软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(5):1079−1095 [doi: 10.13328/j.cnki.jos.004659]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于 MapReduce 与相关子空间的局部离群数据挖掘算法*

张继福[1], 李永红[1], 秦 啸[2], 荀亚玲[1]

[1](太原科技大学 计算机科学与技术学院,山西 太原 030024)

[2](Department of Computer Science and Software Engineering, Auburn University, Auburn, USA)

通讯作者: 张继福, E-mail: jifuzh@sina.com, http://www.tyust.edu.cn/

摘 要: 针对高维海量数据,在 MapReduce 编程模型下,提出了一种基于相关子空间的局部离群数据挖掘算法.该算法首先利用属性维上的局部稀疏程度,重新定义了相关子空间,从而能够有效地刻画各种局部数据集上的分布特征;其次,利用局部数据集的概率密度,给出了相关子空间中的局部离群因子计算公式,有效地体现了相关子空间中数据对象不服从局部数据集分布特征的程度,并选取离群程度最大的 N 个数据对象定义为局部离群数据;在此基础上,采用 LSH 分布式策略,提出了一种 MapReduce 编程模型下的局部离群数据挖掘算法;最后,采用人工数据集和恒星光谱数据集,实验验证了该算法的有效性、可扩展性和可伸缩性.

关键词: 局部离群数据;相关子空间;MapReduce;局部稀疏度;概率密度

中图法分类号: TP311

中文引用格式: 张继福,李永红,秦啸,荀亚玲.基于 MapReduce 与相关子空间的局部离群数据挖掘算法.软件学报,2015,26(5):1079−1095. http://www.jos.org.cn/1000-9825/4659.htm

英文引用格式: Zhang JF, Li YH, Qin X, Xun YL. Related-Subspace-Based local outlier detection algorithm using MapReduce. Ruan Jian Xue Bao/Journal of Software, 2015,26(5):1079−1095 (in Chinese). http://www.jos.org.cn/1000-9825/4659.htm

Related-Subspace-Based Local Outlier Detection Algorithm Using MapReduce

ZHANG Ji-Fu[1], LI Yong-Hong[1], QIN Xiao[2], XUN Ya-Ling[1]

[1](School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China)

[2](Department of Computer Science and Software Engineering, Auburn University, Auburn, USA)

Abstract: In this paper, a related-subspace-based local outlier detection algorithm is proposed in MapReduce programming model for high-dimensional and massive data set. Firstly, the relevant subspace, which can effectively describe the local distribution of the various data sets, is redefined by using local sparseness of attribute dimensions. Secondly, a local outlier factor calculation formula in the relevant subspace is defined with probability density of local data sets. The formula can not only effectively reflect the outlierness of data object that does not obey the distribution of the local data set in relevant subspace, but also select N data objects with the greatest-outlierness as local outliers. Furthermore, a related-subspace-based local outlier detection algorithm is constructed by using LSH distributed strategy in MapReduce programming model. Finally, experimental results validate the effectiveness, scalability and extensibility of the presented algorithms by using artificial data and stellar spectral data as experimental data sets.

Key words: local outlier; relevant subspace; MapReduce; local sparsity; probability density

离群数据(outlier)就是明显偏离其他数据、不满足数据的一般模式或行为、与存在的其他数据不一致的数据[1],蕴含着大量的不易被人类发现却很有价值的信息.离群挖掘作为数据挖掘的一个重要分支,已广泛的应用在天文光谱数据分析[2]、信用卡诈骗[3]、网络入侵挖掘[4,5]、数据清洗[6]等领域.目前,离群数据挖掘大致分为基

* 基金项目: 国家自然科学基金(61272263)
收稿时间: 2013-12-19; 修改时间: 2014-02-17; 定稿时间: 2014-05-21; jos 在线出版时间: 2014-08-19
CNKI 网络优先出版: 2014-08-19 14:29, http://www.cnki.net/kcms/doi/10.13328/j.cnki.jos.004659.html

于统计[7]、基于距离[8,9]、基于密度[10,11]、基于偏离[12]和基于角度的[3,13]等方法,然而,这些方法都假设各维对度量离群数据作用是相同的[14],无关维的存在会增加“维灾”对挖掘效果的影响,可能会无法挖掘一些隐藏在子空间中的离群数据.

在高维海量数据中,由于数据量大和维度高,严重地影响了离群数据挖掘效果和效率,可能无法发现隐藏在子空间中的一些离群数据.在大多数情况下,离群数据是与局部数据集的分布特征明显不一致的数据对象.但在有些属性维上,可以提供不一致的有价值信息,而在其他属性维上,则无法提供有价值的信息,对度量离群数据几乎是无关的[15].因此,寻找和删除高维数据集中无法提供有价值信息的属性维,可以有效地降低“维灾”的干扰,并能有效地发现隐藏的离群数据.本文针对高维海量数据,充分利用集群系统和 MapReduce 编程模型的强大数据处理能力,提出一种基于相关子空间的局部离群数据挖掘算法.该算法首先从局部稀疏程度的角度,利用定义在局部数据集上的局部稀疏度因子,重新定义了相关子空间,从而有效地刻画各种局部数据集上的分布特征;其次,对相关子空间中的数据对象,利用局部数据集的概率密度和高斯误差函数给出了其局部离群因子的计算公式,从而有效地体现了相关子空间中数据对象与局部数据集分布特征不一致的程度;然后,选取离群程度最大的 N 个数据对象作为局部离群数据;在此基础上,采用分布 Hash 索引策略和近似 K 近邻方法,提出了一种 MapReduce 编程模型下的局部离群数据挖掘算法;最后,采用人工数据集和恒星光谱数据集,实验验证了该算法的有效性、可扩展性和可伸缩性.

1 相关工作与分析

由于传统的离群数据挖掘方法[7−13]大多数是在所有维构成的空间中挖掘离群数据,一些无关维的存在会对离群点的挖掘效率和精度带来影响.将数据投影到子空间上进行离群挖掘[15−17]是目前高维离群数据挖掘研究的热点之一,然而在寻找有意义的子空间时比较困难[16],具有维度指数级的复杂度[15].针对数据挖掘任务,在高维数据中选择有意义子空间的典型方法主要有稀疏度子空间法[2,18,19]和相关子空间法[14,20,21]两大类.

稀疏度子空间法是根据用户设定稀疏系数阈值,将数据投影到稀疏子空间中,包含在该子空间中数据对象定义为离群数据,因此是一类基于全局的子空间方法.典型工作有:Agarwal 等人[18]采用遗传算法搜索稀疏子空间,但该算法受初始种群影响,离群挖掘结果的完备性和准确性等无法得到保证;张继福等人[2,19]针对文献[18]中存在的不足,采用概念格作为子空间描述工具,通过引入稠密度系数,在概念格内涵中确定稀疏子空间,进一步提高了挖掘结果的准确性和完备性,并已应用在天文光谱数据中,取得了较好效果,但由于概念格构造的复杂性,挖掘效率较低.

相关子空间法[14,15,17,20,21]是在数据集中寻找由有意义的属性维构成的相关子空间,从而在相关子空间中度量离群数据,主要采用的方法是基于局部参考数据集的线性相关性[14,20,21]以及基于局部参考数据集的统计模型[15,17]等.典型工作有:Kriegel 等人[20]提出了轴平行子空间离群挖掘(SOD)方法,该方法通过共享最近邻居(SNN)为每一个数据对象寻找一个相似子集,并在相似子集中确定轴平行线性相关子空间,在 SOD 中,仅仅在一个相关子空间中来刻画数据对象的离群程度,当数据对象分布在两个或两上以上的子空间中时,SOD 将不能区分它们的离群程度[15],显然,在相关子空间中,采用欧式距离的维平均方法度量离群具有明显的不足;Muller 等人[15]利用柯尔莫哥洛夫-斯米尔诺夫检验的统计方法,提出了在数据集上选择有意义的子空间方法,从低维到高维采用递归的方式逐步寻找非均匀分布的子空间,并将数据对象在各个相关子空间中的局部离群值累乘,作为该数据对象最终的离群程度,解决了不同子空间中的离群因数可比性问题,但在该算法中,确定所有相关子空间具有维度指数级的时间复杂性[3],因此其效率较低,对高维数据集的维度可扩展性差,无法适应于海量高维数据;Keller 等人[17]利用蒙特卡洛方法寻找相关子空间集,该方法由数据对象对应在每个相关子空间中的局部数据子集确定该数据对象的离群程度,但该相关子空间实际上是从全局的角度考虑得到的;Kriegel 等人[14]以主成分方法为理论依据,采用马氏距离结合伽玛分布的方法实现了离群数据挖掘,该方法得到的相关子空间是线性相关的任意子空间,相关子空间对线性分布的数据具有很好的适应性,但由于该方法是基于对局部数据分布趋势的偏离,当离群数据所在稀疏区域体现出一种不明显的相关性时,由最大似然估计得到的相关子空间容易出现

错误,因此需要足够的局部数据来体现明显的趋势,同时,该算法的时间复杂度是维度的三次方,难以适用于高维数据;此外,Bouguessa 等人[21]利用稀疏度密度矩阵寻找相关子空间,并在该相关子空间中实现聚类分析.

