

一种基于 SOM 划分的 FP-growth 算法

郝奎奎, 刘海滨

(中国航天系统科学与工程研究院, 北京 100048)

摘要: FP-growth 算法只能处理较小数据集, 在面对海量数据集时显得无能为力。对此, 对 FP-growth 算法的挖掘过程进行改进, 提出一种基于 SOM (self-organizing map) 划分的 FP-growth 算法。在数据预处理阶段, 将原始数据中的每条事务标准化为相同维度的数据; 考虑到大数据集较难处理的问题, 首先利用系统抽样方法从大数据集中抽取具有代表性的样本; 由于包含频繁项的事务具有较小的欧氏距离, 再对样本进行 SOM 聚类分析; 根据聚类结果, 将大数据集分成若干个子集, 在各个子集上并行进行 FP-growth 算法挖掘。实验结果表明, 改进算法降低了内存占用量, 缩短了数据挖掘时间, 提高了对海量数据的处理能力和效率, 并且具有较好的加速比。

关键词: FP-growth; 自组织映射; 数据挖掘; 聚类; 数据划分

中图分类号: TP181

文献标识码: A

文章编号: 1673-629X(2018)04-0071-06

doi: 10.3969/j.issn.1673-629X.2018.04.015

A FP-growth Algorithm Based on SOM Partition

JIA Kui-kui, LIU Hai-bin

(China Aerospace Systems Science and Engineering Research Institute, Beijing 100048, China)

Abstract: FP-growth algorithm can only handle smaller data sets, and can't do much in the face of massive data sets. For this, we improve the mining process of FP-growth and propose a FP-growth algorithm based on SOM partition. In the data preprocessing, each transaction in the original data is normalized to the same dimension. Considering the difficulty of large data sets processing, systematic sampling methods are used to extract representative samples from large data sets firstly. Because transactions with frequent items have smaller Euclidean distances, these samples are used to do SOM cluster analysis. The large data sets are divided into several subsets according to the clustering results. In each subset FP-growth algorithm is executed in parallel, and association rules are mined. The mining result of the subset is combined to get the total association rules. The experiments show that the improved algorithm reduces the memory consumption, shortens the time of data mining, and increases the capacity and efficiency to mass data with a good speedup.

Key words: FP-growth; SOM; data mining; cluster; data partitioning

0 引言

数据挖掘被称为数据集中的知识发现, 是在大量数据集中提取对于决策过程有用的知识的过程。数据挖掘自 20 世纪 90 年代被提出后, 在许多领域得到了很好的应用。关联规则挖掘是数据挖掘的重要组成部分。1993 年, R. Agrawal 等^[1]提出了关联规则的概念及模型, 该模型主要是对一个事物和其他事物相互关联的一种描述。目前, 主要的关联规则挖掘算法有 Apriori 和 FP-growth, 二者都是串行化的算法。Apriori^[2]算法需要多次扫描数据集, 过程中产生了大量候选集, 测试这些候选集需消耗大量时间。FP-growth

算法是一种基于频繁模式树的挖掘算法。该算法可以有效挖掘频繁模式, 并且比 Apriori 算法快大约一个数量级。但是随着数据量的增大和数据集中的有用信息变得越来越稀疏, 在建立 FP-tree 时会消耗大量的内存空间, 以至于内存不够用, 无法完成挖掘任务^[3-4]。Park 等^[5]提出利用系统抽样的方法进行数据挖掘, 然而单纯只依靠抽样的数据进行数据挖掘很容易造成结果的畸形和不准确。因此, 学者们开始考虑通过并行计算环境来解决上述问题。文献[6]采用基于多线程的并行算法, 虽然缓解了存储及计算的压力, 但是内存资源的局限制约了算法的扩展能力; 文献[7-8]中对

收稿日期: 2017-04-17

修回日期: 2017-08-24

网络出版时间: 2017-12-05

基金项目: 国家自然科学基金重点支持项目 (U150120175)

作者简介: 郝奎奎 (1991-), 男, 硕士研究生, 研究方向为大数据技术与数据挖掘; 刘海滨, 研究员, 博士, 国家“千人计划”特聘专家, 研究方向为人工智能和信息技术等。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20171205.1429.076.html>

MPI 并行环境进行了详细的叙述,然而该环境使用进程间通信的方式协调并行计算,导致并行效率较低、内存开销大并且很难解决多节点的扩展性问题。并行算法通常具有较大的进程间的调度和通信开销,并且很难将构建 FP-tree 的任务进行分割。

