软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(3):839−852 [doi: 10.13328/j.cnki.jos.005454]

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

多维图结构聚类的社交关系挖掘算法*

李振军[1], 代强强[1], 李荣华[1], 毛 睿[1], 乔少杰[2]

[1](深圳大学 计算机与软件学院,广东 深圳 518060)

[2](成都信息工程大学 网络空间安全学院,四川 成都 610225)

通讯作者: 代强强, E-mail: 1134133685@qq.com

摘 要: 社交关系的数据挖掘一直是大图数据研究领域中的热门问题.图聚类算法如 SCAN(structural clustering algorithm for network)虽然可以迅速地从海量图数据中获得关系紧密的社区结构,但这类社区往往只表示了社交对象的聚集,无法反馈对象间的真实社交关系,如家庭成员、同事、同学等.要获取对象间真实的社交关系,需要更多维度地挖掘现实中社交对象间复杂的交互关系.对象间的交互维度很多,例如通话、见面、微信、电子邮件等,而传统 SCAN 等聚类算法仅能够挖掘单维度的交互数据.在研究社交对象间的多维社交关系图数据与传统图结构聚类算法的基础上,提出了一种有效的子空间聚类算法 SCA(subspace cluster algorithm),对多维度下子空间的图结构聚类进行研究,目的是探索如何通过图数据挖掘发现对象间真实的社交关系.SCA 算法遵循自底向上的原则,能够发现社交图数据中所有子空间的聚类集.为提升 SCA 的运行速度,利用其子空间聚类的单调性进行了性能优化,进而提出了剪枝算法 SCA+.最后进行了大规模的性能测试实验以及真实数据的案例研究,其结果验证了算法的效率和效用.

关键词: 图聚类;多维图数据;社交关系;子空间

中图法分类号: TP311

中文引用格式: 李振军,代强强,李荣华,毛睿,乔少杰.多维图结构聚类的社交关系挖掘算法.软件学报,2018,29(3):839−852. http://www.jos.org.cn/1000-9825/5454.htm

英文引用格式: Li ZJ, Dai QQ, Li RH, Mao R, Qiao SJ. Social relationship mining algorithm by multi-dimensional graph structural clustering. Ruan Jian Xue Bao/Journal of Software, 2018,29(3):839−852 (in Chinese). http://www.jos.org.cn/1000-9825/5454.htm

Social Relationship Mining Algorithm by Multi-Dimensional Graph Structural Clustering

LI Zhen-Jun[1], DAI Qiang-Qiang[1], LI Rong-Hua[1], MAO Rui[1], QIAO Shao-Jie[2]

[1](College of Computer Science & Software Engineering, Shenzhen University, Shenzhen 518060, China)

[2](School of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Social relationship mining is a hot topic in the area of massive graph analysis. Graph clustering algorithms such as SCAN (structural clustering algorithm for networks) can quickly discover the communities from the massive graph data. However, relationships in these communities fail to reflect the ‘real’ social information such as family, colleagues and classmates. In reality, social data is very complex, and there are many types of interaction among each individual, such as calling, meeting, chatting in WeChat, and sending emails. However, traditional SCAN algorithm can only handle single dimensional graph data. Based on the study of multidimensional social

* 基金项目: 国家自然科学基金(61402292, 61772091); 国家自然科学基金广东省联合基金(U1301252); 教育部人文社会科学研究规划基金(15YJAZH058)

Foundation item: National Natural Science Foundation of China (61402292, 61772091); National Natural Science Foundation of China Guangdong Joint Fund Project (U1301252); Planning Foundation for Humanities and Social Sciences of Ministry of Education of China (15YJAZH058)

本文由基于图结构的大数据分析与管理技术专刊特约编辑林学民教授、杜小勇教授、李翠平教授推荐.

收稿时间: 2017-08-02; 修改时间: 2017-09-05; 采用时间: 2017-11-07; jos 在线出版时间: 2017-12-05

CNKI 网络优先出版: 2017-12-06 15:37:05, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1536.017.html

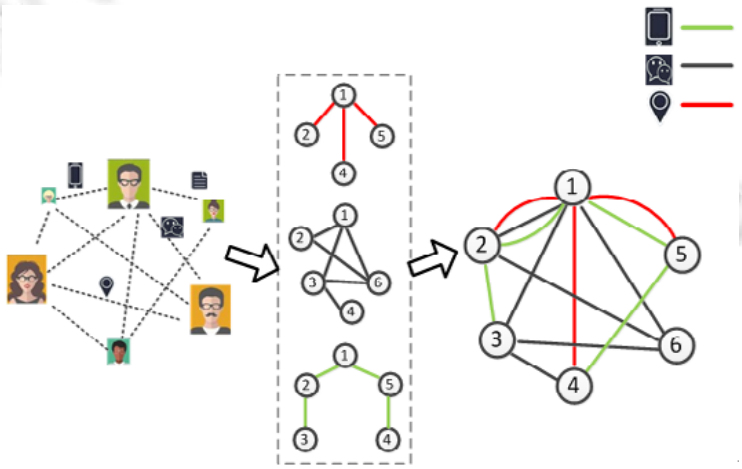
graph data and traditional clustering algorithms, this paper first proposes an efficient subspace clustering algorithm named SCA by mining multi-dimensional clusters in subspaces as a mean to explore real social relationships. SCA follows the bottom-up principle and can discover the set of clusters from the social graph data in all dimensions. To improve the efficiency of SCA, the paper also develops a pruning algorithm called SCA+ based on the monotonicity of subspace clustering. Extensive experiments on several real-world multi-dimensional graph data demonstrate the efficiency and effectiveness of the proposed algorithms.

Key words: graph clustering; multi-dimensional graph data; social relationship; subspace

随着社会快速发展,人们之间的社交关系形成了非常复杂的数据集合,而这些数据集内部的数据相互之间存在着一定的联系,使得每个数据集可以映射为图.与之类似的例子还有很多,如生物中的各种分子之间的关联、交通网络以及天文星体网络等.其中,社区聚类挖掘是近年来研究的热点.传统的社交网络数据挖掘研究内容主要是如何对社交对象间频繁发生某种交互进行聚类,获得的社区仅表示这类交互的一个聚集紧密度,不能表示社交对象间的真实关系.

社交对象间的真实关系往往是最具价值的信息.举例来说,两个频繁交互(如通过手机联系)的对象可能是夫妻或亲友关系,也可能是某种临时性的工作关系(如外卖送餐),而两种关系在对象间的亲疏上代表的意义完全不同,外卖的聚类发现基本不具备应用价值,这就需要分析更多维度的信息以获得对象间的真实关系.

现实中,社交数据往往是巨大的,对象间的联系是多维度的,体现了对象交互的多样性[1,2].例如,人群的社交方式包括电话、见面、即时通信软件(微信、QQ)、E-mail 等.如图 1 所示,图 1(a)为真实世界的社交关系,其中,5 个对象的各类交互用虚线表示.通常传统的聚类算法将各类交互进行特征抽取,分别映射成图数据,使用相应算法进行挖掘,即,将图 1(a)中的交互区分为 3 类,分别是通话(绿线代表)、微信(黑线代表)以及见面(红线代表),映射成为图 1(b)中的 3 个子图后,分别进行单维度的聚类分析.



(a) (b) (c)

Fig.1 Mapping between social relationships and complex graph data

图 1 现实社交关系与复杂关系图数据的映射

通过上述分析得知:仅对某一种特定社交方式的聚类挖掘,反映的是对象在这种交互上的亲疏关系,这种单维度关系分析通常不能显示对象间的真实社会关系[3].要挖掘对象间的真实关系,需要在聚类挖掘中引入更多维度的交互.如图 1(c)所示,将图 1(b)中的 3 个单维度的图数据进行组合,同时在多个维度对图数据进行挖掘将会获得更多价值的信息.例如:仅对对象间的通话进行聚类,得到的聚类反映对象间通话的频率;同理,仅聚类对象间的见面关系,反映的是碰面的频次.以上两种聚类仅能表示对象间的亲疏关系.但如果结合通话与见面两个关系进行挖掘分析,得到聚类(通话与见面都频繁)中社交对象间的关系更紧密,更有可能体现对象间的真实社交关系.