针对离群挖掘任务,其并行与分布式算法尚不多见,一些相关的典型研究工作有:文献[22]将 LSH(locality sensitive hashing)策略应用在 MapReduce 框架中,并有效地解决了在高维海量数据集中近似 K 近邻查询问题;文献[23,24]在 MapReduce 框架中提出了 k-近邻连接算法,为大数据中的 K-近邻查询处理提供了一种较为有效的方法,具有较好的实用价值;文献[25]在 MPI 编程模型下提出了一种基于距离的局部离群数据挖掘算法,但由于该算法是在所有属性维够成的空间中度量离群数据,很难适应于高维数据离群挖掘任务.

2 基本概念

2.1 相关子空间与局部离群概率

针对不同的数据挖掘任务,数据集中的数据对象仅仅在特定属性维上能够体现出有价值的信息,而其他一些属性维可能很少甚至不会体现出有价值的信息[15].设 DS 是任意一个 d 维数据集,属性集 $FS=\{A_1,A_2,\ldots A_d\}$,$x_{ij}(i=1,2,\ldots,n;j=1,2,\ldots,d)$表示第 i 个数据对象 obj_i 在第 j 个属性上的取值.参照文献[15,26],相关子空间和局部离群概率的基本概念和定义描述如下:

对于任意子空间 $S\subseteq FS$ 和数据对象 $o\in DS$ 的最近邻 $N(o,S)$,如果 S 是相关子空间,则 $N(o,S)$是不均匀分布的;如果 S 是不相关子空间,则 $N(o,S)$是均匀分布的.

在文献[15]中,非均匀分布的子空间才能有效地体现出“离群数据”的有价值信息,均匀分布的属性维无法体现出“离群数据”的有价值信息.因此,相关子空间有效地反映了“离群数据”的有价值信息,度量和寻找相关子空间成为离群数据挖掘的关键.

在文献[26]中,对于任意数据对象 $o\in DS$,设 $S\subseteq DS$ 为关联数据集且 $o\in S$,则 o 在 S 中的概率距离 $pdist(o,S)$满足:$\forall s\in S:P[d(o,s)\leqslant pdist(o,S)]\geqslant\varphi$.

直观地说,$pdist(o,S)$是在以 o 为球心和以 $pdist$ 为半径的球中,包含在 S 中的数据对象个数的概率.$pdist(o,S)$可以间接地用于估计 S 的密度,也就是说,S 的密度可定义为

$$pdens(o,S)=\frac{1}{pdist(o,S)} \tag{1}$$

假设 o 处于 S 的中心,并且 $s\in S$ 到 o 的距离集合近似服从半高斯分布,则类似于标准差,计算 s 到 o 的标准距离如下:

$$\sigma(o,S)=\sqrt{\frac{\sum_{s\in S}d(o,s)^2}{|S|}} \tag{2}$$

在公式(2)中,由于使用距离 $d(o,o)=0$ 代替了平均值 $E[d(o,S)]$,o 的标准距离与标准差 $Stddev(d(o,S))$不同.因此,不能假设 $s\in S$ 到 o 的距离值集合是正态分布的,而可以假设 S 服从以 o 为中心的正态分布.可以利用该假设确定关联数据集 S,即,$S=o$ 与其 K 最近邻.o 到 S 的 λ相关的概率数据集合距离定义如下:

$$pdist(\lambda,o,S)=\lambda\cdot\sigma(o,S) \tag{3}$$

其中,参数λ用于控制密度的近似值.

在公式(3)中,$pdist(\lambda,o,S)$用于估计关联集 S 的密度.对于任意数据对象 $o\in DS$,设 $S\subseteq DS$ 为 o 的关联数据集,则λ相关的概率局部离群因子(PLOF)定义如下:

$$PLOF_{\lambda,S}(o)=\frac{pdist(\lambda,o,S(o))}{E_{s\in S(o)}[pdist(\lambda,s,S(s))]}-1 \tag{4}$$

其中,E 表示均值,$S(o)$表示 o 的关联数据集.

采用高斯误差函数,表示数据对象 $o\in DS$ 是离群数据的概率值(局部离群概率(LoOP))公式定义如下:

$$LoOP_S(o)=\max\left\{0,erf\left(\frac{PLOF_{\lambda,S}(o)}{\sqrt{2}\cdot nPLOF}\right)\right\} \tag{5}$$

其中,$nPLOF=\lambda\cdot E[(PLOF)^2]$.

2.2 LSH策略

由于标准的 KNN 算法使用的是全局搜索方法,不适合应用于 MapReduce 框架.文献[22]提出了一种 LSH 策略,并在 MapReduce 框架下处理大数据中的近似 KNN 问题,其关键思想是:使用一组哈希函数,使距离较近的两个数据对象以较高的概率分配到同一个哈希桶中;在查询每个数据对象 *obj* 的近似 KNN 时,通过 *obj* 对应的哈希值找到对应的哈希表中的对应哈希桶,并将哈希桶中的所有数据对象导入程序中,查询数据对象 *obj* 的近似 KNN.参照文献[22,27,28],哈希函数定义如下:

$$h_{a,B(v)}=\left\lfloor\frac{a\cdot v+B}{W}\right\rfloor \tag{6}$$

其中,v 为输入数据集中的任意一数据对象;a 是 d 维随机向量,a 中的每一维取值均是从 *p-Stable* 分布中独立地选择出来的;$W\in R$(实数集),B 是从$[0,W]$中随机选出的一个实数.

每一个哈希函数将一个 d 维数据点映射到一个整数集.如果有 k 个哈希函数,则最终是一个长度为 k 的向量函数 $g(v)$:

$$g(v)=(h_{a_1,B_1(v)},\ldots,h_{a_k,B_k(v)}) \tag{7}$$

$g(v)$对应一个哈希表,为了获得近似 KNN 的较高的准确率,有必要建立多个哈希表,并且采用每个哈希表对应一个探针的策略.

3 相关子空间、子空间定义向量与离群数据

3.1 相关子空间

由于真实数据集往往包含由多种不同机制产生的簇(cluster),不同机制可能作用于不同的属性维子集[14].属性维子集体现了相应的机制可以生成对应的数据子集,并体现一种相似的局部分布特征,即,对于由特定机制生成的数据子集,属性维子集具有局部相关性(局部稀疏分布特征共性,即,有些属性维是非均匀分布的,有些是均匀分布的).通常,由特定机制生成的数据子集达到一定的数量时,局部分布特征的共同特征才能变得明显.离群数据可以认为不是由该种机制产生的数据对象,即,离群数据是与局部相关性明显不一致的数据对象.对于任意数据对象 *obj*,与局部稀疏分布特征共性不一致的属性维体现了一类有价值信息,并可由局部数据属性维取值不均匀分布来刻画;与局部稀疏分布特征共性一致的属性维则不能体现有价值信息,并可由局部数据属性维取值均匀分布来刻画.参照文献[15],相关子空间重新定义如下:

定义 1. 针对离群数据挖掘,设 *DS* 是一个 d 维数据集,全空间 $FS=A$(属性集 $A=\{A_1,A_2,\ldots,A_d\}$,数据对象 *obj* 的最近邻为 $N(obj,FS)$(即,局部数据集 *LDS*),如果 $N(obj,FS)$在 A_i 属性维上的取值是均匀分布,则称 A_i 不能提供有价值的信息,属于不相关子空间中的属性维;反之,如果 $N(obj,FS)$在 A_i 属性维上的取值是非均匀分布的,则称 A_i 可以提供有价值的信息,属于相关子空间中的属性维.

在文献[15]中,利用最近邻 $N(o,S)$来确定每个子空间 S 是否为相关子空间,对于数据集 *DS*,随着 S 的维度调整,包含 S 中的数据集的分布特征也会改变.因此,当 S 不为 FS 时,$N(o,S)$会失去局部数据集的意义,不能有效地体现 *DS* 在 *FS* 中局部数据集的局部分布特征的共同特性.而在定义 1 中,仅利用 *FS* 中的 $N(obj,FS)$,确定相关子空间,有效地避免了文献[15]中的上述不足.

3.2 子空间定义向量

在局部数据集 *LDS* 中,均匀分布属性维的取值较均匀,稀疏程度(密度)相似,差异较小,因此可以采用局部数据属性维的稀疏差异程度来度量均匀分布属性.因此,通过删除均匀分布的属性维,保留非均匀分布属性维,可

有效地确定数据对象的相关子空间.

设 *obj* 是数据集 *DS* 中第 *i* 个数据对象,*LDS*(*obj*)是由 *obj* 与其 *K*-NN 构成的局部数据集.相对于 *LDS*(*obj*),*obj* 在各属性维上的局部稀疏程度(密度),可由如下局部稀疏因子λ_{ij}来度量:

$$\lambda_{ij}=\frac{\sum_{y\in p(x_{ij})}(y-x_{ij})^2}{K+1} \tag{8}$$

其中,$p(x_{ij})$是 *LDS*(*obj*)在属性维 A_j 上的属性取值组成的取值序列,*K* 是最近邻个数.

由公式(8)可知,局部稀疏因子λ_{ij}是局部数据集 *LDS*(*obj*)中属性维 A_j 上的各个取值到 x_{ij} 的欧式距离的平方均值.类似于公式(2):λ_{ij}越大,表明 *LDS*(*obj*)在属性维 A_j 上越稀疏;反之,λ_{ij}越小,表明 *LDS*(*obj*)在属性维 A_j 上越稠密.总之,λ_{ij}可以体现局部数据集 *LDS* 在其属性维 A_j 上的局部稀疏程度(密度).

