伪代码如下。

算法 2 K-means||(k,l) 聚类中心初始化算法

1. C ←从数据集 X 中随机选择选择一个点
2. $\varphi \leftarrow \varnothing_x(c)$
3. for $O(\log(\varphi))$ do
4. $C' \leftarrow$ 从数据集 X 中以 $P(x)=\frac{l \cdot D(x)^2}{\sum_{x\in X} D(x)^2}$ 的几率抽取每一个点
5. $C \leftarrow C \cup C'$
6. end for
7. 将 C 中的点重新聚类成 K 类

1.3 二分 K-means 算法

有研究者针对 K-means 算法收敛域局部最小的问题提出了二分 K-means 算法[1]。算法一开始将所有数据作为一个簇,然后将簇一分为二,之后选择误差平方和(Sum of the squared error,SSE)较大的簇继续二分,不断重复,直到得到用户指定的簇数目为止。其主要执行流程如下。

算法 3 二分 K-means

1. C←将所有点作为一个簇
2. repeat
3. 从 C 中挑选出一个簇
4. for i to n
5. k-means. trian($k=2$)
6. 计算两个子簇的 SSE 的和
7. end for
8. C←选取 SSE 和最小的两个子簇
9. until C 中含有 K 个子簇

2 任务定义

本文根据第三届全国高校云计算应用创新大赛任务要求,设计基于 Spark 的大数据聚类系统实现给定数据集的聚类,本次比赛提供的数据集为 KDD10%和 Tr11 数据集,其具体信息如表 1 所示。

表 1 两测试数据集的规模

Tab. 1 Size of two data sets

数据集名称	来源	实体数/个	特征数/个	标签集规模/个
KDD10%	UCI	489 843	41	23
Tr11	TREC	414	6 429	9

KDDC10%数据集是 KDD 竞赛在 1999 年举行时采用的数据集,它为密集数据集,每个数据用 41 个特征来描述。Tr11 是来自于 TREC 的一个文本数据集,它是个稀疏集,每项数据有 6 429 个特征数。Tr11 特征值向量的维度较 KDD10%要高出许多,但就数据规模,KDD10%则要远超 Tr11,可见两项测试数据之间差别较大,有利于系统性能的测试。

3 系统设计与实现

3.1 聚类系统框架

针对上述数据聚类任务,设计了基于 Spark 平台的聚类系统 KCluster。图 1 是该系统的结构框图,主要包括 3 个部分:数据的预处理、数据内存持久化及聚类分析模块。

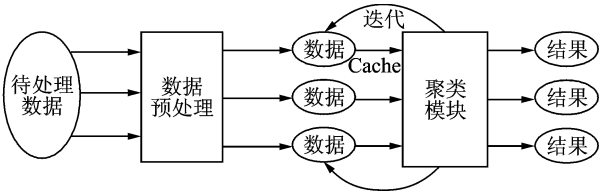


图 1 KCluster 系统框图

Fig. 1 KCluster system diagram

KCluster 系统的主要执行流程如图 2 所示。通过 Spark Context 的 textFile 方法读入存储于 HDFS 当中的数据,也就创建了一个弹性分布式数据集(Resilient distributed dataset,RDD),直接读入的数据为 RDD[Array(String)]类型。作为 Spark 的核心,PDD 是一个高度抽象的分布式集合,能像本地集合一样被操作;可通过 Parallelize 从普通集合中创建,也可以从 Hadoop 文件系统(如 HDFS, HIVE 等)创建而来。

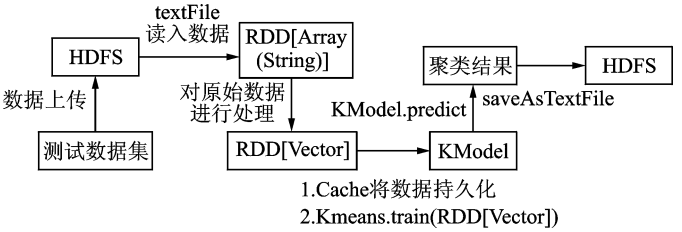


图 2 KCluster 系统流程图

Fig. 2 Flow chart of KCluster system

随后,对数据进行预处理,其处理过程也是 KCluster 系统的设计重点,将数据包装到 RDD[Vector]当中之后通过 Train 方法开始聚类模型的训练,Train 方法最后会返回 KModel 对象,通过该对象的 predict 方法完成类标记的预测。最后,通过 saveAsTextFile 算子将聚类结果保存到 HDFS 当中。

