计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(9):242−246 [doi: 10.15888/j.cnki.csa.008019]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Hadoop 的改进型遗传聚类算法①

潘俊辉, 王　辉, 张　强, 王浩畅

(东北石油大学 计算机与信息技术学院, 大庆 163318)
通讯作者: 潘俊辉, E-mail: pjhdqpi@126.com

摘　要: 针对经典 K-means 聚类算法存在易陷入局部最优解的缺点, 提出并实现了一种基于 Hadoop 的改进型遗传聚类算法. 该算法利用遗传算法具有全局性和并行性的特点去处理 K-means 聚类算法易陷入局部最优的缺点, 在此基础上对遗传算法进行改进, 然后将改进后的遗传算法与 K-means 算法相结合, 为提高算法执行效率, 将其基于 Hadoop 平台进行了实现. 通过实验将该改进方法与经典聚类算法进行对比分析, 实验结果表明该方法在聚类准确性和聚类效率上均有较大的提高.

关键词: K-means; 文本聚类; 遗传算法; Hadoop; 并行性

引用格式: 潘俊辉,王辉,张强,王浩畅.基于 Hadoop 的改进型遗传聚类算法.计算机系统应用,2021,30(9):242–246. http://www.c-s-a.org.cn/1003-3254/8019.html

Improved Genetic Clustering Algorithm Based on Hadoop

PAN Jun-Hui, WANG Hui, ZHANG Qiang, WANG Hao-Chang

(School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China)

Abstract: Concerning the shortcoming that the classical K-means clustering algorithm is easy to fall into the local optimum, an improved genetic clustering algorithm based on Hadoop is proposed and implemented. The algorithm overcomes the above shortcoming with the globality and parallelism of the genetic algorithm. On this basis, the genetic algorithm is improved and then combined with the classical K-means algorithm. To improve the implementation efficiency, we implement the improved genetic clustering algorithm on Hadoop. The proposed method is compared with the classical clustering algorithm through experiments. The results show that the proposed method can greatly improve the clustering accuracy and efficiency.

Key words: K-means; text clustering; genetic algorithm; Hadoop; parallelism

目前互联网上产生了海量的数据, 而单机串行处理海量数据时存在瓶颈, 因此使用云计算、大数据对海量数据进行挖掘的技术不断涌现. 文本聚类可将相似的文本进行归类处理, 提取出文本中的关键信息, 该技术目前被广泛的应用在众多领域[1,2]. 针对目前互联网上的海量数据, 采用 Hadoop 对海量数据并行化实现文本聚类可有效的解决单机处理数据效率低的缺点.

目前, 国内外的一些学者对文本聚类的算法进行了研究. 国内的贾瑞玉等[3,4] 基于 MapReduce 模型并行化实现了遗传 K-means 聚类算法; 赵卫中等[5] 通过对云计算平台 Hadoop 进行研究, 将 K-means 聚类算法与遗传算法相结合并基于 Hadoop 平台进行了并行实现; 而国外的 Er 和 Erdoğan[6] 通过改进遗传算法中的染色体编码和适应度函数来选取聚类算法中的最优 K 值;

① 基金项目: 国家自然科学基金 (61702093); 东北石油大学青年科学基金 (2020QNL-02)
Foundation item: National Natural Science Foundation of China (61702093); Young Scientists Fund of Northeast Petroleum University (2020QNL-02)
收稿时间: 2020-11-17; 修改时间: 2020-12-21; 采用时间: 2020-12-25; csa 在线出版时间: 2021-09-02

文献 [7] 中的 Jiang 为减少算法的计算量, 以遗传算法中染色体的基因编码值作为 K-means 聚类算法的簇中心.

本文针对 K-means 聚类算法存在易陷入局部最优解等缺点, 而遗传算法所具备的特点正好可解决其缺点, 由此提出并实现了一种基于 Hadoop 的改进型遗传聚类算法.

1 Hadoop 平台及文本聚类

1.1 Hadoop 平台

Hadoop 平台是由 Apache Software Foundation 开发的开源项目[8,9]. 该平台的核心组件是 MapReduce 和 HDFS.

HDFS 以分布式的形式将划分的数据块存储到多节点中. 其中名字节点的工作用于管理名字空间和处理有关客户端对数据的访问等操作; 数据节点的工作则用以完成所在节点的存储、根据名字节点的要求实现对数据块的创建和删除等操作[10,11].

MapReduce 的核心是 map 和 reduce 函数, 通过它们可实现对海量数据的并行化编程. MapReduce 中用户输入的键值对在 map 的映射下会得到新的键值对作为中间输出结果, 而该中间结果经过 reduce 函数的规约处理后, 可得到最终结果[12,13].

