基于改进K-means算法的海量数据分析技术研究

李 欢,刘 锋,朱二周
(安徽大学 计算机科学与技术学院,安徽 合肥 230601)

摘 要:针对海量数据难处理的难题,利用Hadoop平台下的Map-Reduce模型,实施了一种改进的对海量数据进行并行处理的K-means算法.为了解决传统的K-means算法对初始聚类中心和聚类数敏感的问题,改进算法首先对海量数据进行多次采样,找出采样数据的聚类个数;其次,利用密度法找出采样数据的聚类中心;最后,将各个样本中心点归并得到原始数据的全局初始聚类中心点.通过在Hadoop集群上部署的实验结果表明,改进后的算法相比较于传统的算法具有高效、准确、可扩展以及良好的加速比等特性.

关键词:Map-Reduce;K-means;并行挖掘
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1000-7180(2016)05-0052-06
DOI:10.19304/j.cnki.issn1000-7180.2016.05.011

Research of an Impoved K-means Algorithm for Aanalyzing Mass Data

LI Huan, LIU Feng, ZHU Er-zhou
(School of Computer Science and Technology, Anhui University, Hefei 230601, China)

Abstract: Aiming at solving the problem of mass data, this paper proposes an improved K-means algorithm for processing massive data by making use of the Map-Reduce model on the Hadoop platform. In order to solve the problem that faced by traditional K-means algorithm, such as it is sensitive to initial clustering center and clustering number, the improved algorithm firstly finds out the clustering number from sampling data by implementing multiple sampling of massive data; Secondly, with the help of density method the clustering center of data sampling is founded. Finally, the global initial clustering centers of original data are obtained by merging the central points of each sample. The results of the experiments deployed on the Hadoop cluster have shown that the improved algorithm is more efficient, accurate, scalable and has better acceleration ratio than the traditional algorithms.

Key words: Map-Reduce;K-means;parallel mining

1 引言

当今,世界已经进入大数据[1]的时代.为了从海量的数据中发现有价值的信息,急需一些有效的数据挖掘方法,以便从中发掘出潜在的规律.

因为数据量非常大,传统的数据挖掘算法无法在有效的时间内处理海量数据[2].由于并行环境在处理海量数据时具有很高的效率[3],所以本文研究利用Hadoop并行化计算平台[4]对海量数据进行处理.Hadoop是一种分布式系统架构,其中Map-Reduce编程模型把数据的处理任务分配给集群中每个节点,因而实现了数据的并行化处理[3].

在传统的数据挖掘算法中,聚类分析方法得到了广泛的应用[5].K-means算法是聚类算法中主要算法之一,它是一种基于划分的聚类算法.传统的K-means聚类算法具有聚类个数难以确定和初始聚类中心难以选取两大缺点.而这些缺点直接导致了它们在面对海量高维数据的聚类时呈现出准确性差、效率低、扩展性差等现象[3],故传统的算法已经很难满足现实的需要.

收稿日期:2015-08-04;修回日期:2015-09-15
基金项目:国家自然科学基金项目(61300169)

本文针对传统 K-means 算法的不足，提出了一种改进的 K-means 算法[6].

2　改进的 K-means 算法

K-means 算法的基本步骤是：(1)选择 k 个数据对象代表聚类中心；(2)将整个数据集的每一个数据对象分配到离簇中心最近的簇中；(3)计算每个簇的新的中心，即取属于该簇的所有数据对象的平均值；(4)如果至少有一个中心发生了变化，则转到步骤(2)，否则转到步骤(5)；(5)输出所有簇.

在传统的 K-means 算法中，初始聚类中心是随机选择的，聚类容易陷入局部僵局，并且算法还具有聚类个数难以确定的缺点. 本文针对传统算法的两大缺点，提出了基于采样和密度的改进算法. 首先对海量数据进行多次采样，然后利用密度法[7]找出采样数据的中心点，根据样本中心点确定原始数据的聚类个数 k 值和全局初始中心点.

定义 1　数据集合 $S=\{X_1, X_2, \cdots, X_n\}$, $X_i=(X_{i1}, X_{i2}, \cdots, X_{it})$.

集合 S 中有 n 个数据对象，每个数据对象的维数为 t.

定义 2　$C_k(k=1, 2, \cdots, K)$ 表示初始的 K 聚类中心.