对象间的交互关系在映射的图中表现为不同类型的边,如在图 1(c)中共存在 3 种不同类型的边.本文将图中边类型称为维度,简称维,用 d 表示,如图 1(c)中,d=3,即有 3 个维度.不同维度间的组合称为子空间,子空间共有 2^d–1 种组合,如图 1(c)中的子空间包括通话、见面、微信、通话+见面、通话+微信、见面+微信、通话+见面+微信共 7 种组合.

综上,本文提出了子空间聚类算法 SCA(subspace cluster algorithm),该算法可以对多维度图进行聚类挖掘,算法自底向上发现复杂社交图数据中所有子空间下的聚类集合,探索通过分析对象间多维度交互,来挖掘真实社交关系的方法.

本文的主要贡献总结如下:

(1) 首次在图数据上研究子空间的结构聚类;

(2) 通过各个子空间聚类的数据挖掘,探索真实社交关系朋友圈;

(3) 提出了可对多维度图进行挖掘的子空间聚类算法 SCA;

(4) 自底向上进行多维聚类时,对核心节点、候选子空间进行剪枝优化,提出优化算法 SCA+.

本文第 1 节介绍传统社交图数据挖掘模型和多维聚类的相关工作.第 2 节对 SCA 相关问题进行描述,进一步解释 SCA 模型,同时给出 8 个关键定义与子空间单调性的证明.第 3 节描述 SCA 算法及伪代码执行的过程.第 4 节介绍通过对核节点、候选子空间进行剪枝,给出优化算法 SCA+.第 5 节在 3 个数据集上分别运行 SCA,SCA+,并分析对比算法在各种参数下的时间效率.第 6 节在真实数据集上进行实例分析,验证算法的正确性.最后,在第 7 节总结全文并给出未来的工作.

1 相关工作

社区搜索拥有巨大的潜在价值,产生了众多优秀的聚类算法,包括 k-团[4]、k-*plex*[5]、*LS* 集[6−8]、k-核[9]、k-truss[10,11].在无权图中,k-团表示一个子图,其任意两个顶点之间的最短路径的长度都小于或等于 k 条边.k-*plex* 是由 r 个顶点诱导的子图,其任意顶点的度都至少为 r–k.*LS* 集为顶点集合的子集,该子集具有以下特性:子集中的顶点与其补集中的顶点相连接的边数,少于该子集的任意真子集中的顶点与其相应的补集中的顶点相连的边数.而且,该子集是具备这种特性的最大子集.这里着重讨论两种模型:k-核与 k-truss.

k-核的概念是由 Seidman 首次提出的[9].k-核是一个诱导子图,该子图中节点的度都大于或等于 k,且该子图是具备这种性质的最大子图.为了求解大图数据的 k-核分解问题,Vladimir 和 Matjaz 率先提出了一种线性时间算法[12].该算法依次从图中删除度最小的节点,并利用类似于桶排序的数据结构存储节点,从而实现快速的 k-核计算.该算法首先发现核数较低的节点,然后依次发现核数较高的节点.因为 k-核模型主要关注图中度数较高的节点,往往会忽略一些度数较低、但是却在现实中有关联的社区.

相对于 k-核,k-truss 是较新的概念,由 Cohen 首次提出[10].同样,k-truss 也是一个诱导子图,该子图中的任意一条边都至少包含在(k–2)个三角形中,且该子图是具备这种性质的最大子图.值得注意的是:一个 k-truss 是(k–1)-核,反之不一定成立.由此可见,k-truss 是一种精炼的 k-核结构.然而与 k-核不同的是, k-truss 的定义是基于图中顶点所形成的三角形结构.

另外一个重要的算法是基于结构的聚类算法 SCAN(structural clustering algorithm for network)[13],算法在处理分析现实数据形成的图时取得了很好的效果,并在此基础上提出了一系列剪枝算法,诸如 SCAN++[14],pSCAN[15]等.在大规模社交网络的结构聚类上,也提出了相关算法[16],社交网络数据挖掘研究一直是大数据研究的热门领域.

目前,对多维数据聚类的处理大多采用降维的方式[17,18],即:将多维数据映射到多个单维,将繁多复杂的关系简单化.但降维聚类操作有几个重大的缺陷[19]:对象间多维关系在降维简单化后,往往不代表任何实际意义;在某些情况下,降维后不能直接聚类;多维数据降维后,只能使用 SCAN 等聚类算法针对单维关系数据进行挖掘,目前还没有有效的聚类算法能够对多维数据聚类.

文献[20]提出了一种基于密度连接的子空间聚类,聚类对象主要为空间多维数据,没有涉及结构化数据.

可以看出,当前的研究工作忽视了对结构化数据多维度交互关系的聚类,导致无法辨析社交数据挖掘中聚类结果的价值.本文针对此问题提出的多维度图聚类的社交网络挖掘算法 SCA,通过对多维子空间的聚类得到对象交互更紧密的社区,为挖掘对象间的真实社交关系提供了解决方案.

2 问题描述及相关定义

基于结构的聚类算法在图数据挖掘和分析中具有重要的意义,尤其在社区搜索应用中取得了很好的效果,因此吸引了大量学者在这方面进行相关研究.

2.1 关键定义

存在图 $G=(V,E)$,任意两个顶点 u,v 之间最多存在 d 种不同类型的边(维度),即代表它们之间的交互类型最多为 d 种.设定 $E^d=\{e^1\ldots e^d\}$为图 G 中所有维度的集合,任意子集 $S\subseteq E^d$ 称为子空间(subspace).S 的基数$|S|$称为 S 的度.本文将度值大的子空间称为高维子空间,度值小的称为低维子空间.例如图 1(c)中,$d=3$,$E^3=${通话,见面,微信}.如 $S=${通话,见面},则 S 的度等于 2.ε为实数,μ为整数.

定义 1(顶点结构相似性). 若顶点 u,v 之间存在边 $e^i=(u,v)$,则 u 和 v 在 $e^i(1\leqslant i\leqslant d)$上的结构相似性的值为

$$\sigma^{e^i}(v,u)=\frac{|\Gamma^{e^i}(v)\cap\Gamma^{e^i}(u)|}{\sqrt{|\Gamma^{e^i}(v)||\Gamma^{e^i}(u)|}} \tag{1}$$

其中,$\Gamma^{e^i}(u)$ 和 $\Gamma^{e^i}(v)$ 分别表示 u,v 在 e^i 上的邻居数(含自身).即:u 所有 e^i 的邻居个数为 5,则$|\Gamma^{e^i}(u)|$的值为 6.