在前人研究成果的基础上,文中提出了一种改进的 FP-growth 算法。众所周知,数据集中高频率的项集是包含在每条事务中的,那么含有频繁项集的事务之间的欧氏距离总是较小的,所以数据集中的事务会具有聚类现象。利用 SOM 等聚类算法对其做聚类分析,再在每个类别上并行执行数据挖掘算法就可以得到挖掘结果了。首先利用聚类算法对数据集进行分割,然后在每个数据子集上并行执行 FP-growth 算法,这样就将算法所需要的大内存空间分割成小内存进行运算了,并且由于子集是根据聚类结果划分的,所以在各个子集上能够高效地挖掘出有用的关联规则,而非由于随机划分数据集导致挖掘出具有不确定性的规则。由于数据量较大,对大数据集做聚类分析会耗费大量的计算资源,所以文中利用系统抽样方法从大数据集中抽取出具有代表性的样本,然后利用 SOM 算法对样本进行聚类分析,得到分类模型,利用该模型将整个数据集分成若干个子集,并在各个子集上采用 FP-growth 算法挖掘出关联规则,最后将关联规则合并得到总的结果,并通过实验验证了该算法的有效性。

1 SOM 算法相关研究

聚类分析是目前数据挖掘的研究热点之一,聚类分析能够将数据聚类成为多个类或簇,从而使得在同一个类中的数据具有比较高的相似度,而不同类的数据之间有比较大的差别^[9]。聚类分析过程是无监督的分析过程,在对数据进行分类之前,对类别数据是未知的。经典的聚类分析算法有 K-means、CURE 等,这些算法需要提前设定聚类的类别数。这种事前设定类别数的做法大多基于程序设计经验,并且需要经过不断试验才能得到比较正确的分类,这种做法具有一定的不确定性,在一定程度上降低了聚类分析结果的可信度。自组织映射 (self-organizing map) 是由芬兰学者 Kohonen 提出的一种只有两层神经网络的算法。SOM 的权值调整方法与其他神经网络相似,都是采用梯度下降法不断地调整权值。它比较特有的地方就是将高维数据点按照原有的顺序拓扑关联到二维平面网格节点上,这样就实现了高维数据的二维结构可视化。此外,由于 SOM 的高维数据的低维表达能力,SOM 在数据分类、聚类等领域有很多成功的应用案例。文中利用 SOM 聚类算法实现对海量数据的聚类分析,进而利用聚类模型对数据库进行分割,并在各个子集

上进行 FP-growth 数据挖掘。

1.1 SOM 算法描述

SOM 神经网络算法是一种特殊的神经网络,由输入层和输出层构成。在输入层上只存在一个神经元节点,该神经元节点对应于输入向量 $\mathbf{x} = [x_1, x_2, \dots, x_d]$,其中 d 是输入数据维数;在输出层上有一系列的组织在二维平面网格上的神经元节点。每个神经元节点都相应地有一个权矢量 $\mathbf{m} = [m_1, m_2, \dots, m_d]$ 。SOM 神经网络权值的训练步骤如下:

(1) 设定输出层每个节点的初始权重值。通过预先定义一个训练长度或者设定程序终止的误差阈值,来判断训练的终止条件。

(2) 在数据集中选取一个样本 x , 计算样本与每个输出层神经元节点之间的欧氏距离,然后选取与样本 x 距离最近的神经元节点,该神经元节点称为该样本的最佳匹配节点 (best-match unit, BMU), 记为 m_c 。

$$\|x - m_c\| = \min \{ \|x - m_i\| \} \quad (1)$$

(3) 依据预先设定的邻域函数来确定 BMU 邻域的范围,进而利用调节函数调整 BMU 及其邻域内神经元节点的权重值:

$$m_i(t+1) = m_i(t) + a(t)h_{ci}(t)[x(t) - m_i(t)] \quad (2)$$

其中, $m_i(t)$ 为第 t 步节点 i 的权值; $a(t)$ 为第 t 步的学习率; $h_{ci}(t)$ 为邻域函数。

学习率一般伴随着训练的推进而逐渐变小,这里一般选择按线性减小、指数减小的方式。这里的邻域函数通常使用高斯函数或者气泡函数。

(4) 如果还没有达到训练结束的条件,则返回步骤(2)继续训练。

1.2 聚簇分布特征图

人们通常难以从高维数据中区分出数据的分布特征,然而 SOM 算法能够将高维数据拓扑保序地映射到二维平面网格上,映射后的数据结构具有显著的聚类特征,从而能够高效地识别出聚类数据。文中利用 SOM 算法的这一特征来实现大数据集的聚类分析。众所周知,很多算法在进行聚类分析之前往往需要盲目地确定聚类的簇数,尤其随着数据的增大,这种方法往往效果很差。针对这种情况,设计了一种聚类分布特征图,这个特征用来在聚类前先对数据进行预处理,这样就能快速获得数据分布的特点,从而确定数据聚类的类别数目。