定义 3　2 个 t 维数据对象 X_i 和 X_j 间的欧式距离 $d(X_i, X_j)$：

$$d(X_i, X_j)=[(X_{i1}-X_{j1})^2+(X_{i2}-X_{j2})^2+\cdots+(X_{in}-X_{jn})^2] \tag{1}$$

定义 4　对象的邻域：对于任意样本对象 X_i，以 X_j 为中心，R 为半径的圆域，记为 $w=\{X_j \mid 0<d(X_i, X_j)<R\}$.

定义 5　高密度邻域：对象的邻域半径为 R 时邻域里面的数据对象个数大于 M 时，此邻域就为高密度邻域.

2.1　确定聚类个数 k 值

多次通过对海量数据进行采集，生成能反映海量数据分布形态的样本. 对于采样数据，通过密度法确定样本中心点，根据各个样本中心点确定原始数据的全局中心点. 通过采样和密度确定了中心点，这就解决了原始 K-means 算法依赖于初始中心点的缺陷. 该部分的主要算法如下：

(1) 根据海量数据的形态对其多次采样，得到 N 个样本，每个样本中有 S 个数据对象；

(2) 计算每个样本 N_i 中 S 个数据对象之间的距离 $d(X_i, X_j)$，用矩阵 $\boldsymbol{D}_{s\times s}$ 存储，每一列表示其中一个数据对象到其他数据对象的距离；

(3) 找出距离的最小值 $\min d(X_i, X_j)$；

(4) 计算 $\boldsymbol{D}_{s\times s}$ 中第 i 列的平均值 R_i，如式(2)所示；

$$R_i=\frac{\sum_{j=1}^{S} d(X_i, X_j)}{S-1} \tag{2}$$

(5) 计算出矩阵 $\boldsymbol{D}_{s\times s}$ 每一列的平均值 A_i；

(6) 遍历 $\min A_i$ 所在的列，如果 $\min d(X_i, X_j)<\min R_i$，$k$ 值加 1；

(7) 利用密度法，找出每次采样数据的中心点；

(8) 归并局部中心点得到全局中心点，作为初始聚类中心点.

2.2　密度法选择初始聚类中心

数据集合中某一数据对象所处邻域内的数据对象越多，邻域内其他对象到该对象的距离越小，说明该数据对象的密度越大. 如果将该数据对象作为聚类中心，就能够较好地反映数据分布的特征[8].

计算初始数据集 S 中所有数据对象之间的平均距离 $\mathrm{Mean}(S)$：

$$\mathrm{Mean}(S)=\frac{1}{n(n-1)}\sum d(X_i, X_j) \tag{3}$$

按照式(4)计算对象 X_i 的密度 $\mathrm{Den}(X_i)$：

$$\mathrm{Den}(X_i)=\sum_{j=1}^{n}\varphi(d(X_i, X_j)-\mathrm{Mean}(S)) \tag{4}$$

$$\varphi(x)=1, \mathrm{if}(x>0)$$
$$\varphi(x)=0, \mathrm{if}(x<0) \tag{5}$$

上式表示如果对象 X_i 到其他对象 X_j 的距离 $d(X_i, X_j)$ 比平均距离 $\mathrm{Mean}(S)$ 小，那么对象 X_i 的密度加一.

将 S 中所有数据对象按照其密度从大到小排序，取密度最大的数据对象作为第一个初始聚类中心 C_1，加入集合 T(聚类中心存储集合)，然后从集合 S 中删除该数据对象. 并且根据高密度邻域定义计算高密度点的集合 M，然后取 M 中距离 C_1 最远的点作为第二个初始聚类中心 C_2，加入集合 T，同时从集合 S 中删除. 然后计算每个对象 X_i 到中心 C_1 和 C_2 的距离 $d(X_i, C_1)$ 和 $d(X_i, C_2)$，搜索集合 T 中每个点距离乘积最大的点，将其作为下一个聚类中心，并且加入集合 T，同时从集合 S 中删除. 那么第三个中心 C_3 为使得 $d(X_i, C_1)\times d(X_i, C_2)$ 取得最大值的数据对象 X_i. 以此类推，第 k 个中心 C_k 为满足 $\max(d(X_i, C_1)\times d(X_i, C_2)\times\cdots\times d(X_i, C_{k-1}))$ 的数据对象 X_i.

3 基于 Map-Reduce 的 K-means 算法具体实现

3.1 基于 Map-Reduce 的改进 K-means 算法的流程

基于 Map-Reduce 的改进的 K-means 算法的具体流程如图 1 所示. 主要有三个过程：map 过程，combine 过程，reduce 过程. map 过程的任务是从中心点集合中找出离数据对象最近的中心点；combine 过程的任务是从 map 过程的结果中找出属于同一中心点的数据对象并求和；reduce 过程将 combine 过程的局部结果汇总并计算所有簇的新中心. 然后判断算法是否满足收敛条件，如果满足则聚类结束，如果不满足则回到 map 过程重新执行.