由此引申出 u 和 v 在 S 上的结构相似性的值为

$$\sigma^S(u,v)=\{e^i\in S\,|\,\min(\sigma^{e^i})\} \tag{2}$$

即,u,v 在 S 中各个维度下的相似性最小值.如在图 1(c)中,取 $S=${通话,见面,微信},则:

$$\sigma^S(u,v)=\{\min(\sigma^{通话}(u,v),\sigma^{见面}(u,v),\sigma^{微信}(u,v))\}.$$

定义 2(ε邻居). 顶点 u 在 S 下的ε邻居,即 S 下与 u 之间结构相似性大于等于阈值ε的顶点的集合:

$$N_\varepsilon^S(u)=\{w\in\Gamma^S(u)\,|\,\sigma^S(u,w)\geqslant\varepsilon\} \tag{3}$$

定义 3(核心节点). 假设顶点 u 在 S 下的ε邻居的个数大于等于阈值μ,则 u 为核心节点:

$$Core_{\varepsilon,\mu}^S(u)\Leftrightarrow|N_\varepsilon^S(u)|\geqslant\mu \tag{4}$$

定义 4(直接结构可达). 顶点 u 在 S 下直接结构可达顶点 v,当且仅当 u 在 S 下是核心顶点且 u 和 v 之间在 S 下的结构相似性大于等于ε(即,v 是 u 的ε-邻居):

$$DirREACH_{\varepsilon,\mu}^S(u,v)\Leftrightarrow Core_{\varepsilon,\mu}^S(u)\wedge v\in N_\varepsilon^S(u) \tag{5}$$

定义 5(结构可达). 假设在 S 下顶点 $p\in V$ 结构可达顶点 $q\in V$ 当且仅当存在 $p_1,\ldots,p_n$,$p_1=q$,$p_n=p$,并且 p_{i+1} 与 p_i 直接结构可达,即:

$$REACH_{\varepsilon,\mu}^S(q,p)\Leftrightarrow\exists p_1,\ldots,p_n\in V:p_1=q\wedge p_n=p\wedge\forall i\in\{1\ldots n-1\}:DirREACH_{\varepsilon,\mu}^S(p_i,p_{i+1}) \tag{6}$$

定义 6(结构连接). 给顶点 p,q 在 S 下结构连接,当且仅当存在节点 o,p 与 q 在 S 下分别与 o 结构可达:

$$CONNECT_{\varepsilon,\mu}^S(q,p)\Leftrightarrow\exists o\in V:REACH_{\varepsilon,\mu}^S(q,o)\wedge REACH_{\varepsilon,\mu}^S(o,p) \tag{7}$$

定义 7(结构连接集). 设 C 是非空节点集合,如果 C 中的所有节点在 S 下结构连接,则 C 称为 S 下的结构连接集:

$$CONSET_{\varepsilon,\mu}^S(C)\Leftrightarrow\forall p,q\in C:CONNECT_{\varepsilon,\mu}^S(p,q) \tag{8}$$

定义 8(子空间聚类). 给定一个图 G,非空子图 G'在 S 下的子空间聚类当且仅当 G'满足如下两个条件.

(1) G'中每个顶点之间在 S 下满足结构可达;

(2) G'最大.

2.2 子空间聚类的单调性

按照上述定义,最直接地搜索各个子空间上的结构聚类方法就是在所有子空间 S 上使用聚类算法.本文提出的 SCA 就是基于这种思想,采取从低维到高维自下而上的搜索策略挖掘各个子空间上的聚类,探索发现对象间真实的社交关系.

值得注意的是:子空间聚类并不具备单调性,即,低维的聚类在高维不一定是聚类,这是因为部分节点在高维上不满足结构可达;反之,高维的聚类顶点在低维上必定结构可达,但不一定满足最大性原则.

但在子空间聚类过程中,核心节点、直接结构可达、结构可达、结构连接以及结构连接集都具备单调性,即,高维子空间上的核心节点 u 在低维子空间上一定是核心节点,节点 u,v 在高维子空间直接结构可达,在低维子空间同样直接结构可达,以此类推.于是有如下的单调性引理.

引理 1(单调性). 设 ε 为实数,μ 为整数,C 是非空节点集合,即 $C\neq\varnothing,o,q\in V,S\subseteq E^d,T\subseteq S$,有如下的单调性规则.

(1) $Core^S_{\varepsilon,\mu}(o)\Rightarrow Core^T_{\varepsilon,\mu}(o)$;

(2) $DirREACH^S_{\varepsilon,\mu}(o,q)\Rightarrow DirREACH^T_{\varepsilon,\mu}(o,q)$;

(3) $REACH^S_{\varepsilon,\mu}(o,q)\Rightarrow REACH^T_{\varepsilon,\mu}(o,q)$;

(4) $CONNECT^S_{\varepsilon,\mu}(o,q)\Rightarrow CONNECT^T_{\varepsilon,\mu}(o,q)$;

(5) $CONSET^S_{\varepsilon,\mu}(C)\Rightarrow CONSET^T_{\varepsilon,\mu}(C)$.

引理 1 的证明如下.

(1) 核心节点 o 在子空间 S 上为核心节点,则对于子空间 $T\subset S,o$ 在 T 上也是核心节点.这是因为在 S 上所有维的 σ^{e^i} 都大于 μ,T 上所有维的 σ^{e^i} 是 S 上的子集,所以也都大于 μ:

$$Core^S_{\varepsilon,\mu}(o)\Leftrightarrow|N^S_\varepsilon(o)|\geqslant\mu\Leftrightarrow\min_{e^i\in S}(\sigma^{e^i})\geqslant\mu\overset{(T\subset S)}{\Rightarrow}\min_{e^i\in T}(\sigma^{e^i})\geqslant\mu\Leftrightarrow|N^T_\varepsilon(o)|\geqslant\mu\Leftrightarrow Core^T_{\varepsilon,\mu}(o).$$

(2) 节点 o,q 在子空间 S 上直接结构可达,则对于子空间 $T\subset S,o,q$ 在 T 上也直接结构可达.这是因为 o 在 T 上是核心节点,q 在 T 上同时是 o 的 ε-邻居:

$$\begin{aligned}&DirREACH^S_{\varepsilon,\mu}(o,q)\Leftrightarrow Core^S_{\varepsilon,\mu}(o)\wedge q\in N^S_\varepsilon(o)\Leftrightarrow Core^S_{\varepsilon,\mu}(o)\wedge\sigma^S(o,q)\geqslant\varepsilon\Leftrightarrow\\&Core^S_{\varepsilon,\mu}(o)\wedge\min(\sigma^{e^i})\geqslant\mu\overset{T\subset S}{\Rightarrow}Core^T_{\varepsilon,\mu}(o)\wedge\min(\sigma^{e^i})\geqslant\mu\Leftrightarrow\\&Core^T_{\varepsilon,\mu}(o)\wedge\sigma^T(o,q)\geqslant\varepsilon\Leftrightarrow Core^T_{\varepsilon,\mu}(o)\wedge q\in N^T_\varepsilon(o)\Leftrightarrow DirREACH^T_{\varepsilon,\mu}(o,q).\end{aligned}$$

(3) 节点 o,q 在子空间 S 上结构可达,则对于子空间 $T\subset S,o,q$ 在 T 上结构可达.这是因为在 S 上存在 $p_1,\ldots,p_n$,$p_1=q,p_n=p$,并且 p_{i+1} 与 p_i 直接结构可达,根据定义 5 与引理 1 的证明(2),上述条件在 T 上亦成立.

$$\begin{aligned}REACH^S_{\varepsilon,\mu}(o,q)&\underset{T\subset S}{\Leftrightarrow}\exists p_1,\ldots,p_n\in V:p_1=o\wedge p_n=q\wedge\forall i\in\{1,\ldots,n-1\}:REACH^S_{\varepsilon,\mu}(p_i,p_{i+1})\\&\Rightarrow\exists p_1,\ldots,p_n\in V:p_1=o\wedge p_n=q\wedge\forall i\in\{1,\ldots,n-1\}:REACH^T_{\varepsilon,\mu}(p_i,p_{i+1})\\&\Leftrightarrow REACH^T_{\varepsilon,\mu}(o,q).\end{aligned}$$

(4) 节点 o,q 在子空间 S 上结构连接,则对于子空间 $T\subset S,o,q$ 在 T 上结构连接.这是根据在 S 上存在的 x 使得 o,q 分别与之结构可达,则在 T 上,x 与 o,q 分别结构可达,根据定义 6 与引理 1 的证明(3)得证.

$$\begin{aligned}CONNECT^S_{\varepsilon,\mu}(o,q)&\underset{T\subset S}{\Leftrightarrow}\exists x\in V:REACH^S_{\varepsilon,\mu}(x,o)\wedge REACH^S_{\varepsilon,\mu}(x,q)\\&\Rightarrow\exists x\in V:REACH^T_{\varepsilon,\mu}(x,o)\wedge REACH^T_{\varepsilon,\mu}(x,q)\\&\Leftrightarrow CONNECT^T_{\varepsilon,\mu}(o,q).\end{aligned}$$

(5) 设 C 是非空节点集合,并且 C 在子空间 S 上为结构连接集,则对于子空间 $T\subset S,C$ 在 T 上亦为结构连接集.根据定义 7 与引理 1 的证明(4)得证:

$$CONNECT^S_{\varepsilon,\mu}(C)\Leftrightarrow\forall o,q\in C:CONNECT^S_{\varepsilon,\mu}(o,q)\overset{T\subset S}{\Rightarrow}\forall o,q\in C:CONNECT^T_{\varepsilon,\mu}(o,q)\Leftrightarrow CONNECT^T_{\varepsilon,\mu}(C).$$

上述单调性是自底向上的子空间聚类算法剪枝优化的核心思想.对于如何将单调性运用于算法优化,本文将在第 4 节中详细讨论.