通常训练好的 SOM 神经网络的输出层往往组织成一个网络结构,然后对每个神经元节点按列优先进行编号。假定 SOM 输出层共有 $m * n$ 个神经元节点,则定义下面的距离矩阵 (distance-matrix), 简称 D-matrix:

$$\mathbf{d_mat} = \begin{bmatrix} d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{m1} & \cdots & d_{mn} \end{bmatrix} \quad (3)$$

其中, d_{ij} 表示第 i 个神经元节点与第 j 个神经元节点之间的欧氏距离。

接着将距离与一个颜色范围建立一一对应的关系,一般在灰度模式下,距离自小到大,距离所对应的颜色由浅至深,所以就可以得到一个与距离矩阵相对应的颜色矩阵(color-matrix),简称 C-matrix。

$$\mathbf{c_mat} = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mn} \end{bmatrix} \quad (4)$$

其中, c_{ij} 表示与 d_{ij} 相对应的颜色取值。

对每个输出层节点根据颜色矩阵进行灰度值的着色,那么就得到了一张聚类分布特征图,根据聚类分布特征图就可以确定聚类个数。

2 FP-growth 算法研究

2.1 关联规则挖掘的定义

关联规则是对一个事物和其他事物相互关联的一种描述。关联规则的挖掘就是从大量的数据集中找出这些数据之间的相互联系。衡量规则的 2 个度量是支持度和置信度。

定义 1:规则 $A \Rightarrow B$ 在事务集 D 中成立,具有支持度 $S(\text{support})$, S 是 D 中包含 $A \cup B$ 的事务数与所有事务数之比,记为 $\text{support}(A \Rightarrow B)$,它等于事务包含集合 A 和 B 的并的概率 $P(A \cup B)$ 。即

$$\text{support}(A \Rightarrow B) = P(A \cup B) = D(X) / |D| \quad (5)$$

其中, $D(X)$ 是数据集 D 包含 X 的事务数。

定义 2:规则 $A \Rightarrow B$ 在事务集 D 中的置信度 $C(\text{confidence})$ 是指包含 A 和 B 的事务数与包含 A 的事务数之比,它是条件概率 $P(A|B)$ 。即

$$\text{confidence}(A \Rightarrow B) = P(A|B) = \text{support}(A \cup B) / \text{support}(A) \quad (6)$$

关联规则挖掘首先要找出所有满足最小支持度的项集,再计算出置信度大于阈值的所有关联规则。

2.2 FP-growth 算法

FP-growth 算法主要是利用一个树型结构来存储数据项之间的顺序关系,它通常需要扫描两次数据库来构建 FP-tree,第一次扫描获得频繁 1-项集,并筛选出满足支持度大小的频繁项,根据频繁项的支持度计数进行从高到低的排序。第二次扫描数据库就是按照第一次的频繁项排序对原始事务中的数据项重新排序,并将其添加到以“NULL”为根节点的 FP-tree 中。在构建 FP-tree 的过程中,通常会建立一个项头表 T,

表中包括三部分信息,分别是频繁项、对应的支持度计数和指向 FP-tree 节点的指针。根据 FP-tree 提取出条件模式基,满足置信度阈值的条件模式基和后缀就是挖掘出的关联规则。

FP-growth 算法主要由三个模块构成:FP-growth 模块主要是 FP-growth 算法执行的流程;Insert 模块主要完成 FP 树的构建过程;Search 模块用来获取条件模式基集合,该条件模式基集合可以用来确定最后的关联规则。

(1) Insert 模块。

输入:已排序的频繁项集 L_i ,FP(子)树根节点 R_i

输出:FP(子)树 R_i

If $L_i \neq \text{null}$ then

取出 L_i 的第一项 i

If R_i 存在某子节点 $N = i$ then

++ $N.\text{count}$

Else

创建 R_i 的子节点 N ,节点内容为 i

$N.\text{count} = 1$

将 N 加入项头表中

Insert(L_{i-1}, N)

(2) Search 模块。

输入:FP 树 T ,后缀模式 α

输出:频繁项集合 L_S

If T 中只有一个分支 P then

For P 上节点的每个组合 β

$\beta = \alpha \cup \beta$

$L_S = L_S \cup \{\beta\}$

Else

For T 中的每个频繁项 i

构造 β 的条件模式及条件 FP 树 R_β

If $R_\beta \neq \emptyset$ then

Search(R_β, β)

(3) FP-growth 模块。

输入:事务集合 T ,最小支持度 min_sup ,最小置信度 min_conf

输出:强关联规则集合 R_S

扫描 T 找出频繁 1-项集 L

按支持度计数降序排序 L

创建 FP 树的根节点 NULL

For T 中的每个事务 t

找出 t 中的频繁 1-项集合 L_i

L_i 中的项按 L 中的顺序排序

Insert(L_i, NULL)

