软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(3):650−668 [doi: 10.13328/j.cnki.jos.006173]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

时序图节点嵌入策略的研究*

吴安彪[1], 袁 野[2], 马玉亮[3], 王国仁[2]

[1](东北大学 计算机科学与工程学院,辽宁 沈阳 110169)

[2](北京理工大学 计算机学院,北京 100081)

[3](东北大学 工商管理学院,辽宁 沈阳 110169)

通讯作者: 袁野, E-mail: yuanye@mail.neu.edu.cn

摘 要: 相较于传统的图数据分析方法,图嵌入算法是一种面向图节点的新型图数据分析策略.其旨在通过将图节点向量化表达,进而在节点向量基础上,利用神经网络相关技术,更有效地进行图数据分析或挖掘工作,如在节点分类、链接预测及交通流预测等经典问题上效果显著.虽然研究者们在图嵌入方面已取得了诸多成果,但是面向时序图的节点嵌入问题却未被充分重视.在先前研究工作的基础上,结合信息在时序图中的传播特性,提出一种对时序图节点进行自适应嵌入表达的方法 ATGEB(adaptive temporal graph embedding).首先,为了解决不同类型时序图节点活跃程度不同的问题,通过设计一种自适应方式对其活跃时刻进行聚类;而后,在此基础上设计一种游走模型,用以保存节点对之间的时间关系,并将节点游走序列保存在双向多叉树上,进而可以更快速地得到节点时间相关的游走序列;最后,在基于节点游走特性和图拓扑结构的基础上对节点向量进行重要节点采样,以便在尽可能短的时间内训练出满足需求的网络模型.通过充分的实验证明:面向时序图的嵌入策略相较于现流行的嵌入方法,在时序图时序中节点间时序可达性检测以及节点分类等问题上得出了更好的实验效果.

关键词: 时序图;节点嵌入;重要采样;时序可达;节点分类

中图法分类号: TP311

中文引用格式: 吴安彪,袁野,马玉亮,王国仁.时序图节点嵌入策略的研究.软件学报,2021,32(3):650−668. http://www.jos.org.cn/1000-9825/6173.htm

英文引用格式: Wu AB, Yuan Y, Ma YL, Wang GR. Node embedding research over temporal graph. Ruan Jian Xue Bao/Journal of Software, 2021,32(3):650−668 (in Chinese). http://www.jos.org.cn/1000-9825/6173.htm

Node Embedding Research over Temporal Graph

WU An-Biao[1], YUAN Ye[2], MA Yu-Liang[3], WANG Guo-Ren[2]

[1](School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China)

[2](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

[3](School of Business Administration, Northeastern University, Shenyang 110169, China)

Abstract: Compared with the traditional graph data analysis method, graph embedding algorithm provides a new graph data analysis strategy. It aims to encoder graph nodes into vectors to perform graph data analysis or mining tasks more effectively by using neural network related technologies. And some classic tasks have been improved significantly by graph embedding methods, such as node classification, link prediction, and traffic flow prediction. Although plenty of works have been proposed by former researchers in graph

* 基金项目: 国家自然科学基金(61932004, 62002054, 61732003, 61729201); 中央高校基本科研基金(N181605012); 中国博士后科学基金(2020M670780)

Foundation item: National Natural Science Foundation of China (61932004, 62002054, 61732003, 61729201); Research Funds for the Central Universities (N181605012); China Postdoctoral Science Foundation Funded Project (2020M670780)

本文由“支撑人工智能的数据管理与分析技术”专刊特约编辑陈雷教授、王宏志教授、童咏昕教授、高宏教授推荐.