3 子空间聚类算法

子空间聚类算法基于结构聚类算法 SCAN,加入维度与子空间的概念,其思想是自底向上、从低维到高维搜索所有子空间的结构聚类.首先介绍算法的基本思想.

3.1 算法的基本思想

SCA 算法的基本思想是自底向上地在各个子空间上使用 SCAN 进行聚类.SCAN 在发现社区的同时,还能够检测到 Hub 节点和离群节点,使得该算法在图聚类中具备特殊的优势.SCAN 在聚类的过程中需要指定两个阈值:判断相连节点之间是否满足结构相似性的阈值ε以及判断节点是否为核心顶点的阈值μ.在聚类过程中,需要对每条边计算其两个端点之间的结构相似性.算法的复杂度是 $O(m^{1.5})$[9],当聚类数据过于庞大时,计算全图节点的结构相似性极为耗时.

SCA 对子空间的图聚类过程如图 2 所示,设聚类社交图数据中包含 3 种类型的边(通信、见面、微信),即,图数据维度为 3.算法需要在 3 个维度上,自底向上分别对 7 个子空间进行运算.即:算法首先进行一维聚类(通话、见面、微信),其次为二维聚类(通话+见面、通话+微信、见面+微信),最后是三维聚类(通话+见面+微信).

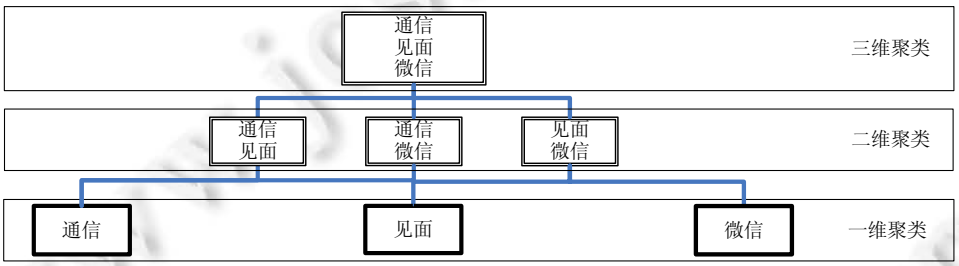


Fig.2 Cluster diagram of SCA

图 2 SCA 算法聚类示意图

3.2 SCA代码分析

算法 1 为 SCA 的伪代码,首先将代码中的部分变量解释如下.

(1) S^i:G 中所有的子空间组合,i=1,2,…,2^d–1;

(2) $Core^{S^i}$:S^i 下所有核心节点;

(3) D_k:k 维度下的所有核心节点集 k=1,2,…,d;

(4) S_k:k 维度下的所有子空间 k=1,2,…,d;

(5) C_{S^i} :S^i 下所有聚类集合.

首先,算法 1 在第 1 次迭代生成一维度下所有子空间的组合 $CanS_1$(第 4 行),如在图 2 中为一维聚类的 3 个子空间:通信、见面、微信.接着,选取 $CanS_k$ 中的所有子空间 S^{can},分别在参数ε,μ下使用定义 1 的结构相似性公式,结合定义 2、定义 3 求出核心节点,并存于 $Core^{S^{scan}}$ 中(第 5 行~第 9 行).算法首先计算子空间内所有节点的ε-邻居数 $N_\varepsilon^{S^{can}}(v)$,如满足 $|N_\varepsilon^{S^{can}}(v)|\geqslant\mu$,则将 v 存放到 $Core^{S^{scan}}$ 中,如计算的是通信子空间,则存于 $Core^{S^{通信}}$ 中.当子空间所有节点ε-邻居计算完成后,调用算法 2($GenerateCluster$($Core^{S^{can}}$,S^{can}))生成一维度上的所有子空间的聚类(第 10 行~第 12 行).对于图 2 来说,就是生成了一维度下 3 个子空间的 3 个聚类集(有可能为空集,即,该子空间不存在聚类).下一步迭代计算高一维空间的聚类(第 13 行),直到完成第 d 维下所有子空间的聚类,在图 2 中,d=3.

算法 1. $SCA(G,\varepsilon,\mu)$.

输入:多维度图 G,参数ε,μ;

输出:子空间聚类集合 C.

1. 初始化集合 C;
2. k=1
3. **while** $k \leqslant d$ **do**
4. 计算所有 k 维度子空间组合 $CanS_k$;
5. **for each** $S^{can} \in CanS_k$ **do** //求每个子空间的核心节点
6. $Core^{S^{can}} = \varnothing$;
7. **for each** $v \in V(S^{can})$ **do** //对属于该子空间的节点进行计算
8. compute $N_\varepsilon^{S^{can}}(v)$;
9. **if** $|N_\varepsilon^{S^{can}}(v)| \geqslant \mu$ **then** $Core^{S^{can}} := Core^{S^{can}} \cup \{v\}$; //将核心节点存放 $Core^{S^{can}}$ 中
10. **if** $Core^{S^{can}} \neq \varnothing$ **then**
11. $C_{S^{can}} = GenerateCluster(Core^{S^{can}}, S^{can})$; //通过 S^{can} 维度的核心节点进行聚类
12. C:=$C \cup C_{S^{can}}$
13. k=k+1

算法 2 是根据核心节点集 $Core^{S^i}$ 在子空间 S^i 下生成聚类的算法,算法首先遍历 $Core^{S^i}$ 中所有的核心节点(第 2 行),如果发现该节点 v 没有被标记,对 v 节点生成一个新的聚类,并进行标记,然后将 $N_\varepsilon^{S^i}(v)$ 中不包括 v 的全部节点加入队列 Q 并标记聚类(第 3 行~第 5 行).算法的第 6 行~第 10 行为生成核心节点 v 的聚类过程,首先从队列中取出 y 节点,假如 y 是核心节点,则继续将 y 的 ε-邻居 $N_\varepsilon^{S^i}(y)$ 中没有被标记的顶点加入队列 Q(第 9 行、第 10 行),重复迭代上述过程,直到队列 Q 为空集,算法返回的 $Cluster^{S^i}$ 为子空间 S^i 的聚类集.

算法 2. *GenerateCluster*($Core^{S^i}$, S^i).

1. $Cluster^{S^i} = \varnothing$;
2. **foreach** $v \in Core^{S^i}$ **do** //遍历所有核心节点
3. **if** v is not classified **then** //对没有标记的核心节点聚类
4. $ClusterID$=$\{v\}$;
5. insert $x \in N_\varepsilon^{S^i}(v)$ and $x \neq v$ into queue Q and classify x; // $N_\varepsilon^{S^i}(v)$ 入队并标记聚类
6. **while** $Q \neq \varnothing$ **do**
7. y=$Q.pop()$;
8. $ClusterID$:=$ClusterID \cup \{y\}$;
9. **if** $y \in Core^i$ **then**
10. insert $z \in N_\varepsilon^{S^i}(y)$ and z is unclassified into queue Q and classify z; //标记节点 z
11. $ClusterID^{S^i} := ClusterID^{S^i} \cup ClusterID$;
12. **return** $ClusterID^{S^i}$;

因需要计算每条边对应节点的结构相似性,故算法 1 的时间复杂度同 SCAN 算法,为 $O(m^{1.5})$[9],证明从略.