显然,由公式(8)可以确定所有数据对象的各属性维上的局部稀疏因子,并由此可生成数据集 *DS* 的稀疏因子矩阵$[Z_\lambda]_{n\times d}=[\lambda_{ij}]$,其中,$n=|DS|$,$d=DS$ 的维度.由$[Z_\lambda]_{n\times d}$可构造出 *obj* 在 *LDS*(*obj*)上的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(obj)$,即,*obj* 的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(obj)$,其中,$k=|LDS(obj)|$.

为了刻画属性维的局部稀疏程度差异,可采用如下的局部稀疏差异因子 d_{ij},描述 *obj* 在 *LDS*(*obj*)上第 *j* 个属性维的稀疏程度差异:

$$d_{ij}=\sqrt{\left|\frac{(\lambda_{ij}-C_{\lambda_{ij}})}{C_{\lambda_{ij}}}\right|} \tag{9}$$

其中,$C_{\lambda_{ij}}$是$[Z_{L\lambda}]_{k\times d}(obj)$在属性维 A_j 的对应维取值集的平均值的平方.

由公式(9)可知,d_{ij}刻画了局部稀疏因子λ_{ij}相对于均值的差异程度.d_{ij}越大,数据对象 *obj* 的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(obj)$在第 *j* 维上的局部稀疏因子差异越大,也就是说,*obj* 的局部数据集 *LDS* 在属性维 A_j 上的取值的局部密度越不均匀,即,*obj* 的局部数据集 *LDS* 在属性维 A_j 上的取值越不均匀;反之,d_{ij}越小,数据对象 *obj* 的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(obj)$在第 *j* 维上的局部稀疏因子差异越小,也就是说,*obj* 的局部数据集 *LDS* 在属性维 A_j 上的局部密度越均匀,即,*obj* 的局部数据集 *LDS* 在属性维 A_j 上的取值越均匀.总之,采用 d_{ij}可以度量 *obj* 的局部数据集 *LDS* 在属性维 A_j 上取值是否均匀的程度.

定义 2. 设ε是局部稀疏差异因子阈值,d_{ij}是第 *i* 个数据对象 *obj* 第 *j* 维的局部稀疏差异因子,如果 $d_{ij}<\varepsilon$,则令 $v_{ij}=0$;反之,则令 $v_{ij}=1$.称 $v_i=\{v_{ij}\}$是 *obj* 的子空间定义向量,其中,$i=1,2,\ldots,|DS|$,$j=1,2,..d$,d 为 *DS* 的维度.在 v_i 中,由 v_{ij} 的值为 1 所对应的属性维组成的子空间称为 *obj* 的相关子空间,由 v_{ij} 值为 0 对应的属性维组成的子空间称为 *obj* 的不相关子空间.

定义 2 是对定义 1 的一种更广义的描述,当局部数据集在属性维上的取值不均匀的程度大于ε时,才能对离群数据挖掘提供有用的信息;反之,则不能提供有价值的信息.当局部稀疏差异因子 d_{ij} 小于其给定的ε阈值时,即可将该属性维归并为 *obj* 的不相关子空间中,因此由子空间定义向量,可以快速地确定该数据对象的相关或不相关子空间.

当数据集维度增加时,距离差异越来越不明显,采用距离度量数据对象之间的差异将变得无意义,但是最近邻的排序却是有意义的[29].

由公式(9)和定义 2 可知:在 *LDS*(*obj*)中,每个近邻 $loc_obj\in LDS(obj)$的局部稀疏因子都是在 *LDS*(*loc_obj*)上获得的,即,一个数据对象 *obj* 的子空间定义向量是由 *LDS*(*obj*)与 $LDS_{loc_obj\in LDS(obj)}(loc_obj)$在属性维上的取值来确定,因此能准确地反映局部数据的分布特征,保证了子空间定义向量的正确性.此外,由于子空间定义向量各维的取值都是由其对应属性维的取值的分布情况决定,确定子空间定义向量与数据集 *DS* 维度无关.

在文献[14]中,采用皮尔逊或斯皮尔曼相关获得相关子空间是线性的,体现出了局部数据集的线性分布特征,采用该相关子空间仅能发现明显偏离线性分布特征的离群数据.由定义 2 描述的相关子空间是由局部数据集的局部稀疏差异因子来确定,且相关子空间是由密度差异较明显的属性维构成,类似于文献[17]分析,子空间定义向量可以确定任意的非线性相关子空间,并可以较为准确地刻画密度稀疏的数据对象.

3.3 局部离群数据

由上述定义 2 可知:如果第 i 个对象 obj 不存在相关子空间,即,obj 的子空间定义向量 v 各维值都是 0,表明 obj 与局部分布特征一致;如果第 i 个对象 obj 存在着相关子空间,则表明 obj 与局部分布特征不一致.为了刻画不一致的程度(离群程度),结合定义 2,参照文献[26],在相关子空间中,obj 的离群程度($Factor(obj)$)重新描述如下:

设 v_i 是 obj 的子空间定义向量,d=v_i 的维度,λ_{ij} 是 obj 第 j 属性维对应的局部稀疏因子(见公式(8)),$d_{RS}(o,s)$表示数据对象 s 与 o 之间在相关子空间 RS 中的欧式距离,则参照公式(2)~公式(5),obj 相对于局部数据集 $LDS(obj)$,在其相关子空间 RS 中的标准距离为

$$\sigma_{RS}(obj,LDS(obj))=\sqrt{\frac{\sum_{o\in LDS(obj)}d_{RS}(obj,o)^2}{|LDS(obj)|}}=\sqrt{\left(\sum_{j=1}^{d}v_i\cdot\lambda_{ij}\right)} \tag{10}$$

在 obj 的相关子空间 RS 中,obj 的任意一个最近邻 $loc\_obj\in LDS(obj)$(设 loc_obj 是第 m 个数据对象)相对于 $LDS(loc_obj)$的标准距离为

$$\sigma_{RS}(loc\_obj,LDS(loc\_obj))=\sqrt{\frac{\sum_{o\in LDS(loc\_obj)}d_{RS}(loc\_obj,o)^2}{|LDS(loc\_obj)|}}=\sqrt{\left(\sum_{j=1}^{d}v_i\cdot\lambda_{mj}\right)} \tag{11}$$

参照公式(4),obj 在其相关子空间 RS 中的概率局部异常因子 PLOF 为

$$PLOF_{RS}(obj)=\frac{\sigma_{RS}(obj,LDS(obj))}{E_{o\in LDS(obj)}(\sigma_{RS}(o))}-1 \tag{12}$$

在 obj 的相关子空间 RS 中,$loc_obj\in LDS(obj)$的概率局部异常因子 PLOF 为

$$PLOF_{RS}(loc\_obj)=\frac{\sigma_{RS}(loc\_obj,LDS(loc\_obj))}{E_{o\in LDS(loc\_obj)}[\sigma_{RS}(o,LDS(o))]}-1 \tag{13}$$

在公式(12)和公式(13)中,$PLOF_{RS}(obj)$和 $PLOF_{RS}(loc_obj)$类似于在 obj 的相关子空间 RS 中的局部离群因子 LOF,刻画了 loc_obj 在 obj 的相关子空间 RS 中的局部离群程度.不难看出:若不采用相关子空间 RS,仅在 DS 所有维组成的空间 FS 中(即,RS=FS),$PLOF_{RS}(obj)$与 $PLOF_{RS}(loc_obj)$的定义是没有区别的.

PLOF 不是一个概率值,也未标准化.当 $PLOF\leqslant 0$ 时,obj 被刻画为正常数据对象;否则,当 $PLOF>0$ 时,PLOF 值越大,表明 obj 的离群程度越高.但采用 PLOF 度量方式,对于不同分布特征的数据集不具有可比性[26].数据集 DS 存在各种维度的相关子空间,不同相关子空间中的数据分布特征也不同.由此可见,不同相关子空间中的 PLOF 也不具可比性.

为了将 PLOF 值转化成一个概率值,文献[27]假设维之间相互独立,且各个维都对 PLOF 值产生影响.根据中心极限定理,可假设在所有维组成的空间 FS 中的 PLOF 服从 $E[PLOF^2]$(以 0 中心的方差)的正态分布[26].根据定义 2,小于ε时不能提供有价值的信息,即,在定义 2 确定的相关子空间中,其属性维才能对 PLOF 产生影响.因此,可假设相关子空间中的 $PLOF_{o\in LDS(obj)}(o)$服从 $E_{o\in LDS(obj)}[(PLOF_{RS}(o))^2]$的正态分布,并利用高斯误差函数来表示数据对象是离群数据的概率.参照公式(5),obj 在其相关子空间 RS 中的离群程度(离群因子)重新定义如下:

$$Factor(obj)=\max\left\{0,erf\left(\frac{PLOF_{RS}(obj)}{\sqrt{2}\cdot\sqrt{E_{o\in LDS(obj)\cup obj}[(PLOF_{RS}(o))^2]}}\right)\right\} \tag{14}$$

其中,erf 同文献[26],表示高斯误差函数.由于文献[26]中的参数λ不会影响到离群程度的排序,本质是将高斯误差函数的变量值除以λ,因此未考虑其值的选取,相当于文献[26]的λ取 1.

由公式(12)~公式(14)可以看出:$PLOF_{RS}(loc_obj)$并不是在 loc_obj 的相关子空间中确定,而是在 obj 的相关子空间 RS 中确定,因为确定 $Factor(obj)$时,obj 与局部的分布特征不一致的程度仅能在 obj 的相关子空间 RS 中体现.