3.2　数据预处理

数据的预处理是 KCluster 系统中的重要一环,数据集读入之后,需要进行裁剪,裁剪步骤如下:(1)将数据按照空格切分;(2)去除数据中的空字符;(3)转换数据类型。其中的重点在于数据类型转换。

在完成切分和去除空字符之后得到的数据类型为 RDD[Array(String)], train 方法需要的输入数据类型为 RDD[Vector],这就需要将数据包装到 RDD[Vector]当中 ,Spark 的 MLlib 包提供了 Vector(向量)的定义接口,用它可以实现向量的定义。

第 1 项测试数据 KDD10%是一个密集型的数据集,即数据的特征值全部给出。对于密集型数据集调用 Vectors 包中的 dense 方法。dense 方法接收一个参数 values,values 为 Double 类型的数组。所以对于 KDD10%数据集,先将字符串转换为 Double 类型的数,得到 RDD[Array(Double)],再将数组传给 dense 方法。

第 2 项测试数据 Tr11 是稀疏型数据集,数据集中存在大量为零的特征值,为节省空间并提高程序运行时间,数据集中只存放非零特征值。对于稀疏型数据调用 Vectors 包中的 Sparse 方法。

定义稀疏向量,输入参数为两个 size 和 elements,size 为数据集的大小。elements 类型为 Seq[(Int, Double)],即(非零特征值的索引,非零特征值)组成的序列。针对给定的 Tr11 数据集的数据形式为特征索引、特征值交替出现,设计了 arrToSeq 算法来处理稀疏数据集,其流程如图 3 所示。

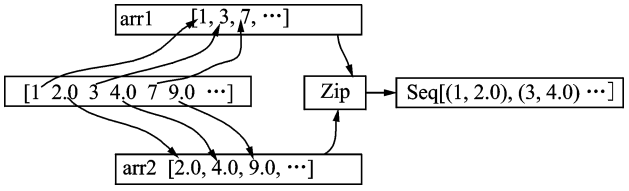


图 3　arrToSeq 算法流程图

Fig. 3　Flow chart of arrToSeq algorithm

函数的基本思路是先将索引和特征值进行分离,分别放入两个数组当中,再使用 RDD 的 zip 算子,zip 算子可将两数组合并,两数组中的元素也会一一对应形成元组。算法流程如下所示:

算法 4　arrToSeq

```
1. for 0 to arr.length   step 2
     arr1← 将索引为偶数的字符串转换为 Int 类型的数据
   end for
2. for 1 to arr.length   step 2
     arr2← 将索引为奇数的字符串转换为 Double 类型的数据
   end for
3. arr1.zip(arr2).toSeq
```

3.3　聚类模块

重点比较标准 K-means,K-means||及二分 K-means 算法在并行化平台中的聚类性能,并选择性能较好的算法用作数据聚类模块。

3.4　集群部署

通过 sbt 将程序及其依赖包打包成一个 Jar 包,启动 HDFS 及 Spark 集群,将测试数据上传到 HDFS,通过 Spark-Submit 将 Jar 包和所需的参数提交给集群,待程序执行完毕即可从指定的 HDFS 输出路径中下载结果。

4 实验与分析

KCluster 系统在聚类模块分别采用标准 K-means, K-means|| 以及二分 K-means 算法实现数据聚类，并采用算法的聚类时间、误差平方和和归一化互信息 3 个指标评估聚类算法性能。

4.1 实验环境

整个测试在由 3 台 Dell PowerEdge R720xd 服务器组成的分布式集群上完成，单节点核心数为 32 个，内存为 62 GB。程序通过 Scala + sbt 实现。Ubuntu 版本为 16.04，Hadoop 版本为 2.7.2，Spark 版本为 1.6.2，java 版本为 1.7.0_80，Scala 版本为 2.10.4。

4.2 数据持久化

为进一步提升算法性能，充分利用 Spark 基于内存的特点，将要重复使用的数据持久化到内存当中，数据无需落地磁盘，减少了大量的 I/O 操作。数据的持久化通过 cache 实现，每一个 RDD 都可以用不同的保存级别进行保存，cache 是使用默认存储级别的快捷方法。当缓存了一个 RDD，每个节点就会缓存该 RDD 的所有分区，这样数据一直在内存中进行计算，使得以后在 RDD 上的动作更快（通常提高 10 倍左右）。当聚类分析完成之后，迭代计算所需的数据还缓存在内存当中，这部分空间不会自动释放，通过 unpersist 算子可以释放这部分内存，节约内存资源。

在 KDD10% 和 Tr11 数据集上进行聚类测试，结果如表 2 所示。当未将数据持久化到内存当中时，KDD10% 数据迭代一共花费了 121 s。当使用 cache 将数据持久化到内存当中后，数据迭代只用了 14 s，仅约为原来未作内存持久化的 11.6%。