1.2 文本聚类

文本聚类是指将无序的对象作为输入, 使用相似性度量函数计算未知类别文本的相似性, 输出则为相似性文本组成的簇集. 其聚类过程采用数学形式表示为: 设文本数据集: $D=\{d_1, d_2, \cdots, d_i, \cdots, d_n\}$, d_i 为数据集中的文本, n 为总文本数, 聚类之后得到的簇集为: $E=\{e_1, e_2, \cdots, e_j, \cdots, e_k\}$, e_j 为相似文本聚成的簇, k 为聚类个数. 当满足条件$\forall d_i(d_i \in D), \exists e_j(e_j \in E), d_i \in e_j$, 将语料库 D 分为 k 个簇, 使其满足如下公式: $\cup_{i=1}^{k} e_i = D$, 且 $e_i \cap e_j = \varphi$.

2 基于 Hadoop 的改进型遗传聚类算法

2.1 遗传算法

遗传算法由学者 Holland 提出. 该算法包括 5 个不可缺少的成分: 种群规模、编码、种群初始化、适应度函数和遗传操作.

2.1.1 编码

常见的编码方法有二进制编码、浮点数编码和符号编码, 因二进制编码在编码和解码上操作简单, 易于实现, 故最为常用.

2.1.2 种群初始化

编码方式确定后, 可对种群进行初始化, 通常可采用两种方法, 一种是参数范围内通过重复随机选择个体的方式达到种群所需的规模; 另一种则在局部最优解区间内, 根据经验选出初始种群.

2.1.3 适应度函数

适应度函数的选择通常和目标函数相关. 当目标函数分别为求最小值和最大值问题时, 适应度函数分别如式 (1) 和式 (2) 所示.

$$Fit(f(x)) = \begin{cases} F_{\max} - f(x), & f(x) < F_{\max} \\ 0, & f(x) \geq F_{\max} \end{cases} \tag{1}$$

$$Fit(f(x)) = \begin{cases} f(x) - F_{\min}, & f(x) > F_{\min} \\ 0, & f(x) \leq F_{\min} \end{cases} \tag{2}$$

式中, $F_{\min}$为最小值, $F_{\max}$为最大值, $f(x)$均为适应度值[14].

2.1.4 遗传操作

选择算子通过选择策略将种群中适应度值较大的个体选择出来, 将较优的个体传给下一代.

交叉算子决定着遗传算法的全局寻优能力, 是算法的核心. 在进行交叉操作时可通过单点交叉、两点交叉和多点交叉等方式进行.

在种群规模较大时, 可通过变异操作提高遗传算法的局部搜索能力. 一般变异概率取值均小于 0.1. 通常在二进制编码中, 变异操作就是互换 0 和 1 的操作.

2.2 改进型遗传算法

针对遗传算法存在的不足和增强遗传算法的全局搜索能力, 从而保证种群的多样性, 提出了一种新的改进型遗传算法 (Improved Genetic Agrithom, IGA), 该算法分别在交叉算子、变异算子和自适应遗传算子上做了相应的改进.

2.2.1 交叉算子、变异算子和遗传算子的改进

遗传算法在执行时不仅要提高收敛速度, 也应在迭代过程中提高种群的多样性以增强其全局寻优能力, 因此本文对交叉算子进行了改进, 分别通过进行族内交叉和族间交叉来实现.

变异算子并没有直接选取所有变异后的个体, 而是通过比较变异前后的个体已防止优秀的个体被破坏.

为防止算法陷入局部最优, 本文采用的自适应算子如式 (3) 和式 (4) 所示.

$$P_c=\begin{cases}\dfrac{(p_{c\max}-p_{c\min})}{1+\exp\left(A\left(\dfrac{2(f-f')}{f_{\max}-f_{\text{avg}}}\right)\right)}+p_{c\min}, & f\geq f_{\text{avg}}\\ P_{c\max}, & f<f_{\text{avg}}\end{cases} \tag{3}$$

$$P_m=\begin{cases}\dfrac{(p_{m\max}-p_{m\min})}{1+\exp\left(A\left(\dfrac{2(f''-f')}{f_{\max}-f_{\text{avg}}}\right)\right)}+p_{m\min}, & f''\geq f_{\text{avg}}\\ P_{m\max}, & f''<f_{\text{avg}}\end{cases} \tag{4}$$

式中, $f'=\frac{1}{2}(f_{\max}+f_{\text{avg}})$; f''为要变异个体的适应度值; $p_{c\max}$, $p_{c\min}$, $p_{m\max}$, $p_{m\min}$均在 (0,1) 之间.