在文献[26]中,假设各维对度量离群数据作用是相同的,且在所有维组成的空间中,计算数据对象的 PLOF.

但是,当将均匀分布的属性参与 PLOF 的计算且均匀分布的属性维较多时,有可能使公式(12)和公式(13)中的 PLOF 值趋近于 0.总之,均匀分布的属性维对于区分正常数据与离群数据不会提供有价值的信息,它的存在可能会将离群数据变成正常数据.公式(14)是在相关子空间 RS 中确定数据对象离群程度,RS 仅含有非均匀分布的属性维,有效地避免了均匀分布的属性维参与离群程度的计算过程.同时,公式(14)可以适应不同的数据分布特征以及不同的子空间,即,可以作为相关子空间中确定离群数据的统一度量方式,有效地改善了文献[20]中度量方式在多个子空间中不具可比性的问题.特别需要注意的是:文献[26]是从基于数据集角度归一化 PLOF,假设整体数据集的 PLOF 服从正态分布;而公式(14)是从广义的角度,同时考虑数据集中不同聚集簇的归一化,仅假设局部数据集在相关子空间中的 PLOF 服从正态分布.

4 离群数据挖掘的并行化

由公式(14)可知,数据集 DS 中的任意数据对象 $obj_i(i=1,2,...,n;n=|DS|)$ 的离群因子 $Factor(obj_i)$ 的计算步骤是:首先,由 KNN 方法确定 obj_i 的 $LDS(obj_i)$,并利用公式(8)计算 obj_i 对应的稀疏度因子 λ_i,从而生成 DS 的稀疏因子矩阵 $[Z_\lambda]_{n\times d}$;其次,根据 $[Z_\lambda]_{n\times d}$,参照 $LDS(o)(LDS(obj_i))$,生成对应的局部稀疏因子矩阵 $[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$,并由公式(9)与定义 2 生成 obj_i 对应的子空间定义向量 v_i;最后,由公式(14)确定 obj_i 对应的离群因数 $Factor(obj_i)$.因此在 DS 中,任意数据对象的离群因子计算过程如图 1 所示.

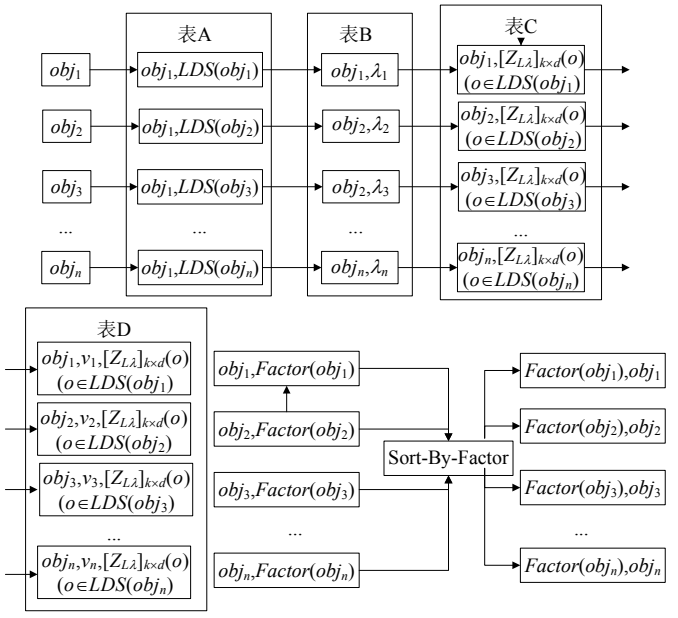


Fig.1 Calculating outlier factor

图 1 离群因子计算过程

由于各个数据对象的离群程度是其对应的 $LDS(obj_i)$ 在相关子空间下计算得到,由图 1 可以看出,在表 A 中,确定 $LDS(obj_i)$ 时采用了 KNN 的操作,需要遍历 DS 中的所有数据对象;在表 C 中,确定 $[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$ 时需要参照表 A 中的 $LDS(o)(o\in LDS(obj_i))$ 以及表 B 中 DS 对应的稀疏因子矩阵 $[Z_\lambda]_{n\times d}$;其他计算步骤都不需要遍历与 DS 相关的全局数据结构,数据对象之间的计算过程相互独立,不需要进行通信,因此可以直接并行处理.对于生成表 A 与表 C,可采用文献[22]的 LSH 策略进行并行化处理,但是其主要思想不相同:

- 在生成表 A 时,是文献[22]直接引用:首先,利用公式(7)将各个数据对象 Hash 散列到不同的哈希表的不同的哈希桶中;然后,在确定数据对象 obj_i 的近似 KNN 生成 $LDS(obj_i)$ 时,将数据对象 obj_i 相关的哈希桶

导入程序中进行操作.

- 在生成表 C 时,对文献[22]的方法进行了调整:首先,利用公式(7)将各个数据对象 Hash 散列到不同的哈希表的不同的哈希桶中;然后,将数据对象 obj_i 相关的哈希桶导入程序中进行操作,不但确定数据对象 obj_i 的近似 KNN 生成 $LDS(obj_i)$,间接生成$[Z_{L\lambda}]_{k\times d}(obj_i)$,而且,还要同时在导入的数据集中确定 $LDS(o)$ $(o\in LDS(obj_i)\cap o\neq obj_i)$,间接生成$[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i)\cap o\neq obj_i)$.

由于对表 A 与表 C 的生成采用了文献[22]的 LSH 策略,不必将全部数据集放入内存进行查询,因此,整个离群因子计算计算过程可以并行化.

图 1 刻画和描述了离群因子的计算过程,当数据集 *DS* 较小时,表 A 与表 B 都较小,可以首先生成表 A,并将其保存在内存中;然后由表 A 通过生成表 B,并将其也保存在内存中.由于表 A 与表 B 中都在内存,可采用索引技术,从表 B 中的对应项对应到表 A 的相应项来生成表 C.当数据集 *DS* 大到一定程度时,表 A 与表 B 都相当大.在生成表 C 时,需将表 A 中的每一个数据对象和对应局部数据集 $LDS(obj_i)$逐项输入,以及在表 B 中查找 $LDS(obj_i)$对应的稀疏度因子,因此,表 B 必须被保留在各计算结点上的内存中.当 B 表太大时,会耗尽各个计算结点的内存资源.因此,为了生成表 C,不再采用索引策略对应生成,而是从表 B 中查找 $LDS(obj_i)$对应的局部稀疏因子,采用类似于表 A 中生成 $LDS(obj_i)$的操作.由图 1 可以看出:生成表 A 时,需要遍历 *DS* 中的全部数据对象;生成表 C 时,需要遍历表 A 中的全部数据对象.因此,生成表 A 和表 C 是全局相关的,其他过程都与全局无关,各数据对象对应操作彼此之间不存在通信.因此,可采用文献[22]的分布式策略对表 A 与表 C 的并行生成,且不需要将表 B 中的全部内容放在内存中.

MapReduce 是一个编程模型,其主要操作分为 Map 和 Reduce 两个阶段,每个阶段的输入和输出都是基于键值对.在 Map 阶段,Map 函数将输入文件中的每一行变为键值对$(K1,V1)$的形式,经过 map 函数的处理,输出多个新的键值对 $List(K2,V2)$;在 Reduce 阶段,所有的 Map 阶段的输出按键分组$(K2,List(V2))$,该过程称为混洗(shuffle).每组$(K2,List(V2))$作为 Reduce 函数的输入,经过 Reduce 函数的处理,输出最终的结果$(K3,V3)$.在计算局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$时,可采用文献[22]的分布式策略 LSH,但要改变步骤,即,在 obj_i 索引值关联的数据集中查询 $LDS(o)(o\in LDS(obj_i))$;在计算稀疏度因子λ和离群因子 *Factor* 时,都可由 Map 来实现;在按 *Factor* 大小进行全排序时,需要一个 Map 对 Factor 进行取样,进而实现决定各个$(K2,V2)$分配到哪个结点的 Partition 函数,从而有效地解决了数据分配不均匀问题和局部有序但全局无序问题.需要一个 Reduce 来对分配到各个结点的$(K2,V2)$进行排序,其 MapReduce 编程模型的实现过程如图 2 所示.