$L_S = \text{Search}(\text{FP}, \text{NULL})$

在 L_S 中产生强关联规则集合 R_S

接下来用一个简单的例子解释如何构建 FP 树。假设存在事务集合 T ,如表 1 所示。假定 $\text{min_sup} = 20\%$,那么事务集支持度计数 $= 20\% \times 10 = 2$ 次。首先

扫描一遍事务集,统计各类项的支持度计数,去掉支持度计数小于 2 的项。然后将频繁 1-项按照支持度计数从高到低排序,生成排序的频繁 1-项集 $L_1 = [B:8, A:7, C:5, D:2, E:2]$ 。

表 1 事务集合 T

事务	项	事务	项
T_1	A, B, E	T_6	B, C, G
T_2	B, D, F	T_7	A, C
T_3	B, C	T_8	A, B, E
T_4	A, B, D	T_9	A, B, C
T_5	A, B	T_{10}	A, C

根据排好序的频繁 1-项集集合创建项头表,然后再遍历一遍数据库,构建 FP 树。构建过程按照 L_1 中项的顺序创建,将频繁项按照顺序插入到以 NULL 为根节点的树中,随着事务中数据项的插入更新各个树节点的支持度计数。根据 Insert 算法,遍历所有事务之后得出图 1 所示的 FP-tree。故 FP-growth 算法是将事务中所有符合要求的项、项的计数以及项之间的相互关系都压缩到一棵树中。

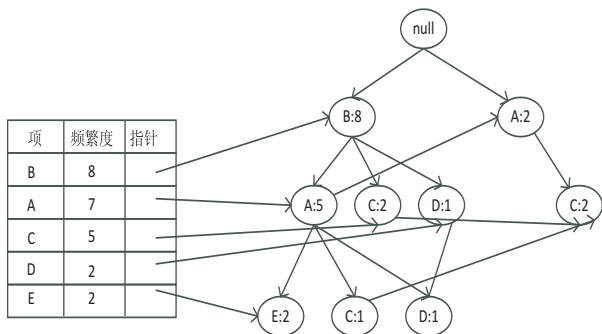


图 1 FP 树与项头表

3 基于 SOM 划分的 FP-growth 算法

3.1 算法过程

FP-growth 通过构造 FP-tree 来寻找频繁模式,然而当数据集达到一定规模时,构造基于内存的 FP-tree 仍然是不现实的。文中提出的基于 SOM 划分的 FP-growth 算法能解决上述问题。该算法利用 SOM 聚类方法得到信息较为聚集的数据子集,在各个子集上并行实施 FP-growth 算法,从而在大型数据集上挖掘出关联规则。该算法总体分成 5 步:数据标准化;按照系统抽样的方法从数据集中抽取样本数据;利用 SOM 算法对样本数据聚类,得到数据集事务的分类模型;将整个数据集的事务按照分类模型分成若干个数据子集,在数据子集上并行执行 FP-growth 挖掘;汇总结果。算法流程如图 2 所示。

(1) 数据标准化。

数据集由多条事务中包含的项都是离散的,为

了便于求解事务之间的欧氏距离,需要把事务中的项改为用数字表示,含有对应的项记为 1,不含的记为 0。具体过程如下:

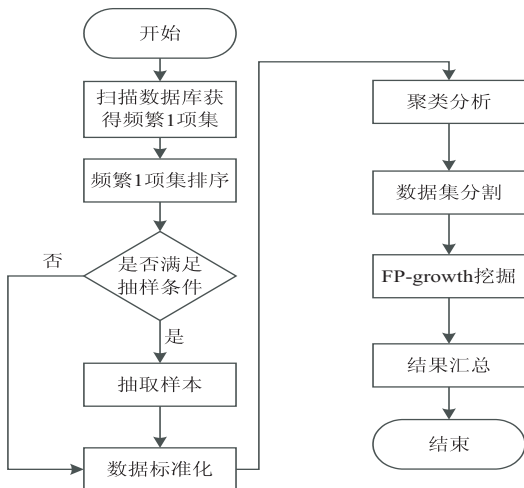


图 2 算法流程

(a) 扫描一遍数据集,求出数据集中大于支持度的频繁 1-项集 L_1 ;

(b) 按支持度大小降序排列 L_1 ,得到新序列 L'_1 ;

(c) 对数据集中的每条事务保留包含于 L'_1 中的项,去除不包含于 L'_1 的项,并按照 L'_1 中的顺序排序;