2.2.2 IGA 算法实现的具体步骤

Step 1. 给出种群规模 N, 交叉概率 P_c, 变异概率 P_m 和最大迭代次数 $G_{\max}$;

Step 2. 生成初始种群, 对初始种群进行二进制编码;

Step 3. 采用最小生成树 Prim 算法对种群规模为 N 的种群进行分类, 得到的最小生成树的每个子连通图作为子类, 对子类编号得到 C_i (i=1, 2, ⋯ , k);

Step 4. 对个体 P_i (i=1, 2, ⋯, N) 的选择采用轮盘赌选择法;

Step 5. 执行交叉操作, 对个体 P_i (i=1, 2, ⋯ , N) 的交叉过程为:

(1) 在 P_i 所属的类 C_{ic}($C_{ic}\in\{C_1, C_2, \cdots, C_k\}$) 中进行查找, 随机选择一个类 C_{ic1}($C_{ic1}\in\{C_1, C_2, \cdots, C_k\}$且 $C_{ic}\neq C_{ic1}$) 并从中选择出适应度值最大对应的个体 P_j, 对 P_i 和 P_j 进行单点交叉, 得出子个体表示为 T_1 和 T_2;

(2) 在 P_i 所属的类 C_{ic}($C_{ic}\in\{C_1, C_2, \cdots, C_k\}$) 中寻找据 C_{ic} 最远的类 C_{ic2} 并随机获得一个最优的个体 P_k, 对 P_i 和 P_k 进行单点交叉, 得出子个体表示为 T_3 和 T_4;

(3) 在 P_i 所属的类 C_{ic}($C_{ic}\in\{C_1, C_2, \cdots, C_k\}$) 内选择最优个体 P_n, 对 P_i 和 P_n 进行单点交叉, 得出子个体表示为 T_5 和 T_6;

(4) 对 T_i, $i\in$(1, 2, 3, 4, 5, 6) 进行比较, 从中得到最优子个体 T_i 并将 T_i 赋值给 P_i.

Step 6. 对 P_i 执行变异操作, 将得到的新个体 P_j 的适应度值与 P_i 的进行比较, 最优的赋值给 P_i.

Step 7. 比较 P_i 的数目与种群规模, 如小于转到 Step 4, 如相等则继续;

Step 8. 将迭代次数与设定的最大迭代次数 $G_{\max}$ 进行比较, 如相等, 算法结束, 否则转向 Step 3.

2.3 改进型遗传聚类算法

将上述改进的遗传算法与 K-means 聚类算法相结合得到改进型的遗传聚类算法 IGA-K, 在 IGA-K 算法中将 K-means 算法的聚类中心和聚类中心的个数分别作为遗传算法的染色体的基因和染色体的长度, 并对其适应度函数和编码方式进行了改进.

2.3.1 适应度函数的设置

为得到合适的分类结果, 将 IGA-K 算法的适应度函数设置为式 (5) 所示:

$$f=\frac{D_{\min}}{C(x)}=\frac{\min\limits_{i,j=1}^{k}\left\|c_i-c_j\right\|^2}{\dfrac{1}{k}\left(\sum\limits_{i=1}^{k}\left(\sum\limits_{j=1}^{n_j}\left\|x_j-c_j\right\|^2/n_i\right)\right)} \tag{5}$$

为使类内更加紧凑, 式中$D_{\min}$的值尽可能取大, 而$C(x)$的值尽可能取小.

2.3.2 编码方式的选择

改进的遗传聚类算法中编码方式采用混合型编码, 具体为:

$$chrom=\begin{Bmatrix} binstr1 & x1 & k(x1) & f(x1)\\ binstr2 & x2 & k(x2) & f(x2)\\ \vdots & \vdots & \vdots & \vdots\\ binstrn & xn & k(xn) & f(xn)\end{Bmatrix}$$

2.3.3 IGA-K 算法的基本步骤

改进型遗传聚类算法 (IGA-K) 的具体实现过程如下:

Step 1. 给出遗传操作的基本参数;

Step 2. 对初始种群采用改进的编码形式实现编码;

Step 3. 采用 K-means 聚类算法的初始聚类中心作为种群中的每个个体并实现对样本的分类, 依据结果计算得到适应度值.

Step 4. 按照 IGA 算法进行个体的分类和执行相应的遗传操作;

Step 5. 算法如满足结束条件则结束, 否则转到 Step 3 接着执行.