4 改进的子空间聚类算法

SCA 算法提出了计算多维度图数据子空间聚类的方法,但是子空间的组合包含(2^d–1)种可能,当 d 很大时,暴力穷举导致算法效率过低.另外,SCA 在每个子空间都要重复计算核心节点,提升了算法时间复杂度.本节通过分析第 2.2 节子空间聚类单调性,提出一种剪枝优化的升级算法 SCA+.

从上述第 2.2 节的单调性证明可得:u 与 v 在子空间 S 上为核心节点/(直接)结构可达/结构相连,必然得到 u 与 v 在子空间 $T \subseteq S$ 上为核心节点/(直接)结构可达/结构相连.如果在子空间 T 不存在聚类,则子空间 S 中也不存在聚类,即,S 不是需要计算聚类的子空间.如图 3 所示:在自底向上计算子空间聚类时,设在一维度的子空间通信

中没有发现聚类,则根据单调性,在所有涉及通信的子空间都不存在聚类,即,子空间通信+见面、通信+微信、通信+见面+微信中都不存在聚类,算法仅需考虑见面、微信、见面+微信这 3 个子空间的聚类计算.

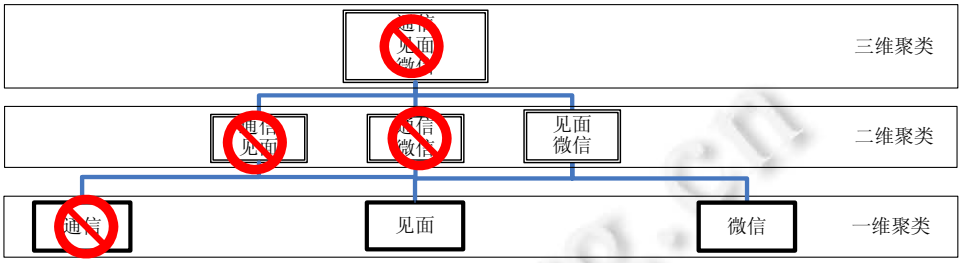


Fig.3 Candidate subspace pruning

图 3 候选子空间剪枝

另外,根据 v 在子空间 S 上为核心节点,在子空间 $T\subseteq S$ 上亦为核心节点的单调性,算法在计算每个子空间的核心节点时,无需对每个节点进行相似性计算,改为仅计算组成 S 的低维子空间核心节点的交集.例如,要计算图 2 中通信+见面子空间的核心节点,按照 SCA,需要遍历所有节点来确定该子空间上的核心节点 $Core^{通信+见面}$.

在 SCA+中,根据核心节点的单调性,仅需遍历子空间通信、见面的核心节点 $Core^{通信}$ 与 $Core^{见面}$ 的交集 $Core^{通信}\cap Core^{见面}$,因为 $Core^{通信+见面}\subseteq(Core^{通信}\cap Core^{见面})$.

算法 3 是对上述两个剪枝(候选子空间、候选核心节点)的实现,首选通过对核心节点的求交集计算,获得候选核心节点 $Core^{S^{scan}}$ (第 1 行~第 3 行);通过 k 维子空间生成(k+1)维候选子空间集,其中,S^{can} 为计算出候选核节点 $Core^{S^{scan}}$ 所在的子空间,然后,将所生成 k+1 维下的候选子空间和候选核节点分别加入集合 $CanS_{k+1}$ 与 $CanD_{k+1}$ 中(第 4 行~第 6 行),算法返回候选子空间 $CanS_{k+1}$ 与候选核心节点 $CanD_{k+1}$.

算法 3. *GenerateCandidate*(D_k,S_k).

1. **for** i from 1 to $|D_k|-1$ **do**
2. **for** j from 2 to $|D_k|$ **do**
3. $Core^{S^{scan}}:=D_k[i]\cap D_k[j]$;
4. **if** $Core^{S^{can}}\neq\varnothing$ and $S^{can}\notin CanS_{k+1}$ and $|S^{can}|=k+1$ **do** //对没有计算过的 k+1 维下子空
5. $CanS_{k+1}:=CanS_{k+1}\cup S^{can}$; //间和该维中的核心节点分别加入
6. $CanD_{k+1}:=CanD_{k+1}\cup Core^{S^{can}}$; //$CanS_{k+1}$ 和 $CanD_{k+1}$ 中
7. **return** ($CanS_{k+1}$,$CanD_{k+1}$);

算法 4 是剪枝算法 SCA+,在一维度子空间聚类的计算上与 SCA 没有很大区别,只是增加了保存一维子空间的核心节点于 D_1 中(第 8 行);在高维度($k\geq2$)的子空间聚类上,SCA+不会对所有子空间进行聚类,而是分别对候选子空间以及候选核心节点进行剪枝优化(第 14 行).另外,在核心节点的查找上,不会对所有的节点进行遍历重新计算核心节点,仅对算法 3 生成的候选核心节点,利用定义 1 引申相似性定义计算相似性,并根据相似性求出ε-邻居,如果ε-邻居大于等于μ则为真核(第 17 行、第 18 行).最后,根据所求的真核 $Core^{S^{scan}}$,使用算法 2 进行聚类(第 19 行~第 22 行),并且保存真核 $Core^{S^{scan}}$ 于 D_{k+1} 中,用于循环迭代(第 20 行).

算法 4. *SCA+*(G,ε,μ) //剪枝算法.

输入:多维度图 G,参数ε,μ;

输出:子空间聚类集合 C.

1. 初始化集合 C,D_1;
2. **for** i form 1 to d do //对各个一维空间聚类
3. $Core^{S^i}=\varnothing$;

4. **for each** $v \notin V(S^i)$ **do**
5. compute $N_\varepsilon^{S^i}(v)$
6. **if** $|N_\varepsilon^{S^i}(v)| \geqslant \mu$ **then** $Core^{S^i} = Core^{S^i} \cup \{v\}$;
7. **if** $Core^{S^i} \neq \varnothing$ **then**
8. D_1:=$D_1 \cup Core^{S^i}$; //每个维度空间的核心节点集合存放于 D_1
9. $C_{S^i} = GenerateCluster(Core^{S^i}, S^i)$;
10. C:=$C \cup C_{S^i}$;
11. k=1
12. **while** $D_k \neq \varnothing$ **do**
13. 初始化 D_{k+1},$CanS_{k+1}$,$CanD_{k+1}$;
14. ($CanS_{k+1}$,$CanD_{k+1}$)=$GenerateCandidate(D_k,S_k)$; //获取候选维度以及该维度上的核心节点
15. **for each** $CanCore^{can} \in CanD_{k+1}$ **do**
16. $Core^{S^{can}} = \varnothing$;
17. **for each** $v \in CanCore^{can}$ **do** //判断该节点在候选维度空间上是否为真正的核心节点
18. **if** $|N_\varepsilon^{S^{can}}(v)| \geqslant \mu$ **then** $Core^{S^{can}} := Core^{S^{can}} \cup \{v\}$;
19. **if** $Core^{S^{can}} := \varnothing$ **then**
20. D_{k+1}:=$D_{k+1} \cup Core^{S^{can}}$
21. $C_{S^{can}} = GenerateCluster(Core^{S^{can}}, S^{can})$;
22. C:=$C \cup C_{S^{can}}$;
23. k=k+1

剪枝算法 SCA+同样需要在一维聚类时对所有边的节点进行结构相似性计算,所以算法的时间复杂度亦为 $O(m^{1.5})$.但 SCA+减少了子空间与核心节点的计算,在一些节点稀疏的多维度图中,可以减少接近 50%的子空间聚类计算.如图 3 所示,因为在计算通信子空间中未发现聚类,所以子空间通信+见面、通信+微信、通信+见面+微信不需要计算,仅需计算剩余的 3 个子空间,减少了 57%的子空间聚类计算;如果在计算图 3 中见面子空间中也未发现聚类,则所有高维度($k \geqslant 2$)的子空间都不存在聚类.