(d) 建立标准数据向量 $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$,其中 n 的大小等于 L'_1 中元素的个数, x_1, x_2, \dots, x_n 对应的属性为 L'_1 对应位置处的项。例如表 1 中的事务集合 T ,按照 20% 的支持度, $L'_1 = \{B, A, C, D, E\}$,就得到 $T_1 = \{1, 1, 0, 0, 1\}$,以此类推到 T_{10} 。

文中提出的算法先扫描一遍数据集获得排序后的列表 L'_1 ,利用 L'_1 对整个数据集按照上述方法进行数据标准化,在标准化的同时完成系统抽样,最后只需再扫描一次数据集就能完成 FP-tree 的构建了。这是因为在算法开始阶段已经对频繁 1-项集完成排序了,所以整个算法过程只需扫描三次数据集即可完成^[10]。虽然相比于未改进的 FP-growth 多扫描一次数据集,但是未改进的 FP-growth 算法对大数据集进行数据挖掘时很可能因为内存不足使程序无法继续进行^[11]。

(2) 抽取样本。

(a) 将数据库 D 中事务编号,每条事务对应一个唯一的编号;

(b) 将编号按某个间隔分成 k 段,当 N/n (N 表示数据集总的的事务数, n 表示样本容量大小)是整数时, $k = N/n$; 当 N/n 为小数时,从总的事务中去除一些事务,使剩下的事务数目能被 n 整除,这时 $k = N'/n$ (N' 为剩下的事务数);

(c) 在第一段中随机确定一个编号 l ;

(d) 从整体中按照编号 $l, l+k, \dots, l+(n-1)k$ 抽

取出来作为样本。

(3) 聚类分析。

利用 SOM 算法对样本数据进行聚类分析,经过若干次迭代运算,得到自组织神经元间的距离数据,神经元之间距离小的为同一类,距离大的为不同类。神经元在二维平面上的分布情况很容易通过肉眼大概分辨出数据集中存在几类数据。类别个数可以根据需要而定,一般情况下分得精细度越高,数据集中的类别数越多。文中在实验验证阶段,依次提高数据划分的精确度来验证算法的加速程度。确定分类个数后,利用点集最小圆覆盖法^[12]确定每个类别的中心位置,得到中心位置集合:

$$C = \{c_1, c_2, \dots, c_n\} \quad (7)$$

其中, n 为分类的类别数。

C 中任何一个元素 c_k 表示如下:

$$c_k = \{c_{k1}, c_{k2}, \dots, c_{km}\} \quad (8)$$

其中, m 为中心坐标点的维数。

(4) FP-growth 挖掘。

将数据集中的每条数据与 C 中的中心点进行比较,距离最小的中心点所属的类别就是该条数据所属的类别,即:

$$\|t_i - c_{\min}\| = \min\{\|t_i - c_k\|\} \quad (9)$$

其中, t_i 表示任意一条事务数据; c_{\min} 表示与 t_i 最近的中心点; c_k 表示 C 中任意一个中心点。

根据事务 t_i 所属的类别将其分配到对应的计算节点上,这个分配过程在分布式计算 Spark 框架内完成,在 Spark 平台上编写的数据分配程序依据事务数据与中心点的欧氏距离来分配数据所属计算节点,详细的 Spark 平台的搭建和配置过程可参考文献[13]。经过数据分割后,数据集 D 被分割成 s 个子集,表示为 $D = D_1 \cup D_2 \cup \dots \cup D_s$ 。

在每个计算节点上构建 FP-tree,构建过程按照 2.2 节的算法。由于这个阶段之前已经得到频繁 1-项集,并且该项集已按照支持度从大到小排序,所以只需将每条事务中的项添加到 FP-tree 上即可。

(5) 结果汇总。

(a) 在每个计算节点上,根据 2.2 节的 Search() 算法求出所有条件模式基;

(b) 筛选出满足预先设定的支持度大小的条件模式基,得到频繁模式。再对频繁模式中的项集求解置信度,大于置信度阈值的频繁模式即为数据中蕴含的关联规则。支持度和置信度求解过程如下:假设 X 和 Y 是频繁模式中的项集,且 $X \cap Y = \emptyset$,关联规则 $X \Rightarrow Y$ 的支持度 $s(X \Rightarrow Y) = P(X \cup Y) = \text{support}(X \cup Y)$,关联规则 $X \Rightarrow Y$ 的置信度 $c(X \Rightarrow Y) = P(Y | X) =$

$$\frac{\text{support}(X \cup Y)}{\text{support}(X)};$$

(c) 将(b)中得到的关联规则合并就得到了整个数据集中蕴含的关联规则。由于之前做了 SOM 聚类分析,所以每个数据子集上包含有相应类别的高密度频繁项,那么也就包含了对应的关联规则。假设数据集 D 中包含的关联规则集 $R = \{r_1, r_2, \dots, r_n\}$ (n 表示数据集中包含的关联规则个数, r_i 表示任意一条关联规则),任意一个数据子集所包含的关联规则集 $R_{D_i} = \{r_{D_i1}, r_{D_i2}, \dots, r_{D_im}\}$ ($i = 1, 2, \dots, n$, m 表示子集中包含的关联规则的个数, r_{D_ij} 表示子集中任意一条关联规则),则有: $R_{D_1} \cup R_{D_2} \cup \dots \cup R_{D_s} = R = \{r_1, r_2, \dots, r_n\}$ 。