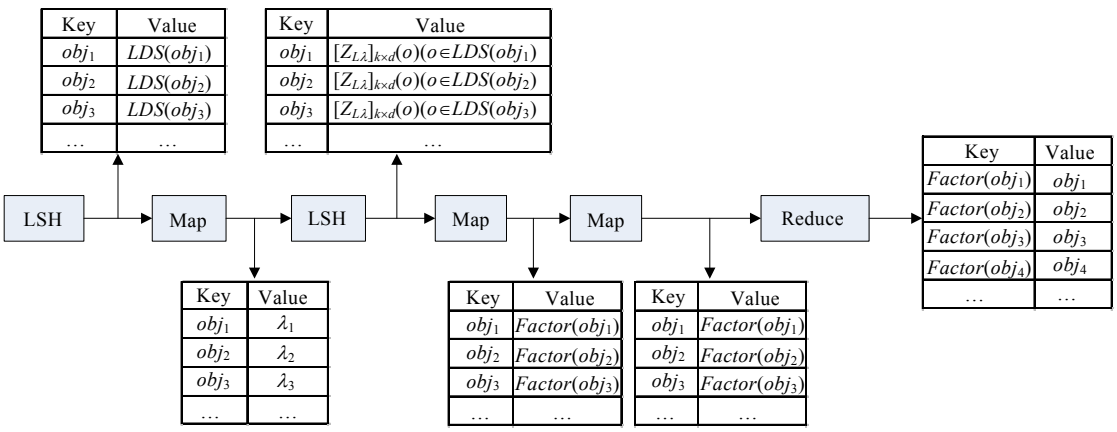


Fig.2 Implementation process of MapReduce program

图 2 MapReduce 程序的实现过程

由图 2 可知,在 MapReduce 编程模型中,离群因子计算过程如下:

(1) 采用文献[22]的分布式策略,确定各个数据对象的 $LDS(obj_i)$;

(2) 采用一个 Map 函数,计算各个数据对象对应的稀疏度因子 λ_i;

(3) 采用文献[22]的分布式策略,确定 $LDS(o)(o\in LDS(obj_i))$对应的局部稀疏因子矩阵:

$$[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i));$$

(4) 采用一个 Map 函数,计算各个数据对象对应的子空间定义向量 v_i,结合$[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$计算对应的离群因子 *Factor*;

(5) 采用一个 Map 函数和一个 Reduce 函数,将各个数据对象按对应的离群因子 *Factor* 进行全排序并输出结果.

5 基于相关子空间的离群数据并行挖掘算法

5.1 并行算法

在图 2 中,

- 采用 LSH 策略计算数据对象 *obj* 的近似 *K*-NN,并确定 *DS* 中各个数据对象 obj_i 的局部数据集 $LDS(obj_i)$,可由两个 MapReduce 任务来实现;
- 计算 obj_i 各属性值的局部稀疏因子 λ_{ij},以(obj_i, λ_i)作为输出并保存,可由 MapReduce 任务中的 Map 阶段来实现;
- 生成 obj_i 的 $LDS(o)(o\in LDS(obj_i))$对应的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$也可采用 LSH 策略,并由两个 MapReduce 任务来实现;
- 根据 obj_i 对应的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(obj_i)$,由公式(9)计算 obj_i 各属性维对应的局部稀疏差异因子,并由定义 2 确定 obj_i 对应的子空间定义向量,并计算 obj_i 的离群因子,可由 MapReduce 任务中的 Map 阶段来实现;
- 根据 *Factor* 值的大小对数据对象并行排序,可由一个 MapReduce 任务来实现;
- 将离群程度明显较大的 *n* 个数据对象作为离群数据.

其并行算法描述如下:

算法. RSLODA(related-subspace-based local outlier detection algorithm using MapReduce).

输入:数据集 *DS*(属性个数为 *d*)、近邻数 *K*、稀疏度差异因子阈值 ε.

输出:*n* 个离群数据.

(1) 执行 LSH 中的 MapReduce 任务,生成{(*obj*,*LDS*(*obj*))}; //*并行计算数据对象 *obj* 的 *LDS*(*obj*)

(2) 以{(*obj*,*LDS*(*obj*))}作为输入,执行 MapReduce 任务,生成{(*obj*,λ)}; //*确定 *DS* 中每个数据对象 *obj* 对应的稀疏度因子 λ,见公式(8)

(3) 以{(*obj*,λ)}作为输入,执行 LSH 中的 MapReduce 任务,生成{(*obj*,($[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$)}; //*确定 *DS* 中每个数据对象 *obj* 的 *LDS*(*obj*)对应的局部稀疏因子矩阵$[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$

(4) 以{(*obj*,($[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$)}作为输入,执行 MapReduce 任务,生成{(*obj*,*Factor*(*obj*))}; //*确定 *DS* 中每个数据对象 *obj* 对应的离群因子 *Factor*(*obj*),见公式(14)

(5) 以{(*obj*,*Factor*(*obj*))}作为输入,执行 MapReduce 任务,对{(*obj*,*Factor*(*obj*))}按 *Factor* 大小进行全排序; //*确定 *DS* 中每个数据对象 *obj* 按对应的离群因子 *Factor*(*obj*)的大小进行全排序

(6) 输出离群程度最大的 *n* 个数据对象. //*选取 *Top*(*N*)作为离群数据

算法说明:

1. 在上述 RSLODA 并行算法中,步骤(1)、步骤(3)中的 MapReduce 任务是实现文献[15]中的 LSH 策略,即查询各个数据对象的近似 KNN,文献[15]对其处理过程给出了详细的分析;步骤(5)中的 MapReduce 对其结果先进行采样,然后进行并行全排序[30].

2. 在 RSLODA 中,步骤(2)的 MapReduce 任务执行流程是:

receive {(*obj*,*LDS*(*obj*))}

For each (*obj*,*LDS*(*obj*)) in {(*obj*,*LDS*(*obj*))}

Map:

1) For (*m*=1; *m*<=*d*; *m*++) {

2) For (*j*=1; *i*<=*K*+1; *j*++){

3) $\lambda Set[j]=(LDS[j][m])$; //*将 *L*[*i*]对应的局部数据的第 *m* 列各属性值加入到一个数组中

4) }

5) λ_{im}=*computer* $\lambda_{im}(\lambda Set)$; //*根据公式(8)计算 *L*[*i*]在第 *m* 维上对应属性值的稀疏因子

6) $\lambda_i[m]=\lambda_{im}$; //*将 *L*[*i*]在第 *m* 维上属性值对应的稀疏因子加放到一个数组中

7) }

8) *emit* ⟨*obj*,λ⟩

3. 在 RSLODA 中,步骤(4)中的 MapReduce 执行流程是:

receive {(*obj*,($[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$))}

For each (*obj*,($[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$)) in {(*obj*,($[Z_{L\lambda}]_{k\times d}(o)(o\in LDS(obj_i))$)}

Map:

1) For (*j*=1; *j*<=*d*; *j*++) {

2) *x*=0;

3) For (*m*=1; *m*<=*K*+1; *m*++){

$x+=[Z_{L\lambda}]_{k\times d}(obj)[m][j]$;

4) }

5) $d_{ij}(obj)$; //*公式(9),计算第 *i* 个数据对象第 *j* 列属性值对应的局部稀疏差异因子

6) If ($d_{ij}(obj)<\varepsilon$)

7) {$v_i[j]$=0;

8) } else {$v_i[j]$=1;}

9) }

10) If ($\|v_i\|_1$=0){

11) *Factor*(*obj*)=0;

12) }else{

13) *Factor*(*obj*); //*根据公式(14)计算第 *i* 个数据对象的离群程度

14) }

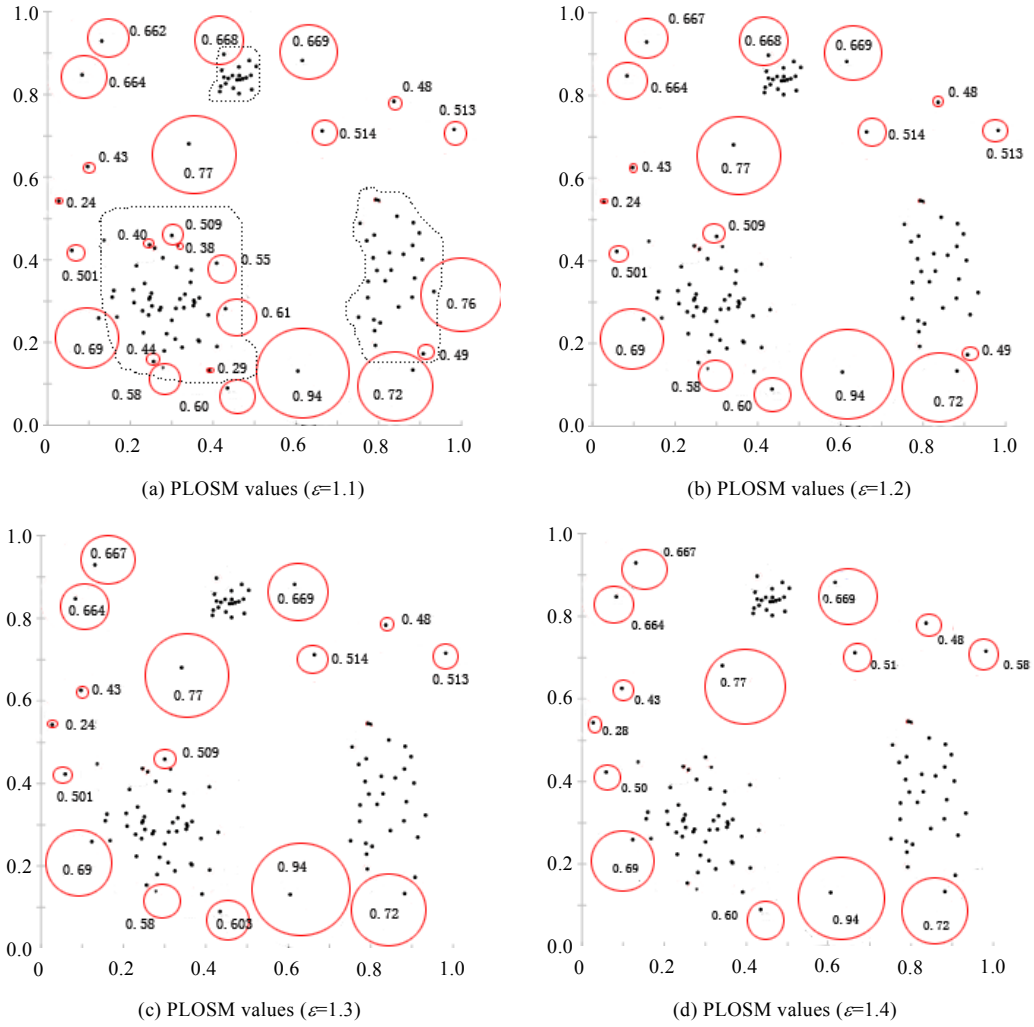
15) *emit* ⟨*obj*,*Factor*(*obj*)⟩

5.2 实例分析

在文献[26]中,给出一个由 3 个聚聚族和离群数据组成的二维数据实例,其中含有 100 条正常数据对象和 10 条离群数据.RSLODA 算法在伪分布下,对该实例运行结果如图 3 所示,其中,LSH 策略中,参数 *k*=1 和 *w*=50,哈希表个数取 4,KNN 个数=20.