5 实验分析

本节首先介绍实验所用的数据集,然后简要说明实验平台和设置方法,最后给出程序运行结果和详细分析.由于目前学术界还没有能够运行在大规模多维结构图数据上的图聚类算法,因此在本实验中,主要是 SCA 与 SCA+在参数ε,μ下的性能对比.

5.1 实验数据集介绍

本实验将使用 3 个真实的数据集,分别来源于斯坦福大学的官方数据集(http://snap.stanford.edu/data/)和科布伦茨网络收集数据集(http://konect.uni-koblenz.de/).3 个数据都包含多种属性的无向边,即为多维度图数据.数据集的相关信息见表 1.

Table 1 Experimental datasets

表 1 实验数据集

数据集名称	节点数	边数
ego-Facebook	4 039	88 234
Flickr	105 938	2 316 948
Skitter	1 696 415	11 095 298

5.2 实验平台介绍

本文所有程序均用 C++语言实现,实验环境为 Intel(R) Xeon(R) CPU E5-2630 v3@2.40GHz,该服务器具有双 CPU,每个 CPU 为八核,并且支持超线程技术,系统内存为 32GB,系统搭载的操作系统为 Linux(Ubuntu16).

5.3 实验结果与分析

实验共分两部分:第 1 部分(实验(1)),ε,μ不变,在 3 个数据集上分别运行 SCA 与 SCA+,比较两种算法运行时间,结果如图 4 所示;第 2 部分(实验(2)~实验(7)),通过分别改变ε,μ的值,在 3 个数据集上运行 SCA 与 SCA+,通过运行时间比较,分析算法在参数改变下的稳定性,结果如图 5 所示.

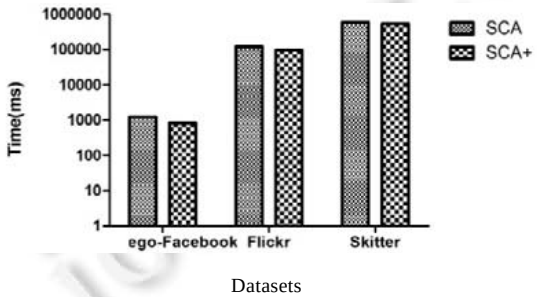


Fig.4 Comparison of operation efficiency between SCA and SCA+ in different datasets

图 4 不同数据集中 SCA 与 SCA+算法效率对比

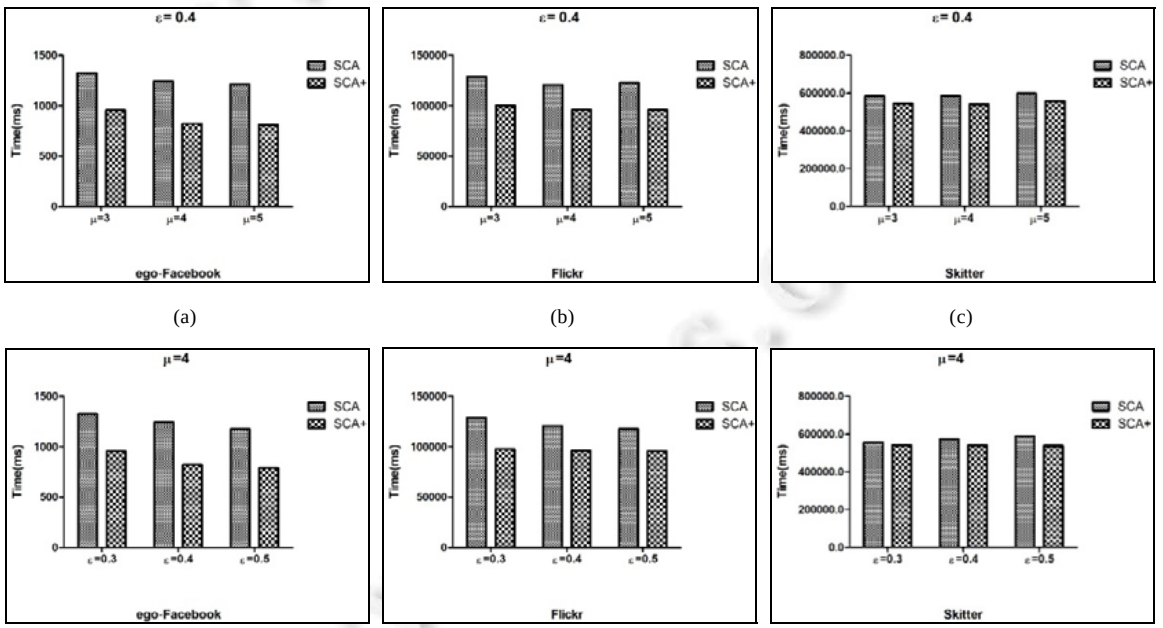


Fig.5 Operational efficiency between SCA and SCA+ when ε=0.4, μ=3, 4, 5 & ε=0.3, 0.4, 0.5, μ=4

图 5 ε=0.4,μ=3,4,5 与ε=0.3,0.4,0.5,μ=4 时,SCA 与 SCA+的算法效率

(1) 分别取ε=0.4,μ=4,在 3 个数据集上分别运行 SCA 与 SCA+算法,分析对比算法的运行时间.从结果来看,ego-Facebook 数据集(8 万条边)运行时间为 SCA(1243ms),SCA+(823ms),共获得各维度聚类 417 个;Flickr(230 万条边)运行时间为 SCA(120575ms),SCA+(96295ms),获得聚类 5 109 个;而 Skitter(1100 万

条边)运行时间为 SCA(586605ms),SCA+(540183ms),共获得聚类 190 080 个.从结果图 4 可以看出:SCA 与 SCA+在各种级别数据集上运行的效率对比,SCA+相对 SCA 有平均 20%的效率提升;

(2) 分别取ε=0.4,μ=3,4,5,在数据集 ego-Facebook 运行 SCA 与 SCA+算法,结果如图 5(a)所示:SCA 分别为 1326ms,1243ms,1216ms;SCA+分别为 917ms,823ms,807ms;

(3) 分别取ε=0.4,μ=3,4,5,在数据集 Flickr 运行 SCA 与 SCA+算法,结果如图 5(b)所示:SCA 分别为 128459ms,120575ms,122893ms;SCA+分别为 100342ms,96295ms,96174ms;

(4) 分别取ε=0.4,μ=3,4,5,在数据集 Skitter 运行 SCA 与 SCA+算法,结果如图 5(c)所示:SCA 分别为 585608ms,586605ms,598531ms;SCA+分别为 545608ms,540183ms,558531ms;

(5) 分别取ε=0.3,0.4,0.5,μ=4,在数据集 ego-Facebook 运行 SCA 与 SCA+算法,结果如图 5(d)所示:SCA 分别为 1325ms,1243ms,1176ms;SCA+分别为 958ms,823ms,798ms;

(6) 分别取ε=0.3,0.4,0.5,μ=4,在数据集 Flickr 运行 SCA 与 SCA+算法,结果如图 5(e)所示:SCA 分别为 128432ms,120575ms,117864ms;SCA+分别为 97986ms,96295ms,95813ms;

(7) 分别取ε=0.3,0.4,0.5,μ=4,在数据集 Skitter 运行 SCA 与 SCA+算法,结果如图 5(f)所示:SCA 分别为 554973ms,586605ms,591069ms;SCA+分别为 541756ms,540183ms,538389ms.