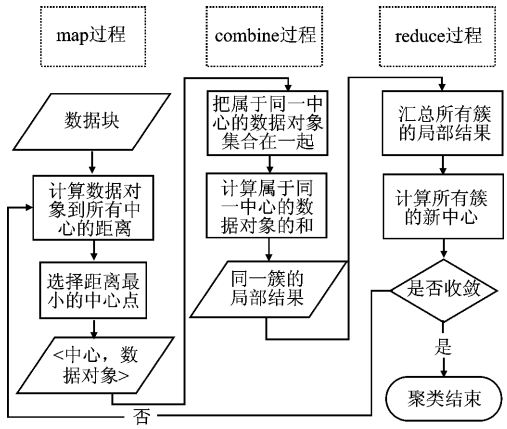

图 1 并行算法流程图

3.2 具体实现

3.2.1 初始中心选取的 Map-Reduce 并行过程

可以按照串行顺序执行初始聚类中心的选取过程，但在数据量较大的情况下，该过程非常耗时，因此利用 Hadoop 平台的并行化特点对数据进行处理，提高求解初始中心的效率. 首先设计了 InitialMap 函数和 InitialReduce 函数.

InitialMap 函数输入为<key，value>对，此处 key 值为当前行对于起始行的偏移量，value 为采样数据 N_i 中数据对象 X_i 的坐标信息. 当前行对于起始行的偏移量作为 key'，对象 X_i 的密度 Den(X_i) 作为 value'. 函数伪代码如下所示：

```
InitialMap(key, value, key', value'){
//从 value 中取出 S 个样本对象 Xi，写入 instance 对象；
Instance instance = new Instance(value.toString());
初始化变量 min，k；
for(i=1;i<=S;i++)
for(j=1;j<=S;j++)
{Dij = d(Xi,Xj);if(Dij <min) min=Dij ;}
Ai =mean(Dij );
while(i<column(minAi))
if(min d(Xi,Xj)<minRi) k++；
利用密度法，找出采样数据的中心点；
中心点标识符 key'，中心点的信息作为 value'；
输出<key', value'>.}
```

InitialReduce 函数输入是<key，value>对，即 InitialMap 函数的输出，Reduce 函数对局部中心点进行归并. 函数伪代码如下所示：

```
InitialReduce(key, value, key', value'){
//从 value 中取出各局部中心点；
Instance instance = new Instance(value.toString());
找出局部中心点中不属于同一个高密度邻域点作为全局中心点；
中心点标识符 key'，中心点的信息作为 value'；
输出<key', value'>.}
```

自此，得到了原始数据的聚类个数和初始聚类中心，解决了海量数据聚类个数和初始聚类中心难以确定的问题. 然后实现了 K-means 算法的并行化过程，并行化由 map，combine，reduce 三个部分构成，每个部分都包含了一些类，下面列出了主要的类，并且描述了它们的作用和它们包含的函数.

3.2.2 主要的类

(1)KMeansDriver

KMeansDriver 负责组织 map 和 reduce 操作. 首先需要初始化程序参数. 程序的参数有初始数据的路径、聚类结果存储的路径、初始聚类中心，最大迭代次数、收敛参数、聚类个数等. 每次迭代时定义一个 Job，将每次迭代产生的聚类中心存储到 HDFS 文件上，并将聚类中心的字符串放到 configuration 中，方便下一次迭代时使用.

(2)Cluster

这个类主要表示一个簇，描述了这个簇的数据个数和中心. 并且定义了一个方法 loadcenter()，该方法从每次迭代的质心文件中读取质心，并返回字符串.