表 2 cache 前后聚类时间对比

Tab. 2 Clustering time comparison before and after cache

数据集	迭代时间（未持久化数据到内存）/s	迭代时间（持久化数据到内存）/s
KDD10%	121	14
Tr11	2.75	0.74

4.3 聚类评估指标

4.3.1 误差平方和

衡量簇的质量通常用 SSE 来度量。在执行聚类分析后，对每个点都计算一个误差值，该误差值为非质心点到最近的质心的距离，各点到其所在簇中心的欧氏距离的和即为误差平方和，x_i 为样本点，C 为样本聚类中心点的集合，D 表示聚类完成后被聚类在一个簇中的所有点的集合，SSE 的定义如式（1）所示。误差平方和越小说明簇内部越紧密，聚类效果越好。通过 KModel 的 computeCost 方法可以得到聚类结果的误差平方和的值。

$$\mathrm{SSE}=\sum_{i=1}^{k}\sum_{x_i\in D}\mathrm{dist}(x_i,c)^2 \tag{1}$$

4.3.2 归一化互信息

误差平方和的大小并不能完全反映出算法聚类质量的好坏，因此本文还通过计算簇标签与真实类标签之间的归一化互信息（Normalized mutual information，NMI）来衡量算法的聚类准确度。用 $\boldsymbol{X}$ 表示 KCluster 系统聚类分析后所得类标记的隶属矩阵，$\boldsymbol{Y}$ 表示真实类标记的隶属矩阵，这两种变量之间的归一化互信息定义可表示为

$$\mathrm{NMI}(\boldsymbol{X},\boldsymbol{Y})=\frac{I(\boldsymbol{X},\boldsymbol{Y})}{\sqrt{H(\boldsymbol{X})*H(\boldsymbol{Y})}} \tag{2}$$

$$U(\boldsymbol{X},\boldsymbol{Y})=2R=2\,\frac{I(\boldsymbol{X},\boldsymbol{Y})}{H(\boldsymbol{X})+H(\boldsymbol{Y})} \tag{3}$$

$$I(\boldsymbol{X},\boldsymbol{Y})=\sum_{y\in Y}\sum_{x\in X}p(\boldsymbol{X},\boldsymbol{Y})\log_2\frac{p(x,y)}{p(x)p(y)} \tag{4}$$

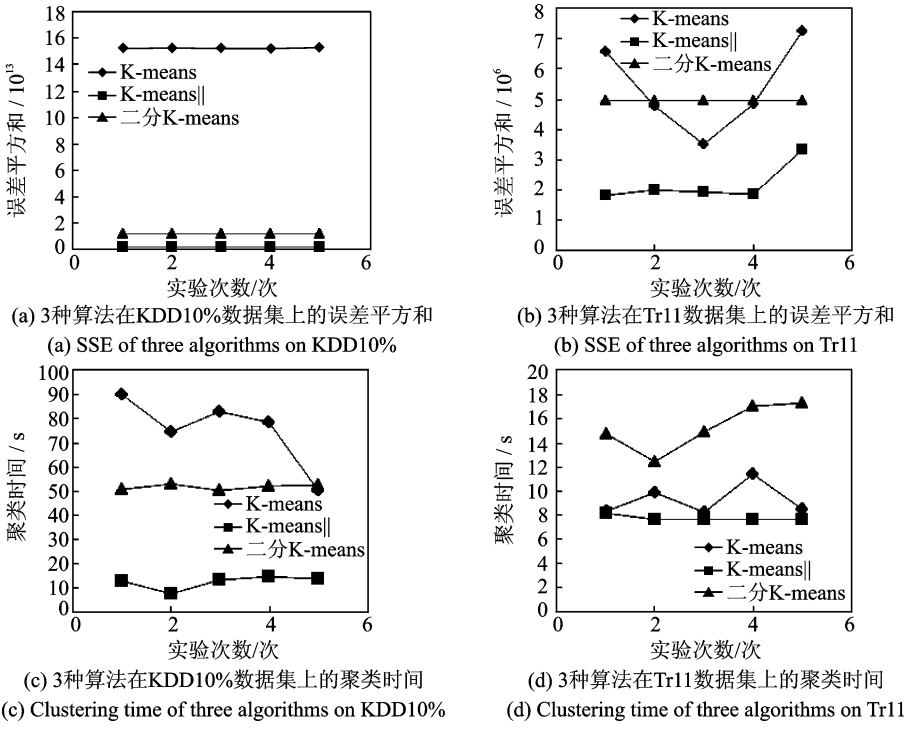
$$H(\boldsymbol{X})=\sum_{i=1}^{n}p(x_i)I(x_i) \tag{5}$$