从实验(2)~实验(7)可以看出各参数改变时,SCA 与 SCA+算法的稳定性.

从图 4 结果可以看出:在 ego-Facebook 上,SCA+性能优化得最为明显,这是因为 SCA+相比 SCA 少遍历了约 32%的子空间;在 Skitter 下,SCA+仅提升了 8.6%的效率,这是因为两种算法遍历的子空间数相同(每个子空间都有聚类,没有出现候选子空间的剪枝).SCA+的性能提升来自于减少了核心节点的计算数量,而 Flickr 上 SCA+的性能改善分别来自于候选子空间与候选核心节点的剪枝.

从图 5(a)、图 5(d)可以看出:在保持ε恒定、改变μ值与保持μ恒定、改变ε值的情况下,SCA+与 SCA 相比,效率提升都超过 30%.这是因为在数据稀疏的情况下,某些子空间无法挖掘出社区聚类,候选子空间的剪枝发生概率较大,算法效率提升明显.

对图 5(b)、图 5(e)的结果进行分析可以看出,算法运行保持稳定,不会出现特别异常的聚类结果,也没有出现异常的算法运行时间,算法效率提升介于中间.

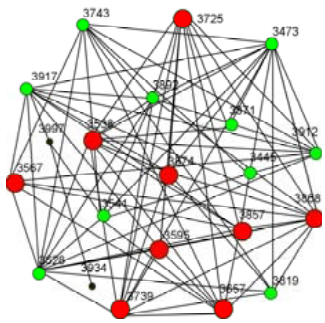
从图 5(c)、图 5(f)可以看出:在保持ε恒定、改变μ值的情况下,算法保持稳定,SCA+与 SCA 相比,效率提升都不超过 10%.这个结果与实验(1)的结果类似,在大数据集上,因为各个子空间都不存在无聚类的情况,所以没有发生候选子空间的剪枝,算法效率提升都集中在候选核心节点的剪枝上.

6 实例分析

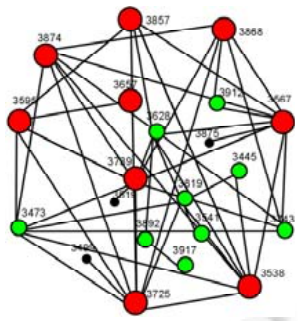
为了验证本文提出算法的有效性与正确性,即,是否能够从多维子空间聚类中获得真实的社交关系,本文与某通信运营商合作,获取部分脱敏数据,运行 SCA+算法聚类后分析本文第 1 作者(节点 3725)以及部分样本节点所在的各子空间聚类的社交圈.

实验数据集为经过运营商脱敏处理的社交数据,数据为文本格式,大小超过 120G,涵盖 9 个月连续的本地交互数据(不含外地交互数据),主要包含 3 个维度的社交关系:通话(移动电话)、见面(处于同一基站扇区)、微信(移动电话号码匹配).根据数据情况,使用 SCA+分别生成 3 个维度下共 7 个子空间的聚类(包含点 3725),如图 6~图 8 所示.图 6~图 8 中,聚类结果中不同维度的聚类用不同颜色的点表示,例如,图 6 为一维聚类的结果,黑色的节点仅在一维聚类中(如节点 3934).绿色的节点不仅出现在一维聚类,还在二维聚类中(如节点 3917,存在图 7(a)).红色的节点存在于一维、二维、三维的聚类(如点 3725,分别存在于图 6(a)、图 7(a)、图 8 中).

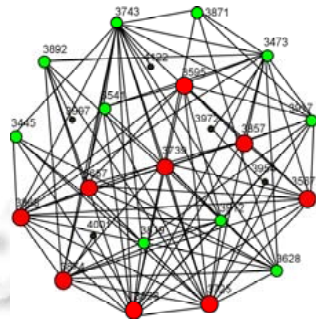
实例分析的结果证明了算法的可行性与正确性:在一维聚类结果中,存在很多与作者(节点 3725)频繁交互但并不熟悉的人,如外卖员、快递员等出现在通话聚类中;二维聚类的人与作者较为紧密,但也存在部分陌生人,如在通话+微信聚类中有网店卖家,作者曾与其多次通话和微信进行联系;三维聚类结果中,都是作者的家人或同事等关系紧密的社交对象.



(a) 通话聚类



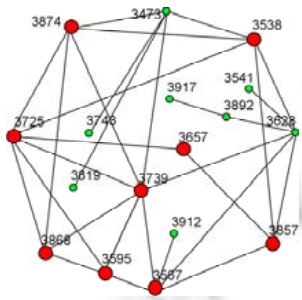
(b) 见面聚类



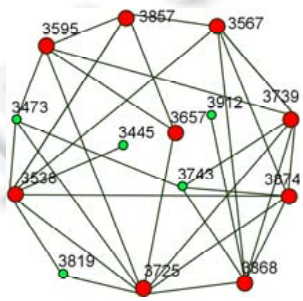
(c) 微信聚类

Fig.6 Clusters in 1st-demention

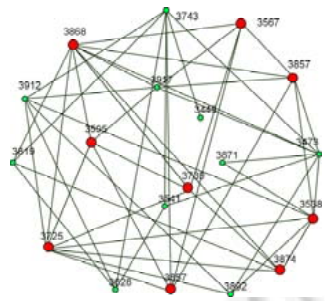
图 6 一维聚类



(a) 微信+见面



(b) 通话+见面



(c) 通话+微信

Fig.7 Clusters in 2nd-demention

图 7 二维聚类

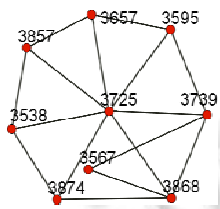


Fig.8 Clusters in 3rd-demention

图 8 三维聚类(通话+见面+微信)

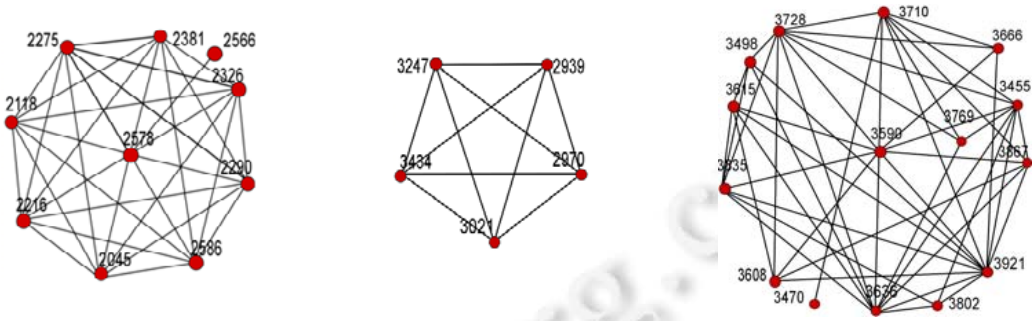
为了更进一步分析、证明算法的正确性,继续选取的几个有代表性的节点(节点 2578,2970,3710)样本进行分析,包含样本节点的三维聚类结果,如图 9(a)~图 9(c)所示.

在图 9(a)中的样本节点 2578 为高中学生,满足三维聚类亲密关系的节点大都为他的同学.可以看到:其他大部分节点都互为三维关系,但点 2566 为 2578 的母亲(单亲家庭),没有与其他节点存在联系.

图 9(b)展现的是样本节点 2970 的三维聚类,2970 为一个初创公司的技术合伙人,三维聚类中,其他 4 个节点都为公司同事,聚类中没有父母、家人等是因为 2970 为单身青年,另外,父母在农村老家,无使用手机习惯.