(3)KMeansMapper

该类继承于 Mapper 类，定义了 setup()和 map()方法. Setup() 方法处在 map()方法之前，用于初始化数据，在这里 setup()将放在 context 中的聚类中心转换为数组的形式，方便使用.

map 函数的输入为键值对 < LongWritable key，Text value>，其中 key 是当前样本相对于输入数据文件起始点的偏移量，value 是当前样本的各

维坐标值组成的字符串. 首先从 value 中取出当前样本各维的值；然后调用函数 getNearest()找出距离当前样本最近的聚簇的下标 id；最后将 id 作为 key，新的聚簇 cluster 的信息作为 value 写入 context 中. Map 函数的伪如下：

```
map(LongWritable key, Text value, Context context){
//从 value 中取出样本对象，写入 instance 对象；
Instance instance = new Instance(value.toString());
int id;
//找出和当前样本最近的聚簇的标号；
id = getNearest(instance);
Cluster cluster = new Cluster(id, instance);
cluster.setNumOfPoints(1);
context.write(new IntWritable(id), cluster); }
```

(4)KMeansCombiner

类中 Combine 函数将每个 Map 函数处理完的数据进行本地合并，减少了算法的通讯时间和传输数据量. combiner 函数的输入为＜IntWritable key，Iterable＜Cluster＞value＞键值对，其中 key 是聚簇的下标，value 是属于该簇的样本集合. 首先从 value 中取出每个样本的各维坐标值，并将每维对应的坐标值分别相加，同时记录其中样本的总数. 将聚簇的下标作为 key，更新信息后的 cluster 作为 value 写入 context. 函数的伪代码如下：

```
combiner(IntWritablekey, Iterable＜Cluster＞value, Context context){
//初始化变量 num，初始值为 0，记录分配给相同簇的样本数量；
int num= 0;
for(Cluster cluster:value){
num++;
初始化一个数组，每个分量初始值为 0，存储各维坐标累加值；
将各维坐标的值累加到数组对应分量；}
Cluster cluster = new Cluster();
将数组信息和 num 写入新的 cluster；
context.write(key, cluster);  }
```

(5) KMeansReducer

该类继承于 Reducer 类，输入格式为＜IntWritable, Cluster＞，输出格式为＜NullWritable, Cluster＞. 汇总所有簇的局部结果，计算所有簇的新中心.

Reduce 函数输入＜IntWritable key, Iterable＜Cluster＞value＞中，key 是聚簇的下标，value 是从各个 Combine 函数传输来的的中间结果. 在 Reduce 函数中首先取出每个 Combine 函数中的样本个数和相应节点各维坐标累加值；然后分别将各维累加值对应相加，再除以总的样本个数，得到新的中心点坐标. 将得到的聚簇 cluster 作为 value 写入 context. 函数伪码如下：

```
reduce(IntWritable key, Iterable＜Cluster＞value, Context context){
初始化一个数组，每个分量初始值为 0，存储各维坐标累加值；
初始化变量 NUM，初始值为 0，存储分配给同一聚簇的总样本数；
for(Cluster cluster : value){
从 value 中取出一个样本的样本数 num 和各维坐标值；
将各维坐标值累加存储到数组相应分量；
NUM+＝num; }
将数组各个分量除以 NUM，得到新的中心点坐标；
将数组信息写入新的 cluster；
context.write(NullWritable.get(), cluster); }
```

经过 Reduce 函数处理后，将得到的新的聚簇存储到 HDFS 文件中，供下一次迭代使用. 然后不停地迭代直到算法收敛.

4　实验

4.1　实验数据

实验数据分为两个部分，第一部分是来自 UCI 的鸢尾花数据集(Iris)，其中包含 150 个样本. 在这 150 个样本中，包含三种鸢尾花，分别是维吉尼亚鸢尾、变色鸢尾和山鸢尾. 每一个样本具有四个属性，分别是花瓣和花蕊的长度和宽度. 因为每种花都有 50 个样本，所以实验采用这个数据集，方便评价算法的准确性. 邻域半径 R 取为 1，阈值 M 取为 40.

第二部分数据来自于 Crowdflow 网. Crowdflow 网收集了从 2011 年四月以来的手机定位数据，本文采用德国整个国家的手机定位数据(crowdflow.net-CellLocation-07-June-2011)进行实验. 实验数据总共有 13 082 242 行，数据大小为 1.01 GB. 数据集中每条数据都包含了很多属性，我们取数据的三个属性：纬度，经度，水平精度，数据的数据结构为(Latitude, Longitude, HorizontalAccuracy). 因为这个数据集是海量数据集，实验用它来测试算法的性能，主要在运行时间，加速比和扩展性上对算法进行评价. 邻域半径 R 和阈值 M 的取值视数据量大小而定.

4.2　实验环境

实验用 4 台相同的电脑搭建了 Hadoop 分布式环境. 系统和软件环境都相同，见表 1，电脑硬件配

置见表 2.