式中：$I(\boldsymbol{X},\boldsymbol{Y})$ 为 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 之间的互信息，$H(\boldsymbol{X})$ 和 $H(\boldsymbol{Y})$ 为信息熵，用于对互信息归一化，使其位于区间[0,1]内。归一化互信息有几种不同的实现方式，但是大体的思想都是用熵做分母使 NMI 介于 0 与 1 之间。一个比较多见的实现如式(3)所示。

如果聚类结果与真实的类标记完全吻合，则 NMI 值为 1；如果数据的聚类效果很差，则 NMI 值趋近于 0。将聚类结果的矩阵和类真实标记的矩阵代入式(2)，即可求得 NMI 值。

4.4　实验结果

图 4 为系统采用标准 K-means 算法、二分 K-means 算法及 K-means||算法分别对两测试数据集进行 5 次聚类的结果。从图 4(a,b)中可以看到二分 K-means 算法在三者中稳定性最好，多次计算的聚类结果的误差平方和几乎不发生改变；K-means||算法稳定性仅次于二分 K-means 算法；标准 K-means 算法聚类的结果波动较大。观察图 4(c, d)，K-means||算法的聚类时间最短，效率最高，而 K-means 算法的效率最差。



(a) 3种算法在KDD10%数据集上的误差平方和
(a) SSE of three algorithms on KDD10%
(b) 3种算法在Tr11数据集上的误差平方和
(b) SSE of three algorithms on Tr11
(c) 3种算法在KDD10%数据集上的聚类时间
(c) Clustering time of three algorithms on KDD10%
(d) 3种算法在Tr11数据集上的聚类时间
(d) Clustering time of three algorithms on Tr11

图 4　3 种算法在两数据集上进行 5 次聚类结果

Fig. 4　Five clustering results of three algorithms on two data sets

表 3 给出了 3 种算法 5 次聚类结果的平均值，从中可以看到 K-means||算法在 KDD 数据集上的聚类时间大约为 K-means 算法的 1/6，为二分 K-means 算法的 1/4。其原因可能在于二分 K-means 通过多次二分实验的方式寻找最优解，使得它在对大规模数据进行聚类时用时较多。从表中还可以看到 K-means||算法聚类结果的误差平方和远小于其他两种算法，结果比标准 K-means 和二分 K-means 的结果要小约一半，甚至一个数量级。

表 3 3 种算法 5 次聚类结果的平均值

Tab. 3 Average values of five clustering results for three algorithms

聚类算法	数据集	聚类时间/s	误差平方和	NMI
标准 K-means	KDD10%	64.0	1.52×10^{14}	0.640
	Tr11	8.2	5 375 008	0.101
K-means||	KDD10%	19.4	1.63×10^{12}	0.694
	Tr11	7.7	2 163 348	0.060
二分 K-means	KDD10%	43.56	1.15×10^{13}	0.510
	Tr11	15.29	4 955 955	0.118

按照数据标签的规模，将 KDD10%数据集分为 23 类，Tr11 数据集分为 9 类。从表 3 中可以看到 KDD10%数据集的聚类效果良好，其中 K-means||算法的 NMI 值接近 0.7，但 3 种算法对于 Tr11 数据集的聚类效果非常不理想，NMI 值都趋于 0。经过分析，发现 Tr11 作为一个文本数据集，其维度很高、数据很稀疏，所以在高维空间中，数据都聚集在一起，若将数据分为 9 类，数据间虽然有距离，但是距离很小无法将其分开。于是尝试通过调节聚类中心数来改善聚类效果，改善后的实验结果如表 4 所示。最终发现 3 种算法均在数据集被分为 60 类左右时，NMI 值达到局部最高，聚类效果提升明显。由实验结果可见，K-means||算法在 NMI 值、聚类时间及误差平方和 3 个评价指标上，均优于其他两种算法。因此，本文设计的大数据聚类系统 KCluster 选择 K-means||算法实现聚类处理。

表 4 数据分为不同类簇时的 NMI 值

Tab. 4 Evaluation of NMI when data are divided into different numbers of clusters

聚类算法	聚类数/个					
	9	30	40	50	60	70
标准 K-means	0.101	0.140	0.201	0.344	0.373	0.368
K-means||	0.060	0.171	0.250	0.359	0.412	0.390
二分 K-means	0.118	0.324	0.377	0.405	0.408	0.390

5 结束语

本系统设计的关键点在于数据的预处理以及聚类性能的优化。经过分析，针对不同类型的初始数据集采用不同的处理策略进行了预处理，在数据聚类阶段采用 K-means||算法；同时，利用 Spark 基于内存的优点，将数据持久化到内存当中，使运行效率得到了进一步的提升。本系统预处理之后的数据同时也适用于其他聚类算法的分析处理，系统具有一定的通用性。

下一步将通过对 Tr11 等高维数据集进行降维处理，尝试提升系统对高维稀疏数据的聚类性能，同时也将继续着力于 K-means 算法的优化工作，深入研究 Spark 平台特性，进行参数调优。

参考文献：

[1] Berkhin P. A survey of clustering data mining techniques [M]. New York: Springer, 2006:25-71.