在图 3(a)中,数据集含有较为明显的 3 个聚集簇(见虚线框),且其分布特征明显不同.在聚集簇边界与边界以外的数据对象(用圆圈标记)被标记为局部离群数据,表明该数据对象不服从或偏离了对应聚集族的数据分布特征,其圆圈中标记数字为该数据对象的离群因子值.由此可见,RSLODA 算法可以有效地发现不服从或偏离不同聚集族的数据分布特征的局部离群数据.

由图 3(a)~图 3(d)可知:随着ε的增大,不存在相关子空间的数据对象(即正常数据)越来越多.例如,图 3(d)中未标记离群程度值的正常数据个数要比图 3(a)~图 3(c)中的要多.也不难发现,有些标记有离群程度的数据对象,由于其局部数据集较难体现聚集簇的分布特征时离群程度值偏小,例如,图 3(f)中离群程度为 0.28 的数据对象.



(a) PLOSM values (ε=1.1)

(b) PLOSM values (ε=1.2)

(c) PLOSM values (ε=1.3)

(d) PLOSM values (ε=1.4)

Fig.3 Data distribution and local outliers

图 3 数据分布特征与局部离群数据

6 实验结果与分析

实验环境:伪分布环境为 1 台 LENOVO 台式机(Pentium D925 CPU,512M 内存);全分布环境为由 16 个计算结点构成的集群,其中 5 个计算结点是在服务器(内存 8G)上搭建的虚拟机(内存 512M),其余 11 个计算结点是 LENOVO 台式机(Pentium D925 CPU,512M 内存);操作系统为 ubuntu Linux 10.10;实验平台:Jdk 1.6.0_37, hadoop1.1.1,juno eclipse.采用 Java 语言作为开发工具,实现了 RSLODA,SOD[20],LoOP[26],COP[14]算法(SOD, LoOP,COP 实现代码参照网址:http://elki.dbs.ifi.lmu.de/wiki#).

实验数据集:

1) 人工数据集

参照文献[14,20,26]中的方式,分别生成了 25 维,50 维,75 维,100 维,125 维,150 维,175 维和 200 维的 3 000 条 8 个人工数据集.在每一个人工数据集中,包含有 30 条离群数据以及 3 个分别由 990 条数据构成的聚类簇(cluster),其中,第 1 个簇中 15 个属性维、第 2 个族中 20 个属性维和第 3 个簇中 25 个属性维的取值分别服从正

态分布,其余属性维取值服从均匀分布;生成了 8 维 3 000 条人工数据集,其中含有 30 条离群数据以及 3 个分别由 990 条数据构成的聚类簇(cluster),其中,第 1 个簇中 3 个属性维、第 2 个族中 4 个属性维和第 3 个簇中 5 个属性维的取值分别服从正态分布,其余属性维取值服从均匀分布;生成了 200 维 500 万条人工数据集,其中含有 1 000 条离群数据以及 100 个由 49 990 条数据构成的簇 Cluster,且从第 1~25 个簇中的 15 个属性维、第 26~50 个簇中的 20 个属性维、第 51~75 个簇中的 25 个属性维和第 76~100 个簇中的 30 个属性维的取值分别服从正态分布,其余属性维取值服从均匀分布;

2) 恒星光谱数据集

国家天文台提供的恒星光谱特征线数据[2],并选取和生成数据量分别为 14.37G(200 维、500 万条),12.59G(175 维、500 万条光谱),10.80G(150 维、500 万条光谱),9.01G(125 维、500 万条光谱),7.21G(100 维、500 万条光谱),5.51G(75 维、500 万条光谱),3.63G(50 维、500 万条光谱),1.92G(25 维、500 万条光谱),11.53G(200 维、400 万条光谱),8.63G(200 维、300 万条光谱),5.75G(200 维、200 万条光谱),2.87G(200 维、100 万条光谱)作为本文的实验数据集.

根据文献[22]的实验结果分析,公式(6)中的参数 k=20,w=500;多一个哈希表就会多一份数据的复本,同时会增加哈希桶的数量,加大 Hadoop 处理多文件的压力,对精度的影响却较小.考虑到存储容量的限制与文献[22]的实验结果分析,实验中的哈希表个数与文献[22]相同,取 4.

6.1 人工数据集

采用人工数据集作为测试数据,实验验证了 RSLODA 算法可有效地确定局部数据的相关子空间,并能够适用于不同分布的数据集,具有较高的挖掘精度.

6.1.1 ε,K 参数对并行挖掘精度和效率的影响

由图 4(a)可知:当局部稀疏差异因子阈值ε取值较小或较大时,挖掘的精度较低.其主要原因是:

- 当ε取值较小时,根据定义 2,大多数属性维被认为密度差异明显,并都会属于构成相关子空间中的维,从而导致相关子空间的维度扩大,受“维灾”影响严重;其次,相关子空间的维度扩大,可能无法挖掘出隐藏在子空间中的离群数据,因此其挖掘准确率降低.
- 当ε取值较大时,大多数属性维被认为密度差异不明显,并都不会属于构成相关子空间中的维,容易将密度差异较明显的离群数据被认为是正常数据,因此其挖掘准确率也会降低.

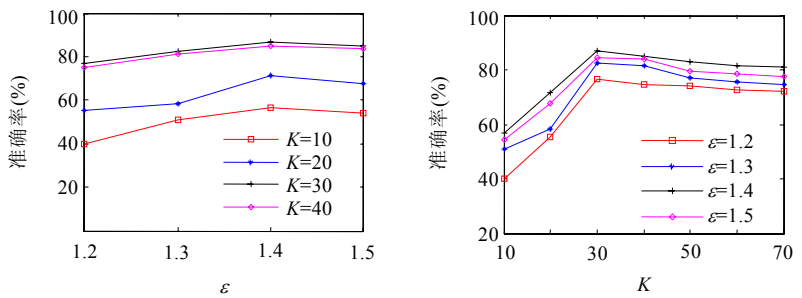


Fig.4 Accuracy impact of parameter ε, K (5×10^6 records, *dimensions*=200, *node*=16)

图 4 参数ε,K 对准确率的影响(200 维、500 万条人工数据集,计算结点=16)

由图 4(b)可知,在一定范围内,随着 K 的取值增加,其挖掘准确率也将提高.其主要原因是:公式(14)的前提是假设局部数据的 PLOF 服从正态分布,因此只有局部数据达到一到数量,才能较好地符合假设条件.不难发现:K=30 时的准确率达到最高;之后,随着 K 值的增大,准确率会出现微小的减小.其主要原因是,RSLODA 算法采用 LSH 策略计算每个数据对象的近似 KNN,当 K 取值较大时,由近似 KNN 得到的离群因子可能会出现较小误差,

说明 K 的取值并不是越大越好.选择一个好的 K 值,会使算法的准确率达到最好.

总之,选择合适的ε,K 值,会使算法的准确率达到最好.

由图 5 可知:局部稀疏差异因子阈值ε阈值与最近邻个数 K 值对 RSLODA 挖掘效率的影响较大,随着ε阈值的增大或者 K 的减小,其挖掘效率将提高;反之亦然.其主要原因是:RSLODA 的主要计算量集中在采用 LSH 策略寻找 K 最近邻以及在相关子空间中计算离群因子,随着ε取值的增大,子空间定义向量全 0 的情况增多,空间定义向量全 0 时,无需计算其局部稀疏差异因子(见公式(9)),且ε阈值不会影响寻找 K 最近邻,因此挖掘效率将提高;随着 K 取值的增大,查询 K 个最近邻的计算量会增多,同时,与 K 相关的向量函数(见公式(7))、局部稀疏因子(见公式(8))和局部稀疏差异因子(见公式(9))的计算量会增加,且 K 的取值不影响ε阈值对相关子空间的确定,因此其挖掘效率会降低.

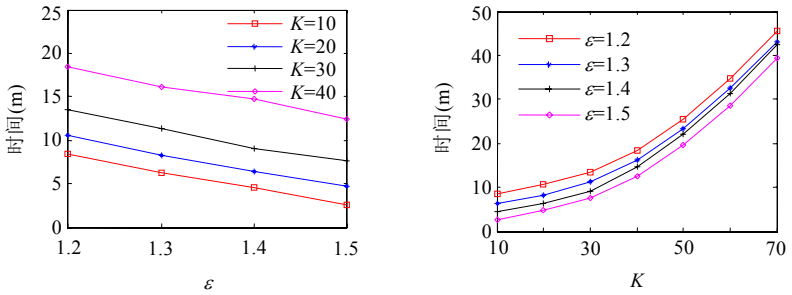


Fig.5 Efficiency impact of parameter ε, K (5×10^6 records, *dimensions*=200, *node*=16)

图 5 参数ε,K 对效率的影响(200 维、500 万条人工数据集,计算结点=16)

6.1.2 不同算法准确率的对比

如图 6 所示,RSLODA 算法在伪分布式环境下执行.