表 1 实验系统环境

操作系统	实验平台	JDK 环境	执行环境
Ubuntu Kylin 14.04	hadoop－1.1.2（完全分布式模式）	jdk1.7.0_79	eclipse

表 2 硬件环境

PC 名称	PC1(master)	PC2(slave1)	PC3(slave2)	PC4(slave3)
内存/GB	2	2	2	2
CPU	双核 Pentium® Dual-Core E6700 3.20 GHz	双核 Pentium® Dual-Core E6700 3.20 GHz	双核 Pentium® Dual-Core E6700 3.20 GHz	双核 Pentium® Dual-Core E6700 3.20 GHz

4.3 实验结果及分析

(1)Iris 数据集聚类结果

首先从 Iris 数据集随机选取三个聚类中心，然后进行 K-means 聚类和改进 K-means 聚类. 表 3 给出了原始算法和改进算法对 Iris 数据集的聚类结果. 可以看出 K-means 算法成功分到相应簇的样本数为 132，准确率为 88%，改进 K-means 算法成功数为 143，准确率为 95.33%. 准确率提高了 7.33%. 这是因为改进算法使初始聚类中心的选取更加准确，从而提高了聚类效果. 但是由于初始聚类中心的选取要花费一定的时间，所以改进算法的迭代次数要多一些.

表 3 K-means 算法和改进 K-means 算法结果对比

聚类算法	聚类中心	聚类个数	准确率/%	迭代次数
K-means	(6.84999999, 3.07368421, 5.74210526, 2.07105263)	32	88	7
	(5.00599999, 3.41799999, 1.46400000, 0.24399999)	50		
	(5.90161290, 2.74838709, 4.39354838, 1.43387096)	68		
改进 K-means	(6.80232558, 3.04418604, 5.64883720, 2.03023255)	43	95.33	9
	(5.85438596, 2.74210526, 4.34561403, 1.40877192)	57		
	(5.00599999, 3.41800000, 1.46400000, 0.24400000)	50		

(2)手机定位数据聚类结果

该实验总共得到 63 个聚类中心. 通过观察每个聚类中心的数值可以发现这些聚类中心和德国的主要城市位置很接近. 也就是说，以主要城市为中心辐射的周围区域构成了城市带，这些区域是手机使用者的密集点. 而且按照每个类中包含点的数目从大到小排序，会发现这些主要城市的排名与德国城市发展水平排名相近. 这个结果反映了一个城市越发达，以这个城市为中心的周围区域手机使用者越多的现象.

4.4 运行时间

为了比较单机环境下的算法执行效率和集群环境下算法执行效率，对手机定位数据进行采样，测试不同大小数据集在不同集群节点数下算法的执行时间(时间单位为 ms). 因为原始 K-means 算法随机选取初始中心，所以进行多次实验取平均值. 实验结果如表 4 所示.

表 4 算法运行时间对比 ms

数据条	原始算法单机	改进算法单机	原始算法集群	改进算法集群
1×10^3	145	233	3 156	2 955
1×10^4	5 922	5 647	4 568	3 852
1×10^5	23 665	15 411	8 513	4 403
1×10^6	928 890	102 720	18 665	28 854
1×10^7	2 004 790	1 556 155	91 468	60 355
1×10^8	3 481 460	2 845 375	153 568	83 784

单机环境下，数据量比较小时改进算法的运行时间较原始算法并没有得到较大减少，数据为 1×10^3 条时时间反而增加了. 这是因为改进算法在选择初始聚类中心时还要遍历整个数据集，所以会多消耗一些时间. 但是这比随机选点更具有针对性，会使算法迭代次数减少，因而能够更快收敛；并行环境下，数据量较小时，算法运行时间也没有得到较大程度的减少，这是因为集群环境下各节点的通信消耗了一定的时间. 但是数据量较大时，算法运行时间得到了很大程度上的降低，而且数据量指数级别越大，时间减少地越来越快. 既体现了集群的高性能，又体现了初始聚类中心选取算法的高效性.

4.5 加速比

加速比反映了并行环境下算法的性能，加速比 $S=T_s/T_p$，T_s 为单节点算法运行时间，T_p 为 p 个相同节点算法运行时间($p=1, 2, 3, 4$). 实验取 1×10^5 条数据和其 1/10, 2/10, 4/10, 8/10 的数据测试加速比. 实验结果如图 2 所示. 从图中可以看出在 Hadoop 平台下改进 K-means 算法具有很好的加速比，而且数据量越大，其加速比越来越接近线性型增长. 但是因为节点的增多会使节点间通信消耗增大，所以随着节点个数的增加，加速比增长会变得平缓.