3.2 算法复杂度分析

假设某挖掘任务中有 s 个关联规则、 m 个符合最小支持度的项,事务有 n 个、平均每个事务中包含 a 个符合最小支持度的项。那么经典算法获取 s 个关联规则,忽略连接数据库等开销,算法需要计算的次数为 $m \times n \times a + m \times s$,由于 n 和 m 占主导作用,所以时间复杂度为 $O(mn)$ 。假设抽取样本的个数为 b ,聚类个数为 c ,改进算法的时间复杂度为 $m \times n \times a + m \times s + c \times b$,由于 b 和 c 相较于其他项很小,可以忽略不计,所以时间复杂度也为 $O(mn)$ ^[14-15]。虽然改进算法与原算法的时间复杂度相同,但是由于改进算法将数据库分割成小的子集,大大降低了算法的空间复杂度,减小了计算过程中的内存空间占用量,增大了对海量数据挖掘的可能性。而且在改进算法中各个子集并行进行数据挖掘,也大大缩减了运算时间。

4 实验及结果分析

通过实验对提出的基于 SOM 划分的 FP-tree 算法进行验证。实验中的数据集均来自 Frequent Itemset Mining Dataset Repository。

4.1 SOM 聚类分析结果

实验中使用的是 connect. data 数据集。对该数据集进行抽样,对抽样样本进行 SOM 聚类分析,经过 100 次迭代后,二维平面上的神经元出现了聚簇,相似的神经元相互靠近,不同类的神经元相互远离。每个神经元都与某些输入点有着强连接,并且神经元之间也存在着连接权值,将距离较近的神经元归为一类,它们对应的样本点也属于同一类。

为了凸显聚类效果,在每一个类别上加入一个收缩因子,在收缩因子的作用下样本点向各自的数据中心靠拢。为了更直观地看到分类效果,在每个类别上分别标记不同的图形:三角形、方形、菱形和圆形。经过多次调整参数,得到的聚类效果如图 3 所示。由于这个过程需要比较复杂的编程,所以利用 Java 程序编

程实现聚类的效果显示。

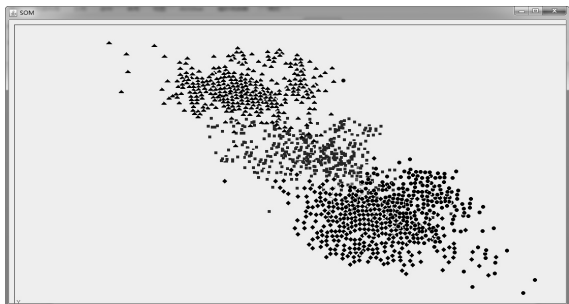


图 3 聚类效果

从图中可以看出,经过 SOM 聚类分析,样本数据分成了 4 类,虽然每个类别有一些数据相互重叠,但是总体上并不影响最后的聚类效果。

4.2 基于 SOM 划分的 FP-growth 算法性能实验

根据 4.1 中的聚类结果,将原数据集分割成 4 个子集,在每个子集上并行执行 FP-growth 算法。算法执行过程中单个计算节点的内存占用情况如图 4(a) 所示。图 4(b) 显示的是未改进的 FP-growth 算法执行过程中的内存占用量。图 4 中显示的是加上计算机操作系统和其他进程所占的内存空间后的内存使用情况,并且是以 60 s 为一个时间片段显示的。通过比较发现,改进算法内存占用量远远低于未改进算法,可以用来对大型数据集进行数据挖掘,而未改进的 FP-growth 在对大型数据集进行挖掘时,很快就会占用大量内存,以至于计算机物理内存空间不够造成数据挖掘任务无法继续进行。