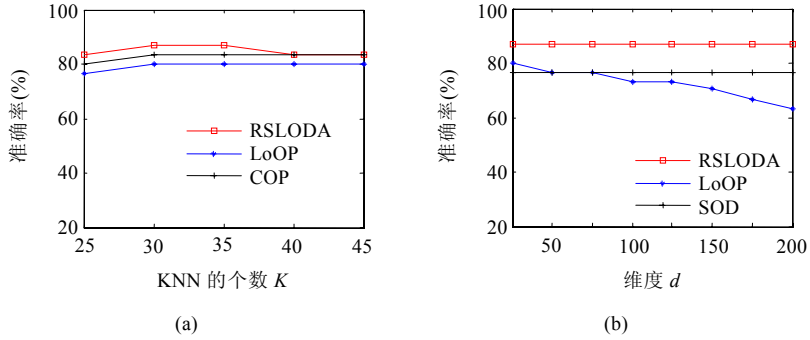


Fig.6 Accuracy comparison of different algorithms (ε=1.4)

图 6 不同算法准确率的对比(ε=1.4)

图 6(a)是 3 000 条 8 维人工数据集的实验结果.

- 当 K 较小时,局部离群数据挖掘算法 RSLODA,LoOP,COP 挖掘准确率较低.
- 当 K 增大到一定值时,其准确率会上升到一个较高的值.其主要原因是:当 K 增大到一定时,局部的最近邻已经能够较好地体现局部数据的疏密特征或分布趋势,且当 RSLODA,LoOP,COP 中的参数 K 取 30 时,准确率达到最高.
- 当 K>30 时,RSLODA 算法的准确率略有下降.这是因为:当 K 取值较大时,由近似 KNN 得到的离群因

子可能会出现较小误差;RSLODA 算法在 $K\in[30,40]$时,其挖掘准确率都比 LoOP 算法和 COP 算法要高,其主要原因是:LoOP 算法是在所有维构成的空间中采用高斯误差函数来度量离群程度,会找不到子空间中的离群数据,COP 算法仅能发现局部数据集的线性相关子空间.

图 6(b)是不同维度 3 000 条人工数据集的实验结果,且 K 值均取 30.当维度增加时,

- LoOP 的准确率会降低.其主要原因是 LoOP 算法是在所有维构成的空间中挖掘离群数据,随着维度的增加,相应地无关维也会增加,有些离群数据可能被隐藏,且受“维灾”的影响也逐渐严重.
- SOD 和 RSLODA 的准确率不变,其主要原因是:它们都是相关子空间中的离群挖掘算法,只要人工数据的相关子空间维度不会发生较大的变化,其挖掘准确率就基本不受维度的影响.

RSLODA 算法在任何数据维度下,其挖掘准确率都比 LoOP 和 SOD 算法要高.其主要原因是:LoOP 和 RSLODA 算法虽然都采用高斯误差函数来度量离群程度,可以有效地应用于不同分布的数据集,但 LoOP 受“维灾”影响较严重,而 SOD 算法当数据对象分布在两个或两上以上的子空间中时,将不能区分它们的离群程度[15].

6.2 天体光谱数据集

采用天体光谱数据集作为测试数据,实验验证数据维度对 RSLODA 算法挖掘效率的影响以及 RSLODA 算法的可伸缩性和可扩展性.

6.2.1 维度对算法效率的影响

在图 7 中,随着维度的变化,图 7(a)体现了 RSLODA 算法的挖掘耗时,图 7(b)体现了 RSLODA 算法挖掘耗时的时间比值.由图 7 可以看出,随着维度按比例增加,

1) 确定各个数据对象的哈希桶以及按索引值将其关联的哈希桶导入程序中时的计算量(见公式(7))、确定数据对象对应 KNN 中计算距离的计算量、计算局部稀疏因子的计算量(见公式(8))以及局部稀疏差异因子的计算量(见公式(9))都与数据维度 d 呈线性关系,也都会按比例增加;
2) 计算离群因子的计算量(见公式(14))仅与相关子空间维度 d_r 呈线性关系;
3) 恒星光谱数据集与人工数据集不同,随着维度的增加,会有较大比例的属性维增加到相关子空间中.

因此,当计算结点与数据条数不变时,其挖掘效率按略高于线性趋势提高.

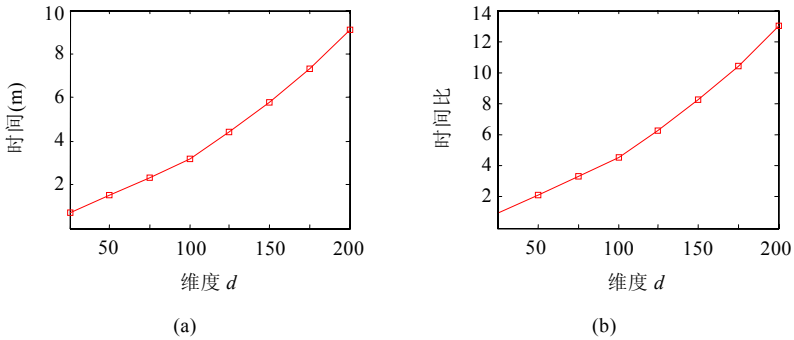


Fig.7 Efficiency impact of dimension (5×10^6 stellar spectral records, K=30, ε=0.1, *node*=16)

图 7 维度对效率的影响(500 万条光谱数据集,K=30,ε=0.1,计算结点=16)

6.2.2 伸缩性

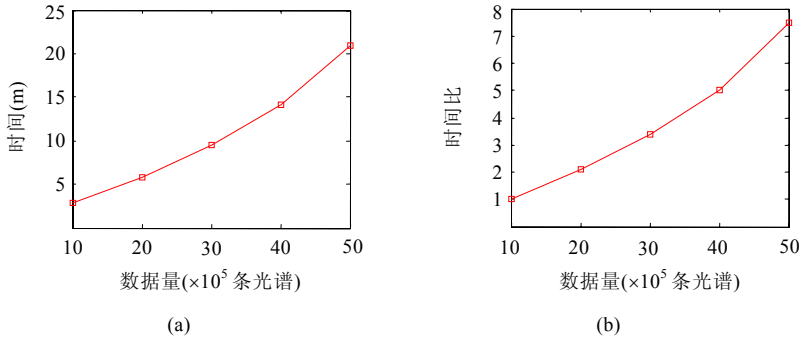
在图 8 中,随着数据量的变化,图 8(a)体现了 RSLODA 算法的挖掘耗时,图 8(b)体现了 RSLODA 算法挖掘耗时的时间比值.由图 8 可知:当计算结点不变时,随数据量的增加,其挖掘效率降低.其主要原因是:

- 首先,随着数据量的增加,数据对象个数的增多,各结点上分配的数据对象个数基本按计算结点比例线性增加;同时,各个数据对象在计算其离群因子的整个过程中的计算量会随着数据对象的增加,分配到各个计算结点的数据对象也按平均比例增多;由于各个数据对象的散列值相同的数据对象数也可能

增多,LSH 策略是按数据对象与其散列值相同的数据对象中查询 KNN,每个数据对象对应的 KNN 查询的时间与散列值相同的数据对象集的个数 n 是 $o(n)$的关系,即使借助索引也只能降到 $o(\log n)$,其他计算过程的计算量与数据对象个数的比值近似是定值,对应的计算量可以按计算结点比例线性分配.

- 其次,数据量的增加还会增加 Shuffle 的消耗.

总之,随数据量的增加,其挖掘效率曲线的倾斜程度要略高于线性.



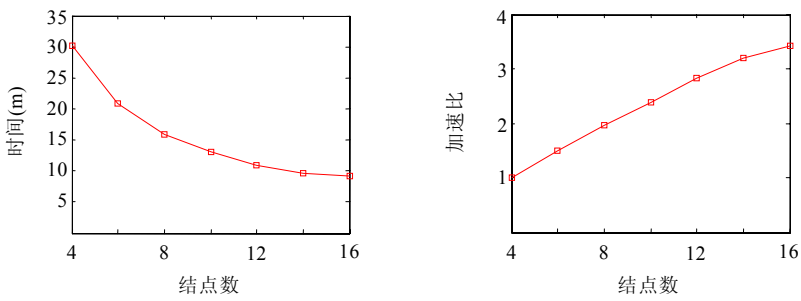
(a) (b)

Fig.8 Efficiency impact of data record (*dimensions*=200, *K*=30, ε=0.1, *node*=6)

图 8 数据量对挖掘效率的影响(200 维光谱数据,*K*=30,ε=0.1,计算结点=6)

6.2.3 可扩展性

在图 9 中,随着计算结点数的增加,图 9(a)体现了 RSLODA 算法挖的掘耗时,图 9(b)体现了 RSLODA 算法的加速比值.