2.4 基于 Hadoop 的改进型遗传聚类算法

由于 K-means 算法和遗传算法二者均具有并行性的特点, 由此本文将改进型的遗传聚类算法迁移到 Hadoop 平台上进行并行化的实现并将其命名为 H-IGA-K 算法. 为并行化实现该算法必须实现两个函数, 其中一个关键函数为遗传算法中用于求个体适应度的内部

子函数 (K-inner), 另一个函数则是 H-IGA-K 算法的主函数.

2.4.1 K-inner 内部子函数的实现

K-inner 中包含 3 个函数, 分别为 Map、Combine 和 Reduce, 其中 Map 函数的功能为计算数据集中各个样本所属的类别, 输入输出均为键值对. Map 的输入键值对中的 *key* 表示当前样本相对于数据文件起始点的偏移量; *value* 则为样本属性值生成的字符串; 输出键值对中的 *key'*表示所属类的序号, *value'*和 *value* 一样, 意义没发生变化.

Combine 函数为计算本地节点中各类的样本数及其属性值组成的字符串. 该函数的输入键值对中的 *key* 表示样本所属类的序号, *value* 为样本属性值生成的字符串; 输出键值对中的 *key'*表示样本所属类的序号, *value'*则为样本总数和样本属性值组成的字符串.

在 K-inner 中, Reduce 函数的功能尤为重要, 因此这里主要说明 Reduce 实现的伪代码.

```
Reduce(<key, value>,<key', value'>)
{   Initialize an Array A; //初始化数组 A
    A.Length=Count of Class; //类的个数为数组长度
    for i=0 to A.Length-1 //初始分量值为 0
      A[i]=0; //A[i] 为各个类内距离
    While (value not null)
    { 分解得到 each sample 的各维坐标值;
      Get sample number; //得到样本个数
      Compute distance between each sample and the center of corresponding class;
      Add distance; //计算得到的距离值相加
      value=value.next()
    }
    Compute minimum distance base center;
    //计算最小类间距
    Compute fitness of population individual;
    //计算个体适应度的值
    key'=value of individual fitness
    value'=string of individual chromosome;
    // value'的值为个体染色体字符串
    output <key', value'>;
}
```

2.4.2 H-IGA-K 算法主函数的实现

H-IGA-K 算法主函数实现的过程中的关键点同样是 Reduce 函数, 下面是 Reduce 实现过程的伪代码.

```
Reduce(<key, value>,<key', value'>)
{
  for i=1 to n
    {K-inner Function; }
  调用 Prim 算法对种群进行分类;
  if (满足问题最优解||达到最大遗传代数)
    { output(key, maxValue); }
  else
    { 按照改进交叉方法执行交叉操作,
    优选得到下代个体;
    按照改进变异方法执行变异操作,
    得到新个体;
    }
  output(key', value');
}
```

3 实验结果

本文对基于 Hadoop 的改进型遗传聚类算法进行了实验. 实验采用 10 台相同的计算机搭建了实验平台; 该平台使用的软件为 Hadoop-2.9.1.tar.gz、CentOS-7-x86_64-NetInstall-1804.iso 和 jdk-8u181-linux-x6. 实验用数据集为从 UCI 数据库的 Iris、Glass 及 Adult 三个数据集中选择出的数据, 并经人工构造而得, 构造的 3 个数据集 Dataset1、Dataset2 和 Dataset3 的规模分别为 100、1000 和 5000, 3 个数据集对应的分类数分别为 3、5 和 10. H-IGA-K 算法参数分别为: G_{max}=300、P_c=0.92、P_m=0.02、自适应算子中的 p_{cmin}、p_{mmin}、p_{cmax} 和 p_{mmax} 的取值分别为 0.1, 0.05, 0.9 和 0.1. 实验分别从聚类准确性和聚类效率两个方面进行了测试.

3.1 聚类准确性实验

为测试基于 Hadoop 改进型遗传聚类算法的聚类准确性, 本实验使用基于 Hadoop 的 K-means 算法和 H-IGA-K 算法对实验用数据集中的数据分别进行了处理, 表 1 给出了二者的聚类准确率情况.

由表 1 可见, 随着数据规模的增加, 两种算法的聚类准确率均在下降, 但下降的程度有所差距, 并且在任何一个数据集下 H-IGA-K 算法在聚类准确率上均比基于 Hadoop 的传统 K-means 聚类算法要高, 特别是 Dataset3 数据集, 由此说明基于 Hadoop 的改进型遗传

聚类算法优于基于 Hadoop 的 K-means 聚类算法, 而且适用于处理大规模的数据集.