收稿时间: 2020-07-19; 修改时间: 2020-09-03; 采用时间: 2020-11-06; jos 在线出版时间: 2021-01-21

embedding, the nodes embedding problem over temporal graph has been seldom studied. This study proposed an adaptive temporal graph embedding, ATGED, attempting to encoder temporal graph nodes into vectors by combining previous research works and the information propagation characteristics together. First, an adaptive cluster method is proposed by solving the situation that nodes active frequency is different in different types of graph. Then, a new node walk strategy is designed in order to store the time sequence between nodes, and also the walking list will be stored in bidirectional multi-tree in walking process to get complete walking lists fast. Last, based on the basic walking characteristics and graph topology, an important node sampling strategy is proposed to train the satisfied neural network as soon as possible. Sufficient experiments demonstrate that the proposed method surpasses existing embedding methods in terms of node clustering, reachability prediction, and node classification in temporal graphs.

Key words: temporal graph; node embedding; importance sampling; temporal reachability; node classification

基于计算机超高速的运算能力,人们设计出了面向高维数据、多层次的神经网络的模型.在深度学习领域中的图数据方向,由于图数据结构的非欧式属性,在神经网络研究初期,人们并没有找到一种针对图数据行之有效的网络模型.所以,如何将高维度的图数据结构抽象成低维度、结构性的向量,然后通过神经网络对得出的节点向量进行参数训练,是一件非常有挑战性的工作.

2009 年,Scarselli 等人[1]首次提出了图神经网络模型(graph neural network model,简称 GNN)的概念,并且文中的基本也是通过整合邻居节点的属性来进行对节点的向量化表达,给出了一种图表达学习的研究雏形,一定程度上指导了后来一些研究者们在基于属性图表达学习方向的研究思路,但是在当时并未引起学者们的广泛关注.直到 2014 年,随着 DeepWalk[2]算法的提出,真正引爆了人们对图(节点)嵌入的研究热潮.受启发于自然语言处理中的 word2vec[3]算法,DeepWalk 算法设计出了一种非常简单且有效的面向节点的向量表达方式,即通过 *skip-gram*[3]模型训练各个节点随机游走出的序列(节点 ID 序列),进行计算节点间的相似性和节点向量.而后在基于 DeepWalk 算法的基础上,研究者们陆续提出了 LINE[4],PTE[5],node2vec[6]以及 stru2vec[7]等节点嵌入算法,但是这些算法都没有脱离 DeepWalk 算法的原有框架,本质上说,相比较于 DeepWalk 算法,这些算法只是在节点游走策略上更倾向于节点游走的有偏性,即在关注节点拓扑结构的情况下定义出节点的游走概率.在 2018 年,Dong 等人[8]给出了 DeepWalk,LINE,PTE 和 node2vec 这 4 种算法的矩阵表达形式,更进一步说明了这些算法在思想上的统一性.

虽然这种基于游走策略的图表达学习在一定程度上可以保留节点拓扑结构这一属性,特别是基于有偏游走策略的方法如 node2vec 等,并且在实验中取得了令人满意的结果,但是基于游走策略的算法同样也有一个非常明显的缺点,即完全忽略了节点间的属性值对实验结果的作用,这就使得在某些属性图的数据集上无法达到令人满意的效果.在 2017 年,Hamilton 等人[9]设计了一种名为 GraphSAGE 的新型采样策略,首先,节点 v 对邻居节点 v_i(节点 v 的所采样的第 i 个邻居)进行采样;而后对采样后的节点 v_i 再次向下一层节点 v_{ij}(节点 v_i 所采样的的第 j 个邻居)进行采样;然后由外向内对所采样的节点特性信息进行聚合,从而得到节点 v 的一个新的聚合后的特征向量.GraphSAGE 算法虽然考虑了节点特征向量这一因素,但在一定程度上弱化了对节点拓扑结构属性的保留.同年,Kipf[10]提出了一种半监督分类的图卷积网络(graph convolutional networks,简称 GCN).与 GraphSAGE 的策略有所不同,文献[10]中通过共享权值对单一节点所有邻居节点特征进行卷积操作,本质可以看成加权求和,并且由公式 $H^{(l+1)}=\sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)})$ 可以看出:作者通过由单位矩阵 I_N 和邻接矩阵 A 相加得到的矩阵 $\tilde{A}$、节点特征向量 H 以及矩阵 $\tilde{D}=\sum_j \tilde{A}_{i,j}$,在进行拉普拉斯变换后,借助于可训练权重参数 W 来得出新的节点向量.并且可以看出,作者进行了一个多层卷积的操作.同时,从文献[10]中所提供的实验结果可以看出:随着层数的增加,会得出精度更高的实验结果;但是一旦超过一定的层数,实验结果会急剧下降.这是因为一旦单一节点所聚合的信息过多,就不能很好地保持各个节点间向量的差异性,从而导致模型泛化.文献[10]中对邻居节点的卷积操作是在共享权值的前提下统一进行的,但在现实情况中,节点的属性受某些特定邻居节点的影响会大于其余节点,所以这些邻居节点理所应该享有更高的权重.基于此,Velickovic 等人[11]在 2018 年提出了图注意力机制网络(graph attention networks,简称 GAT),即在共享权重参数 W 和节点特征向量 $\vec{h}_i$ 的基础上,通过 LeakyReLU 和 softmax 函数来计算两节点 v_i 和 v_j 间的权重系数:

$$\alpha_{i,j}=\frac{\exp(LeakyReLU(\vec{a}^T[W\vec{h}_i \| W\vec{h}_j]))}{\sum_{k\in N_i}\exp(LeakyReLU(\vec{a}^T[W\vec{h}_i \| W\vec{h}_j]))}.$$

由于各个邻居节点特征向量的不同,所以权重系数也会不同,且共享权重参数 W 在学习的过程中不断变化,那么节点间的权重系数也会随着训练的过程而改变.

时序图(temporal graph,亦被称为 temporal network[12,13],time-varying graph[14,15])是一种节点边上带有时间标签的基于时间的动态图,对时序图数据的分析工作在生物信息网络、在线社交网络和道路交通网络等有着重要的应用价值.在生物信息网络中,生物功能的连接并不是一直活跃的[16],如在蛋白质互作用网络[17]和基因调控网络中[18],其中的生物结构体的联系是存在一定的先后顺序的,而通过分析这些结构体在不同时间段相互关系,可以更容易确定出它们在网络中的功能.在道路交通网络[19−21]中,可以结合交通网络中的历史数据,向用户进行某一时刻的路径推荐或可达性查询等相关工作;在社交网络中[22−24],可以通过记录用户间的具体互动,更精准地刻画用户间的关系.

现有的关于节点嵌入的研究工作更多的是在关注如何更好地在节点表达向量中保存节点的结构属性,但是在时序图中,节点间的拓扑结构是随时间动态变化的,因为节点之间的联系是时序性的,表明了某种特定信息在节点间传播的先后顺序,而这种情况尚未被研究者们充分地考虑在图嵌入策略中.

在网络图里,时序性不仅仅只限于两相连节点间的时序性,如图 1(a)中的顶点 1 和顶点 2 的连接关系只在时刻 t_1 和 t_2 是存在的,且当不考虑边静态(1,3)时,只是单纯地从拓扑结构上看,在顶点 1 和 2 之间是存在可达路径 1→2→3 的;但是一旦考虑时间因素,如果在静态边(2,3)上的时间戳为 t_3 且 $t_3>t_2>t_1$,则顶点 1 和顶点 2 之间便不再存在可达路径.除了连接时序性和路径时序性以外,节点的属性有时也会是时序性的,以图 1(b)中的顶点 1 为例,节点在不同的时刻可能会有不同的属性.这一现象在电商网络中是非常明显的,在推荐系统领域中是一个非常棘手且待解决的挑战.

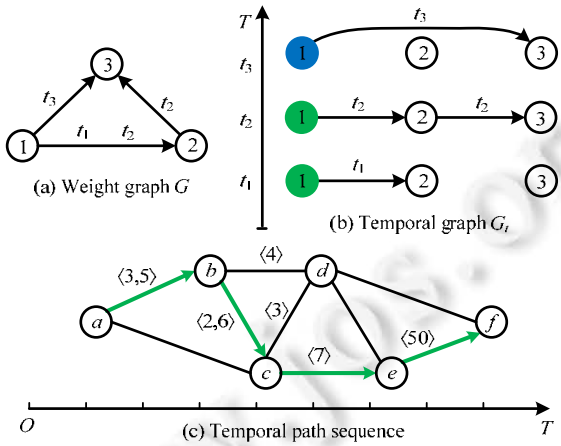


Fig.1 Instance of temporal graph

图 1 时序图示例

基于时序图相比较于静态图所特有的时间属性,面向时序图的图神经网络模型可总结有如下挑战.