4.6 扩展性

扩展性反映了集群的性能，扩展性由扩展率衡量. 扩展率 $E=S/P$，S 为加速比，P 为节点数. 实验结果如图 3 所示，从图中可以看出：在数据量一样的情况下，随着集群节点数的增加，扩展率逐渐降低，

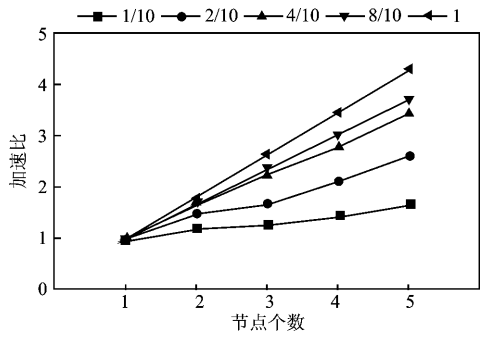

图 2　加速比实验结果

这是因为节点数增加会导致节点之间的通信消耗增加；当数据量增大时，随着集群节点的增加，扩展率的变化趋于平缓，这是因为每个节点会处理更多的数据，计算性能得到提升；而数据量较小时扩展率降低速度非常快，比如数据量为 1/10 时，节点数从 1 增加到 2 时，扩展率急剧降低，扩展性较差. 所以改进的 K-means 算法在处理海量数据时具有良好的扩展性.

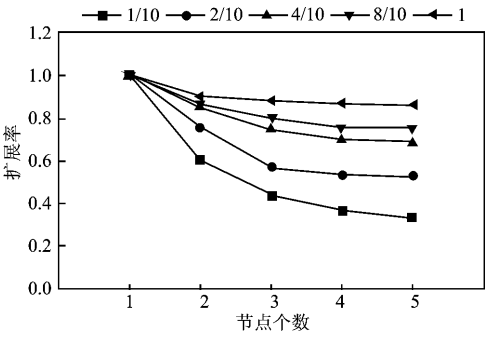

图 3　集群扩展性对比图

5　结束语

本文针对传统的 K-means 聚类算法的两个缺点，提出了基于采样和密度的改进算法，解决了算法随机指定 k 值和初始中心点的缺陷. 并且在 Hadoop 平台下实施该算法，实验结果表明改进算法的聚类准确度得到一定的提高，并且聚类性能有较大地提高.

虽然改进算法的聚类效果得到了提高，但是我们还需对初始中心和 k 值的选取过程作进一步的优化，比如本文中对象邻域半径和高密度邻域阈值 M 的选取都有待改进. 也可以对海量数据集中的噪声点进行处理，从而避免噪声点对初始聚类中心选取的影响.

参考文献：

[1] EMC Corporation. Extracting Value from Chaos [EB/OL]. [2015-08-15]. http://www.em-c.com/collateral/analyst-reports/.

[2] 李国杰，程学旗. 大数据的研究现状与科学思考[J]. 中国科学院院刊，2012，27(6)：647-657.

[3] Li Haiguang, Wu Gongqing, Hu Xuegang, et al. K-means clustering with bagging and mapReduce[C]// Proceedings of the 44th Hawaii International Conference on System Sciences. Hawaii, USA, 2011：1-8.

[4] Apache Hadoop. Hadoop[EB/OL]. [2015-08-18]. 00000http://hadoop.apache.org.

[5] 于海涛，李梓，姚念民. K-means 聚类算法优化方法的研究[J]. 小型微型计算机系统，2012，33(10)：2273-2277.

[6] Renato Cordeiro de Amorim, Boris Mirkin. Minkowski Metric. Feature weighting and anomalous cluster initializing in kmeans clustering[J]. Patt ern Recognition, 2012, 45(3)：1061-1075.

[7] Parimala M, Daphne Lopez, Senthilkumar N C. A survey on density based clustering algorithms for mining large spatial databases[J]. International Journal of Advanced Science and Technology, 2011, 31(1)：59-66.

[8] Murat Erisoglu, Nazif Calis, Sadullah Sakallioglu. A new algorithm for initial cluster centers in k-means algorithm[J]. Pattern Recognition Letter, 2011, 14(32)：1701-1705.

作者简介：

李　欢　男，(1992-)，硕士研究生. 研究方向为大数据与云计算. E-mail：lh1219508753@qq.com.

刘　锋　男，(1962-)，博士，教授，硕士生导师. 研究方向为并行计算与云计算.

朱二周　男，(1981-)，博士，副教授，硕士生导师. 研究方向为虚拟化与程序分析.