表 1 聚类准确率比较

数据集	算法名称	聚类准确率(%)
Dataset1	K-means算法	82.31
	H-IGA-K算法	92.89
Dataset2	K-means算法	70.56
	H-IGA-K算法	84.67
Dataset3	K-means算法	60.25
	H-IGA-K算法	79.16

3.2 聚类效率实验

为了测试基于 Hadoop 改进型遗传聚类算法的聚类效率, 本实验通过对 3 个测试数据集 Dataset1、Dataset2 和 Dataset3 分别在 1、5、10 个节点构建的集群上进行了基于 Hadoop 的改进型遗传聚类算法的实验, 表 2 给出了数据集 Dataset1、Dataset2 和 Dataset3 在不同节点个数上进行文本聚类时运行所需的时间比较.

表 2 3 个测试数据集在不同节点进行文本聚类的运行时间 (s)

数据集	节点数1	节点数5	节点数10
Dataset1	1223.65	1046.83	586.88
Dataset2	2373.26	1937.69	938.48
Dataset3	4317.94	3024.18	1453.37

由表 2 可见, 对 Dataset1、Dataset2和 Dataset3 进行操作时, Hadoop 集群节点个数越多, 程序运行时所需的总时间均在减少, 节点个数增加越多, 程序运行所需时间越少, 并且随着数据集规模增加, 算法运行时消耗的时间减少的也越多, 由此表明基于 Hadoop 的多节点的集群更适合处理大数据集, 特别在节点个数达到 10 个时, 算法运行效率大大提高.

4 结论

本文提出了一种改进型的遗传聚类算法, 并针对单机串行对海量数据聚类效率低的问题, 将改进型的遗传聚类算法基于 Hadoop 平台进行了实现. 实验表明改进型的算法优于经典聚类算法, 在聚类准确性和聚类效率上均有较大的提高, 适用于处理大规模的数据集. 本文主要针对传统 K-means 存在易陷入局部最优解的缺点进行的研究, 而传统的 K-means 算法存在诸多不足, 因此在接下来的研究中, 还可尝试解决 K-means 算法的其他不足点.

参考文献

1 代劲. 云模型在文本挖掘应用中的关键问题研究 [博士学位论文]. 重庆: 重庆大学, 2011.

2 Han JW, Pei J, Kamber M. Data Mining: Concepts and Techniques. 3th ed. San Francisco: Morgan Kaufmann Publishers, 2001. 37–43.

3 贾瑞玉, 管玉勇, 李亚龙. 基于 MapReduce 模型的并行遗传 K-means 聚类算法. 计算机工程与设计, 2014, 35(2): 657–660. [doi: 10.16208/j.issn1000-7024.2014.02.053]

4 杨新武, 杨丽军. 基于交叉模型的改进遗传算法. 控制与决策, 2016, 31(10): 1837–1844. [doi: 10.13195/j.kzyjc.2015.1082]

5 赵卫中, 马慧芳, 傅燕翔, 等. 基于云计算平台 Hadoop 的并行 K-means 聚类算法设计研究. 计算机科学, 2011, 38(10): 166–168, 176. [doi: 10.3969/j.issn.1002-137X.2011.10.037]

6 Er HR, Erdoğan N. Parallel genetic algorithm to solve traveling salesman problem on MapReduce framework using hadoop cluster. The International Journal of Soft Computing and Software Engineering, 2013, 3(3): 380–386.

7 Jiang D, Tung AKH, Chen G. MAP-JOIN-REDUCE: Toward scalable and efficient data analysis on large clusters. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(9): 1299–1311. [doi: 10.1109/TKDE.2010.248]

8 林子雨. 大数据技术原理与应用: 概念、存储、处理、分析与应用. 北京: 人民邮电出版社, 2017.

9 陈文廉. HADOOP 平台与 MAP-REDUCE 编程模型. 信息记录材料, 2019, 20(12): 86–88. [doi: 10.16009/j.cnki.cn13-1295/tq.2019.12.052]

10 White T. Hadoop: The Definitive Guide. 4th ed. Sebastopol: O'Reilly Media, 2015.

11 刘鹏. 云计算. 北京: 电子工业出版社, 2010. 216–219.

12 陆嘉恒. Hadoop 实战. 2 版. 北京: 机械工业出版社, 2012. 56–62.

13 李锐, 王斌. 文本处理中的 MapReduce 技术. 中文信息学报, 2012, 26(4): 9–20. [doi: 10.3969/j.issn.1003-0077.2012.04.002]

14 周理. 基于 Hadoop 的数据挖掘算法的研究与应用 [硕士学位论文]. 北京: 华北电力大学, 2019.