(a) (b)

Fig.9 Efficiency impact of computer node (5×10^6 stellar spectral records, *dimensions*=200, *K*=30, ε=0.1)

图 9 计算结点对挖掘效率的影响(500 万条光谱数据,200 维,*K*=30,ε=0.1)

由图 9 可以看出,当数据量一定时,随着集群计算结点个数的增加,RSLODA 算法的挖掘耗时基本按计算结点比例减小,体现了 RSLODA 算法具有较好的并行程度.其主要原因是:

- 首先,算法 RSLODA 中各个数据对象的离群因子的所有计算过程完全可以并行化,各计算结点的数据对象个数可以按计算结点比例分配.由于 RSLODA 算法采用 LSH 策略寻找 K 最近邻,计算结点的增加并不会影响到各数据对象对应的索引值,即,每个结点对应的查询近似 KNN 的计算量不会改变.因此,近似 KNN 的计算量随着数据对象分配,基本按比例分配到各计算结点.同样,其他计算过程的计算量与数据对象个数呈线性关系,数据对象也按比例分配到各计算结点;
- 其次,随着计算结点的增加也会增加一些网络传输量,但 RSLODA 算法只有一个 Shuffle 过程,受其影响相当小;随着计算结点的增加,由于 Hadoop 已逐渐无法有效地为各计算结点均匀地划分任务,因此并

行化效果会逐渐降低.

7 结束语

在高维海量数据数据集中,从由有意义维构成的相关子空间中实现数据挖掘挖掘任务是数据挖掘领域研究热点和难点之一.本文利用 MapReduce 编程模型的强大数据处理能力,提出一种基于相关子空间的局部离群数据挖掘算法.该算法利用属性维在局部数据集上的稀疏程度,重新定义了相关子空间,有效地避免了无关属性维对离群数据的影响,并能有效地发现隐藏在子空间中的局部离据数据;充分利用了集群系统和 MapReduce 编程模型的数据处理能力,具有较好挖掘效果以及可扩展性和可伸缩性等,并能有效地适应于海量高维数据.下一步研究的工作是:局部数据集较难体现聚集簇的分布特征时,其离群程度值偏小的问题.

References:

[1] Knox EM, Ng RT. Algorithms for mining distance-based outliers in large datasets. In: Proc. of the Int'l Conf. on Very Large Data Bases. 1998. 392−403.

[2] Zhang JF, Jiang YY, Hu LH, Cai JH, Zang SL. A concept lattice based recognition method of celestial spectra outliers. Acta Automatica Sinica, 2008,34(9):1060−1066 (in Chinese with English abstract).

[3] Pham N, Pagh R. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. In: Proc. of the 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2012. 877−885. [doi: 10.1145/2339530.2339669]

[4] Sequeira K, Zaki M. ADMIT: Anomaly-Based data mining for intrusions. In: Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2002. 386−395. [doi: 10.1145/775047.775103]

[5] Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J. A comparative study of anomaly detection schemes in network intrusion detection. Technical Report, University of Minnesota, 2003.

[6] Liu H, Shah S, Jiang W. On-Line outlier detection and data cleaning. Computers & Chemical Engineering, 2004,28(9):1635−1647.

[7] Barnett V, Lewis T. Outliers in Statistical Data. New York: Wiley, 1994. [doi: 10.1016/j.compchemeng.2004.01.009]

[8] Tao Y, Xiao X, Zhou S. Mining distance-based outliers from large databases in any metric space. In: Proc. of the 12th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM, 2006. 394−403.

[9] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. ACM SIGMOD Record, 2000, 29(2):427−438. [doi: 10.1145/335191.335437]

[10] Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: Identifying density-based local outliers. ACM SIGMOD Record, 2000,29(2):93−104. [doi: 10.1145/335191.335388]

[11] Papadimitriou S, Kitagawa H, Gibbons PB, Faloutsos C. Loci: Fast outlier detection using the local correlation integral. In: Proc. of the 19th Int'l Conf. on Data Engineering (ICDE). IEEE, 2003. 315−326. [doi: 10.1109/ICDE.2003.1260802]

[12] Sarawagi S, Agrawal R, Megiddo N. Discovery-Driven Exploration of OLAP Data Cubes. Berlin, Heidelberg: Springer-Verlag, 1998. [doi: 10.1007/BFb0100984]

[13] Kriegel HP, Zimek A. Angle-Based outlier detection in high-dimensional data. In: Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2008. 444−452. [doi: 10.1145/1401890.1401946]

[14] Kriegel HP, Kroger P, Schubert E, Zimek A. Outlier detection in arbitrarily oriented subspaces. In: Proc. of the 2012 IEEE 12th Int'l Conf. on Data Mining. IEEE Computer Society, 2012. 379−388. [doi: 10.1109/ICDM.2012.21]

[15] Muller E, Schiffer M, Seidl T. Statistical selection of relevant subspace projections for outlier ranking. In: Proc. of the IEEE 27th Int'l Conf. on Data Engineering (ICDE). IEEE, 2011. 434−445. [doi: 10.1109/ICDE.2011.5767916]

[16] Aggarwal CC, Yu PS. Outlier detection for high dimensional data. ACM Sigmod Record, 2001,30(2):37−46. [doi: 10.1145/376284.375668]

[17] Keller F, Muller E, Bohm K. HiCS: High contrast subspaces for density-based outlier ranking. In: Proc. of the 28th Int'l Conf. on Data Engineering (ICDE). IEEE, 2012. 1037−1048. [doi: 10.1109/ICDE.2012.88]

[18] Aggarwal CC, Philip SY. An effective and efficient algorithm for high-dimensional outlier detection. The VLDB Journal, 2005, 14(2):211−221. [doi: 10.1007/s00778-004-0125-5]

[19] Zhang JF, Jiang YY, Chang KH, Zhang SL, Cai JH, Hu LH. A concept lattice based outlier mining method in low-dimensional subspaces. Pattern Recognition Letters, 2009,30(15):1434−1439. [doi: 10.1016/j.patrec.2009.07.016]

[20] Kriegel HP, Kröger P, Schubert E, Zimek A. Outlier detection in axis-parallel subspaces of high dimensional data. In: Proc. of the 13th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining. Berlin, Heidelberg: Springer-Verlag, 2009. 831−838. [doi: 10.1007/978-3-642-01307-2_86]

[21] Bouguessa M, Wang S. Mining projected clusters in high-dimensional spaces. IEEE Trans. on Knowledge and Data Engineering, 2009,21(4):507−522. [doi: 10.1109/TKDE.2008.162]

[22] Stupar A, Michel S, Schenkel R. RankReduce—Processing *k*-nearest neighbor queries on top of MapReduce. In: Proc. of the 8th Workshop on Large-Scale Distributed Systems for Information Retrieval. 2010. 15−20.

[23] Lu W, Shen YY, Chen S, Ooi BC. Efficient processing of *k* nearest neighbor joins using mapreduce. Proc. of the VLDB Endowment, 2012,5(10):1016−1027. [doi: 10.14778/2336664.2336674]

[24] Liu Y, Jing N, Chen L, Xiong W. Algorithm for processing *k*-nearest join based on R-tree in MapReduce. Ruan Jian Xue Bao/ Journal of Software, 2013,24(8):1836−1851(in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4377.htm [doi: 10.3724/SP.J.1001.2013.04377]

[25] Angiulli F, Basta S, Lodi S, Sartori C. Distributed strategies for mining outliers in large data sets. IEEE Trans. on Knowledge and Data Engineering, 2013,25(7):1520−1532. [doi: 10.1109/TKDE.2012.71]

[26] Kriegel HP, Kröger P, Schubert E, Zimek A. LoOP: Local outlier probabilities. In: Proc. of the 18th ACM Conf. on Information and Knowledge Management. ACM Press, 2009. 1649−1652. [doi: 10.1145/1645953.1646195]

[27] Zhang C, Li F, Jestes J. Efficient parallel *k*NN joins for large data in MapReduce. In: Proc. of the 15th Int'l Conf. on Extending Database Technology. ACM Press, 2012. 38−49. [doi: 10.1145/2247596.2247602]

[28] Datar M, Immorlica N, Indyk P, Mirrokni VS. Locality-Sensitive hashing scheme based on p-stable distributions. In: Proc. of the 20th Annual Symp. on Computational Geometry. ACM Press, 2004. 253−262. [doi: 10.1145/997817.997857]

[29] Houle ME, Kriegel HP, Kröger P, Schubert E, Zimek A. Can shared-neighbor distances defeat the curse of dimensionality. In: Proc. of the Scientific and Statistical Database Management. Berlin, Heidelberg: Springer-Verlag, 2010. 482−500. [doi: 10.1007/978-3-642-13818-8_34]

[30] White T. Hadoop: The Definitive Guide. O'Reilly, 2012.

附中文参考文献:

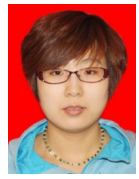
[2] 张继福,蒋义勇,胡立华,蔡江辉,张素兰.基于概念格的天体光谱离群数据识别方法.自动化学报,2008,34(9):1060−1066.

[24] 刘义,景宁,陈荦,熊伟.MapReduce 框架下基于 R-树的 *k*-近邻连接算法.软件学报,2013,24(8):1836−1851. http://www.jos.org.cn/1000-9825/4377.htm [doi: 10.3724/SP.J.1001.2013.04377]

张继福(1963－),男,山西平遥人,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据挖掘与人工智能,并行与分布式计算.

秦啸(1974－),男,博士,副教授,博士生导师,主要研究领域为并行与分布式计算,数据密集型计算,存储器节能.

李永红(1988－),男,硕士生,主要研究领域为数据挖掘,并行计算.



荀亚玲(1980－),女,博士生,讲师,主要研究领域为数据挖掘,并行计算.