(1) 节点可达性:如果两个顶点 v_i,v_j 仅仅在拓扑结构上存在一条可达路径,但是在实际情况下并没有任何想连接的可能,则节点 v_i 在采样时去整合顶点 v_j 的信息是不合适的;

(2) 多边性:两顶点在一个时间跨度之下不同的时刻会存在多条时间相关的边,如图 1(c)中的顶点 a 和顶点 b 分别在时刻 3 和时刻 5 存在两条边,那么在节点做游走或信息整合时,需要考虑选择哪条边更为合适?

(3) 路径选择:选择不同基于时间的路径则会直接影响起始点游走长度或聚合节点信息的多少.如图 1(c)

中的顶点 b 和顶点 d,如果在顶点 b 到顶点 c 时选择时刻 6 的路径,则顶点 b 无法到达顶点 d;而选择时刻 2 的路径时则可达.所以,准确地选择一条尽可能正确的路径,才可以整合到尽可能准确的信息;

(4) 路径时间跨度:当一个顶点沿着一条路径进行游走或者进行信息聚合时,如果这条路径的时间跨度过大,那么在靠近路径顶端的顶点和靠近路径起始位置的顶点应该是不能够享有相同的权重系数的.如图 1(c)中由顶点 a 到顶点 f 的一条路径($\langle a,b,c,e,f\rangle$),可以看出:当在路径的 a 和 e 之间,时间跨度只有 7;但是当到达顶点 f 时,时间跨度陡增为 50,此时可能顶点 f 对顶点 a 的影响已经微乎其微了,但是如果把顶点 f 的信息整合都顶点 a 中去,则不仅使顶点 a 整合后的信息显得冗余甚至是错误的.所以,随着时间的增长,应该弱化联系时间更长的顶点对起始点的影响.

虽然现如今在图嵌入领域已经出现众多研究成果,但是当对这些研究工作的实验结果进行分析时,发现了一个非常明显且普遍的问题,即,基于不同策略的图表达学习在不同类型图数据上的实验结果是存在差异性的.这是因为有些图数据对其本身的拓扑结构具有很强的敏感性,而对其自身的属性等因素并不敏感.对于这种类型的图数据,使用游走策略的节点嵌入方法往往可以得到更好的实验结果;而对其自身属性等因素较为敏感的图数据,显然更适用于卷积策略的图表达策略.基于此发现,可以认识到:在现有的理论框架和技术水平之下,想要通过一种统一的图学习模型来进行对所有类型图数据进行图表达的尝试是几乎不可行的.所以,为了应对面向时序图表达学中的挑战,本文旨在设计出一种对自身时序性敏感的图数据嵌入学习方法,以得到一种更符合时序图特性的图表达学习方式.

本文创新点如下:

(1) 整合现有的图嵌入思想和时序图相关特性,设计出一种新型的、可以满足对时序图数据进行分析的时序图嵌入策略;

(2) 为解决不同类型时序图节点间活跃性差异巨大的问题,设计出一种自适应、满足时序可达性的节点游走模型,尽可能地保留在不同的时间段节点和其周围节点联系的时间性质;

(3) 为了尽可能快速地得到节点在不同时间段所游走出的节点序列,通过将在游走过程中的节点存在双向、时序多叉树中.如此,在游走结束后,可以简单且快速地得到节点的游走序列;

(4) 在嵌入方法特性和图拓扑结构的基础上,尝试通过提出重要节点采样的方式来缩减面向单节点的神经网络模型的训练时间;

(5) 在不同类型的真实时序图数据集上进行了面向时序图不同任务的实验,以此验证了本文中所设计方法的通用性、准确性以及高效性.