[2] Mucherino A, Papajorgji P J, Pardalos P M. Introduction to data mining [M]. Delhi:Pearson Education India, 2007.

[3] Savaresi S M, Boley D L. On the performance of bisecting K-means and PDDP [C]// Proceedings of the First SIAM International Conference on Data Mining. Chicago, USA: [s. n.], 2001:1-14.

[4] Steinbach M, Karypi G, Kumar V. A comparison of document clustering techniques [C]// In KDD-2000 Workshop on Text Mining. Boston, MA:[s. n.], 2000:525-526.

[5] Arthur D, Vassilvitskii S. K-means++: The advantages of careful seeding [C]// Eighteenth ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, USA: SIAM, 2007:1027-1035.

[6] Bahmani B, Moseley B, Vattani A, et al. Scalable K-means++ [J]. Proceedings of the VLDB Endowment, 2012, 5(7):

622-633.

[7] 张翔,王士同. 一种基于马氏距离的可能性聚类方法[J]. 数据采集与处理,2011,26(1):101-105.
Zhang Xiang, Wang Shitong. Mahalanobis distance-based possibilistic clustering algorithm and its analysis [J]. Journal of Data Acquisition and Processing, 2011, 26(1):101-105.

[8] 张玉芳,毛嘉莉,熊忠阳. 一种改进的 K-means 算法 [J]. 计算机应用,2003,23(8):31-33.
Zhang Yufang, Mao Jiali, Xiong Zhongyang. An improved K-means algorithm [J]. Computer Applications, 2003, 23(8): 31-33.

[9] Ostrovsky R, Rabani Y, Schulman L J, et al. The effectiveness of Lloyd-type methods for the K-means problem [J]. Journal of the ACM, 2013,59(6):1-22.

[10] Alsabti K, Ranka S, Singh V. An efficient K-means clustering algorithm [J]. Proceedings of IPPS/SPDP Workshop on High Performance Data Mining, 1998:43.

[11] Franks B. Taming the big data tidal wave: Finding opportunities in huge data streams with advanced analytics [M]. New Jersey: John Wiley & Sons, 2012:1-85.

[12] 张军伟,王念滨,黄少滨,等. 二分 K 均值聚类算法优化及并行化研究[J]. 计算机工程,2011,37(17):23-25.
Zhang Junwei, Wang Nianbin, Huang Shaobin, et al. Research on bisecting K-means clustering algorithm optimization and parallelism [J]. Computer Science, 2011, 37(17):23-25.

[13] 胡俊,胡贤德,程家兴. 基于 Spark 的大数据混合计算模型 [J]. 计算机系统应用,2015,24(4):214-218.
Hu Jun, Hu Xiande, Cheng Jiaxing. Big data hybrid computing mode based on Spark [J]. Computer System and Application, 2015, 24(4):214-218.

[14] 张波. 基于 Spark 的 K-means 算法的并行化实现与优化[D]. 武汉:华中科技大学,2015.
Zhang Bo. The parallelization and optimization of K-means algorithm based on Spark [D]. Wuhan: Huazhong University of Science and Technology, 2015.

[15] 梁鹏. 基于 Spark 的模糊 C 均值聚类算法研究 [D]. 哈尔滨:哈尔滨工业大学,2015.
Liang Peng. Research on Spark oriented fuzzy C-means clustering algorithm [D]. Haerbin: Harbin Institute of Technology, 2015.

作者简介:

王磊(1993-),男,硕士研究生,研究方向:机器学习与数据挖掘,E-mail:531453268@qq.com。

邹恩岑(1985-),硕士,研究方向:大数据与云计算,E-mail:zouencen@qq.com。

曾诚(1986-),男,本科,研究方向:情报检索。

奚雪峰(1978-),通信作者,副教授,硕士生导师,研究方向:自然语言处理,机器学习,E-mail:xfxi2009@qq.com。

陆悠(1977-),博士,副教授,研究方向:下一代网络、用户行为分析。

(编辑:王静)