图 9(c)中的样本节点 3710 为某公司业务员,可以在图中看出:与她进行多维度交互的人员较多,交互对象多为其客户,点 3470 为其配偶,在聚类中只与 3710 发生交互.但是聚类中并无其父母,这是因为实验所用的数据集中无 3710 与其父母在本地见面的联系(父母在外地).据 3710 证实,9 个月内并未和父母在本地见面(实验数据中的见面联系特定发生在本地).

从实例分析结果可以证明算法 SCA 在真实社交关系挖掘中的正确性与可行性.



(a) 点 2578 (b) 点 2970 (c) 点 3710

Fig.9 Clusters of nodes 2578, 2970, 3710 in 3rd-demention

图 9 点 2578,2970,3710 的三维聚类

7 总结及未来的工作

本文首先简述传统的结构聚类算法在发现真实社交关系上的缺陷,提出了在社交关系图数据上多维度挖掘概念.在此基础上,首次提出了基于结构聚类的子空间聚类算法 SCA,算法能够自底向上、低维至高维地搜索出各维度的聚类集.接着分析了子空间聚类的算法单调性并加以证明,在此基础上,通过候选子空间剪枝与候选核心节点剪枝对 SCA 进行性能优化,提出了改进算法 SCA+,实验和实例证明了算法的正确性和可行性以及 SCA+在性能上相对 SCA 的提升与改善.

受限于实验数据的不完备性,部分联系紧密的节点并未在聚类中体现.接下来,我们将使用更大更完备的数据对算法进行测试.为了更深层[21,22]地挖掘社交数据中的真实对象关系,还将引入更多的数据信息对算法进行改进,比如加入见面时间(白天、黑夜)、地点(写字楼、住宅区)等判断因素,可以对社交关系进一步的明确.另外,由于现实世界中所形成的图数据总是在不断发生变化、不断地进行更新,所以有必要对已有的聚类进行动态维护,未来我们也将在这方便展开相关研究.

References:

[1] Ye X, Huang Q, Li W. Integrating big social data. Computing and Modeling for Spatial Social Science, 2017,43(5):377−378. [doi: 10.1080/15230406.2016.1212302]

[2] Peng S, Wang G, Xie D. Social influence analysis in social networking big data: Opportunities and challenges. IEEE Network, 2017,31(1):11−17. [doi: 10.1109/MNET.2016.1500104NM]

[3] Bello-Orgaz G, Jung JJ, Camacho D. Social big data: Recent achievements and new challenges. Information Fusion, 2016,28: 45−59. [doi: 10.1016/j.inffus.2015.08.005]

[4] Mokken RJ. Cliques, clubs and clans. Quality & Quantity, 1979,13:161−173. [doi: 10.1007/BF00139635]

[5] Seidman SB, Foster BL. A graph—Theoretic generalization of the clique concept. Journal of Mathematical Sociology, 1978,6: 139−154. [doi: 10.1080/0022250X.1978.9989883]

[6] Borgatti SP, Everett MG, Shirey PR. LS sets, lambda sets and other cohesive subsets. Social Networks, 1990,12:337−357. [doi: 10.1016/0378-8733(90)90014-Z]

[7] Seidman SB. Internal cohesion of LS sets in graphs. Social Networks, 1983,5:97−107. [doi: 10.1016/0378-8733(83)90020-5]

[8] Seidman SB. LS sets as cohesive subsets of graphs and hypergraphs. Mathematical Social Sciences, 1983,6:87−91. [doi: 10.1016/0165-4896(83)90048-3]

[9] Seidman SB. Network structure and minimum degree. Social Networks, 1983,5:269−287. [doi: 10.1016/0378-8733(83)90028-X]

[10] Cohen JD. Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report, 2008.

[11] Wang J, Cheng J. Truss decomposition in massive networks. Proc. of the VLDB Endowment (PVLDB), 2012,5:812−823. [doi: 10.14778/2311906.2311909]

[12] Batagelj V, Zaversnik M. An *O*(*m*) algorithm for cores decomposition of networks. Computer Science, 2003,1(6):34−37.

[13] Xu X, Yuruk N, Feng Z, Schweiger TAJ. SCAN: A structural clustering algorithm for networks. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. ACM Press, 2007. 824−833. [doi: 10.1145/1281192.1281280]

[14] Shiokawa H, Fujiwara Y, Onizuka M. SCAN++: Efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. VLDB Endowment, 2015,8(11):1178−1193. [doi: 10.14778/2809974.2809980]

[15] Chang L, Li W, Lin X, Qin L, Zhang W. pSCAN: Fast and exact structural graph clustering. IEEE Trans. on Knowledge & Data Engineering, 2017,29(2):387−401. [doi: 10.1109/TKDE.2016.2618795]

[16] Chen JM, Chen JJ, Liu J, Huang YL, Wang Y, Feng X. Clustering algorithm for large scale social network based on structure similarity. Journal of Electronics and Information, 2015,37(2):449−454 (in Chinese with English abstract). [doi: 10.11999/JEIT 140512]

[17] Chen G, Huang RZ, Zhong WL. Multi-Dimensional text representation based on social features. Computer Engineering and Science, 2016,38(11):2348−2355 (in Chinese with English abstract). [doi: 10.3969/j.issn.1007-130X.2016.11.029]

[18] Lu J, Peng X, Deng W, Mian A. Regularization techniques for high-dimensional data analysis. Image & Vision Computing, 2017,60: 1–3. [doi: 10.1016/j.imavis.2017.03.001]

[19] He L, Cai YC, Yang Z. Survey of clustering algorithms for high-dimensional data. Application Research of Computers, 2010,27(1): 23−26 (in Chinese with English abstract). [doi: 10.3969/j.issn.1001-3695.2010.01.006]

[20] Kröger P, Kriegel HP, Kailing K. Density-Connected subspace clustering for high-dimensional data. In: Proc. of the Siam Int'l Conf. on Data Mining. 2004. 246−257. [doi: 10.1137/1.9781611972740.23]

[21] Andris C. Integrating social network data into GISystems. Int'l Journal of Geographical Information Science, 2016,30(10):1−23. [doi: 10.1080/13658816.2016.1153103]

[22] Gao Q, Zhang FL, Wang RJ, Zhou F. Track data: A survey of key technologies in data processing. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):959−992 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5143.htm[doi: 10.13328/j.cnki.jos.005143]

附中文参考文献:

[16] 陈季梦,陈佳俊,刘杰,黄亚楼,王嫄,冯霞.基于结构相似度的大规模社交网络聚类算法.电子与信息学报,2015,37(2):449−454. [doi: 10.11999/JEIT140512]

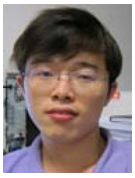
[17] 陈功,黄瑞章,钟文良.基于社交特征的多维度文本表示方法.计算机工程与科学,2016,38(11):2348−2355. [doi: 10.3969/j.issn.1007-130X.2016.11.029]

[19] 贺玲,蔡益朝,杨征.高维数据聚类方法综述.计算机应用研究,2010,27(1):23−26. [doi: 10.3969/j.issn.1001-3695.2010.01.006]

[22] 高强,张凤荔,王瑞锦,周帆.轨迹大数据:数据处理关键技术研究综述.软件学报,2017,28(4):959−992. http://www.jos.org.cn/1000-9825/5143.htm [doi: 10.13328/j.cnki.jos.005143]

李振军(1979－),男,广西柳州人,博士,工程师,主要研究领域为数据挖掘,深度学习.

代强强(1992－),男,硕士,主要研究领域为社交数据挖掘,图算法.



李荣华(1985－),男,博士,讲师,主要研究领域为图数据挖掘,社交网络分析.

毛睿(1975－),男,博士,教授,CCF 高级会员,主要研究领域为数据挖掘,数据库,统计方法,机器学习,计算生物.

乔少杰(1981－),男,博士,教授,CCF 高级会员,主要研究领域为轨迹数据挖掘,机器学习.