本文第 1 节对时序图相关的基本定义和时序图嵌入问题定义进行说明.第 2 节给出一种基本的、时序性的游走方法.第 3 节提出一种更有效的面向时序图的游走策略.在第 4 节中进行重要节点采样的工作.第 5 节中对在时序图上的各项实验分析进行说明.第 6 节中对相关工作进行说明.

1 问题定义

本节将对一些基本的问题进行梳理,对所面向的研究对象时序图的类型进行介绍,对基本的概念做出定义.并在表 1 中对本文所常用到的一些符号的意义进行简要说明.

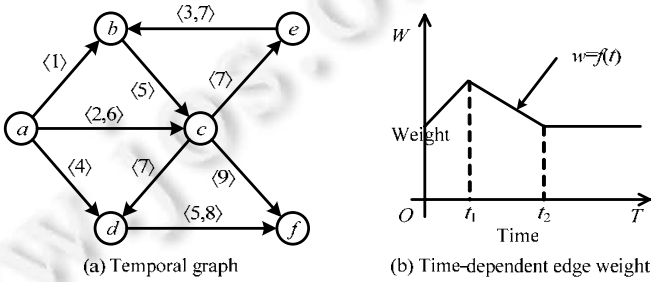
Table 1 Labels and meanings

表 1 常用符号表示

名称	描述	名称	描述
u,v	图中节点	$Arr_{(u,v)}$	节点 u 到节点 v 的时间
G_T	时序图网络	ul	节点 u 的游走序列
Inf_i	网络中的信息类型	WL	所有节点游走序列集合
Lab_i	节点的标签	Win_u	节点 u 在游走序列中的窗口
N_u	节点 u 的邻居节点集合	z_i	节点 v_i 的向量表达
$T_{(u,v)}$	节点 u 和 v 的联系时刻集合	R^d	向量维度空间

一般情况下,时序图边都是离散型表达,如图 2(a),节点 u 在时刻 t 指向节点 v 的边表示为(u,v,t,λ),其中,λ表示到达时间,即节点 u 在 t 时刻出发经λ时间达到节点 v.在本文中,由于并没有涉及到节点间基于时间的路径查询等问题[25],更注重的是节点u到达节点v的时刻,从而无需刻意考虑λ,所以在实际操作中,可以将$t=\lambda+t$来看待.如此,在本文中的时序图便可以简化为(u,v,t)的形式,其中,$t=\lambda+t$.以图 2(a)中顶点 a 和顶点 b 为例,假如在 0 时刻由 a 向 b 发出信息,经历λ=1 个时间单位到达,则其边上权重为 1.而在社交网络数据集中,由于消息的即时性往往将λ=0 做处理.而对于另外一种情况,如果λ表示联系持续时间,即在 t 时刻两节点建立联系,持续λ个时间单位,基于最早到达的原则,将λ忽略处理,两节点边上的权重赋值为 t.需要注意的是:在真实地数据集上,数据表示形式中的两节点的联系往往都是即时性的,所以在λ这个层面上无需做过多考虑.在此种类型的离散型时序图便是本文所研究的对象.

除此之外,还有种特殊的时序图亦称时变图(time-dependent graph)[26,27],其边上的权重是由时间相关的函数$f(t)$所决定,非离散的,如图 2(b)所示.此种类型的时序图由于应用范围实在有限,所以并不在本文研究范围之内.



(a) Temporal graph (b) Time-dependent edge weight

Fig.2 Instance of different types of temporal graph

图 2 不同类型时序图示例

定义 1(时序图). 给定时序网络 $G_T(V,E,T_E,X)$表示为节点间带有时序关系的有向时序图.V 表示节点的集合,$V=\{v_1,\ldots,v_n\}$,E 表示边的集合,且$|V|=n$,$|E|=m$.T_E 表示图中所有节点之间存在联系时刻的集合,$T_{(u,v)}$表示在节点 u 和 v 之间存在联系的时刻的集合,如图 2(a),$T_{(a,c)}=\{2,6\}$,且 $T_{(u,v)}\in T_E$.X 表示节点特征向量集合,$X=\{x_1,\ldots,x_n\}$,其中,x_i 表示节点 v_i 的特征向量.