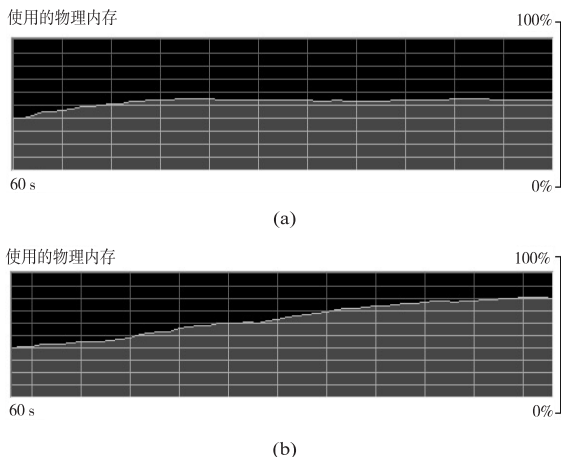


图 4 内存占用量

将改进算法与 Apriori 算法和 FP-growth 算法进行速度上的比较,在支持度为 5% 的情况下,各个算法的运算时间如图 5 所示。

从图中可以看出,随着数据量的增加,三种算法所消耗的时间都在增加,然而 FP-growth 算法的时间消耗明显低于 Apriori 算法,主要因为随着数据量的增大,Apriori 算法会产生大量的候选项,这些候选项的处理耗费了大量的时间。改进的 FP-growth 算法相较于没

有改进的 FP-growth 算法明显降低了算法的运算时间。从图中可以看出,未改进的 FP-growth 算法随着数据量的增加所消耗的时间也在快速增长,而改进的 FP-growth 算法呈现平缓的增长趋势,由此可见改进算法在性能上有明显的提升。

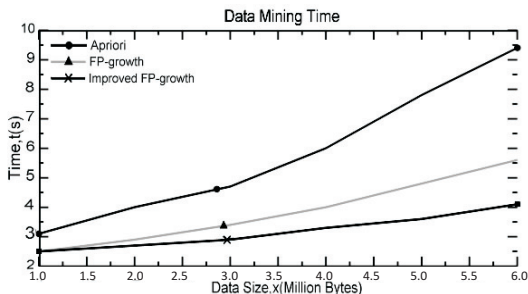


图 5 执行时间对比

5 结束语

主要阐述了对 FP-growth 算法改进的过程。该算法利用 SOM 聚类算法对从大数据集中抽样的样本进行聚类分析,根据聚类结果将大数据集分解成若干个子集,这些子集具有较高密度的关联规则,在各个子集上并行执行 FP-growth 算法就得到了数据集所包含的关联规则。实验结果表明,改进算法降低了内存的占用率,缩短了数据挖掘时间。

参考文献:

- [1] AGRAWAL R, IMIELINSKI T, SWAMI A. Mining association rules between sets of items in large databases[C]//Proceedings of the 1993 ACM-SIGMOD international conference on management of data. New York, NY, USA: ACM, 1993: 207-216.
- [2] AGRAWAL R, SRIKANT R. Fast algorithms for mining association rules[C]//Proceedings of the 1994 international conference on very large data bases. [s. l.]: [s. n.], 1994: 487-499.
- [3] HAN J, PEI J, YIN Y. Mining frequent patterns without candidate generation[C]//Proceedings of 2000 ACM SIGMOD international conference on management of data. New York, NY, USA: ACM, 2000: 1-12.
- [4] 杨 勇, 王 伟. 一种基于 MapReduce 的并行 FP-growth 算法[J]. 重庆邮电大学学报: 自然科学版, 2013, 25(5): 651-657.
- [5] PARK J S, CHEN M, YU P S. Using a hash-based method with transaction trimming and database scan reduction for mining association rules[J]. IEEE Transactions on Knowledge and Data Engineering, 1997, 9(5): 813-825.
- [6] 吴建章, 韩立新, 曾晓勤. 一种基于多核微机的闭频繁项集挖掘算法[J]. 计算机应用与软件, 2013, 30(3): 44-46.
- [7] AOULAD M, LE-KHAC N A, KECHADI T M. Distributed

4 结束语

提出了一种在室内环境中通过分析人体深度图像形态的摔倒检测方法。首先对背景图像和目标图像经过滤波预处理以减少测量噪声对该方法的影响;然后由视差图和最小二乘法获得场景地面信息及地面方程参数,通过分析深度图像中的人体形态信息,比较人体重心高度与高度阈值的结果判定人体是否摔倒。通过设定阈值来进行摔倒检测,当场景地面信息提取后,人体高度信息容易得到,因此该方法简单有效。通过在包含不同摔倒方向的数据上进行摔倒实验,验证了该方法的有效性,为智能摔倒检测的实现和应用提供了一种新途径。

参考文献:

[1] IIO T, SHIOMI M, KAMEI K, et al. Social acceptance by senior citizens and caregivers of a fall detection system using range sensors in a nursing home[J]. *Advanced Robotics*, 2016, 30(3):190-205.

[2] DELAHOZ Y S, LABRADOR M A. Survey on fall detection and fall prevention using wearable and external sensors[J]. *Sensors*, 2014, 14(10):19806-19842.

[3] KOSHMAK G, LINDEN M, LOUTFI A. Challenges and issues in multisensor fusion approach for fall detection; review paper[J]. *Journal of Sensors*, 2015, 5(8):837459.

[4] YANG Lei, REN Yanyun, ZHANG Wenqiang. 3D depth image analysis for indoor fall detection of elderly people[J]. *Digital Communications and Networks*, 2016, 2(1):24-34.

[5] YANG Lei, REN Yanyun, HU Huosheng, et al. New fast fall detection method based on spatio-temporal context tracking of head by using depth images[J]. *Sensors*, 2015, 15(9):23004-23019.

[6] CHENG Juan, CHEN Xiang, SHEN Minfen. A framework for daily activity monitoring and fall detection based on surface electromyography and accelerometer signals[J]. *IEEE Journal of Biomedical & Health Informatics*, 2013, 17(1):38-45.

[7] ZIGEL Y, LITVAK D, GANNOT I. A method for automatic fall detection of elderly people using floor vibrations and sound proof of concept on human mimicking doll falls[J]. *IEEE Transactions on Bio-medical Engineering*, 2009, 56(12):2858-2567.

[8] RIMMINEN H, LINDSTRÖM J, LINNAVUO M, et al. Detection of falls among the elderly by a floor sensor using the electric near field[J]. *IEEE Transactions on Information Technology in Biomedicine*, 2010, 14(6):1475-1476.

[9] SAZONOV E S, BUMPUS T, ZEIGLER S, et al. Classification of plantar pressure and heel acceleration patterns using neural networks[C]//IEEE international joint conference on neural networks. [s. l.]:IEEE, 2005:3007-3010.

[10] FENG W, LIU R, ZHU M. Fall detection for elderly person care in a vision-based home surveillance environment using a monocular camera[J]. *Signal, Image and Video Processing*, 2014, 8(6):1129-1138.

[11] YU M, YU Y, RHUMA A, et al. An online one class support vector machine-based personspecific fall detection system for monitoring an elderly individual in a room environment[J]. *IEEE Journal of Biomedical & Health Informatics*, 2013, 17(6):1002-1014.

[12] CHUA J L, CHANG Y C, LIM W K. A simple vision-based fall detection technique for indoor video surveillance[J]. *Signal, Image and Video Processing*, 2015, 9(3):623-633.

[13] 靳海伟, 彭力, 卢晓龙. 基于轮廓跟踪的摔倒检测算法[J]. *江南大学学报:自然科学版*, 2015, 14(2):172-177.

[14] 罗凯, 金小峰. 基于 Kinect 的跌倒行为识别算法[J]. *延边大学学报:自然科学版*, 2016, 42(2):156-160.

[15] 陈凤东, 洪炳镔. 基于动态阈值背景差分算法的目标检测方法[J]. *哈尔滨工业大学学报*, 2005, 37(7):883-884.

[16] 杨磊, 任衍允, 蔡纪源. 一种基于深度数据的高斯模型运动目标检测方法[J]. *计算机技术与发展*, 2015, 25(9):27-30.

[17] 黄露丹, 严利民. 基于 Kinect 深度数据的人物检测[J]. *计算机技术与发展*, 2013, 23(4):119-121.

(上接第76页)

frequent itemsets mining in heterogeneous platforms[J]. *Journal of Engineering, Computing and Architecture*, 2007, 1(2):1-12.

[8] 邹翔, 张巍, 刘洋, 等. 分布式序列模式发现算法的研究[J]. *软件学报*, 2005, 16(7):1262-1269.

[9] 李文栋. 基于 Spark 的大数据挖掘技术的研究与实现[D]. 济南:山东大学, 2015.

[10] 王乐, 冯林, 王水. 不产生候选项集的 TOP-K 高效用模式挖掘算法[J]. *计算机研究与发展*, 2015, 52(2):445-455.

[11] GOEPPENKAMP MOHAMMED J Z. Advances in frequent

itemset mining implementations: report on FIMI '03[J]. *ACM SIGKDD Explorations Newsletter*, 2004, 6(1):109-117.

[12] 杨中华. 平面点列最小覆盖圆的计算方法[J]. *北京工业大学学报*, 2000, 26(2):96-97.

[13] 毛宇星, 施伯乐. 基于扩展自然序树的概化关联规则增量挖掘方法[J]. *计算机研究与发展*, 2012, 49(3):598-606.

[14] HAN J W, MICHELINE K. 数据挖掘:概念与技术[M]. 范明, 孟小峰, 译. 北京:机械工业出版社, 2012.

[15] 易彤, 徐宝文, 吴方君. 一种基于 FP 树的挖掘关联规则的增量更新算法[J]. *计算机学报*, 2004, 27(5):703-710.