定义 2(到达时间). 给定时序图 $G_T(V,E,T_E,X)$,在时序图 G_T 中,则由节点 u 到达节点 v 的时间表示为 $Arr_{(u,v)}$,且 $Arr_{(u,v)}=T_{(u,v)}+\lambda$.

由于在本文中将λ=0 处理,所以以图 2(a)为例,$Arr_{(a,b)}=T_{(a,b)}=\{1\}$,$Arr_{(a,c)}=T_{(a,c)}=\{2,6\}$.

定义 3(时序可达路径). 给定时序图 $G_T(V,E,T_E,X)$,在时序图 G_T 中,路径$\langle v_1,v_2,\ldots,v_k\rangle$满足时序可达当且仅当 $\min(Arr_{(v_i,v_{i+1})})\leqslant\max(Arr_{(v_{i+1},v_{i+2})})\,|\,(0\leqslant i\leqslant k-2)$.若 $\max(Arr_{(v_{i+1},v_{i+2})})<\min(Arr_{(v_i,v_{i+1})})$,则表示节点 v_{i+1} 和 v_{i+2} 之间的所有联系时间都在节点 v_i 和 v_{i+1} 存在联系之前,即:在节点 v_i 到达节点 v_{i+1} 之后,两节点 v_{i+1} 和 v_{i+2} 之间便不再有关联.

以图 2(a)为例,顶点 a 到顶点 f 存在 3 条时序可达路径:$\langle a,b,c,f\rangle$,$\langle a,c,f\rangle$和$\langle a,d,f\rangle$,以路径$\langle a,d,f\rangle$为例,若顶点 a 到顶点 d 的到达时间由 4 变为 9,则路径$\langle a,d,f\rangle$为非时序可达路径,因为在时刻 9 之后,顶点 d 和 f 之间的联系便中断了.

定义 4(时序图表达学习). 给定时序图 $G_T(V,E,T_E,X)$,时序图节点的表达学习可以形式化地表示为:通过学习函数 f,在采样节点满足时序可达的条件下,将节点 v_i 映射到一个维度为 d 的向量,且 $d<<|V|$,即:

$$f:V\to Z,Z=\{z_1,\ldots,z_n\},z_i\in R^d,$$

其中,向量 z_i 对应节点 v_i 最终的向量表达形式.

2 游走策略在时序图中的限制和不足

基于以上对时序图的定义、特征以及应用场景的介绍,下面对由于受时序图本身特性所限的节点表达问题所面临的限制或挑战进行分析.尽可能地分析出问题的症结所在,是解决问题的第 1 步,然后才能针对症结做出对应的解决策略.

- 限制 1:游走序列不能有效地保留时间因素

在基于游走策略的节点表达学习时,首先需要得到一个节点的游走序列,以图 3 中顶点 a 和 o 为例,当不考虑顶点间的时序关系时,顶点 a 可以通过⟨a,f,n,o⟩和⟨a,e,l,f,n,o⟩两个路径到达顶点 o,并以此得到'a,f,n,o'和'a,e,l,f,n,o'两个游走序列.顶点 a 和 o 的远近只能说明两者在拓扑距离上是接近的.当考虑时序关系时,由于路径⟨a,e,l,f,n,o⟩不满足时序可达性,所以顶点 a 只能通过路径⟨a,f,n,o⟩达到顶点 o.

这样,便可在牺牲些许"拓扑"属性的前提下,更好地保留节点间的时序属性.这在对节点间对时间敏感的实验结果中无疑是更友好的.并且仅仅在拓扑结构上相近的点并不见得就是真的"亲近",当他们不满足时序可达时,可能表明在实际情况中两顶点并没有过多的联系,仅仅受制于拓扑关系,而使得他们看起来很"亲近"而已.

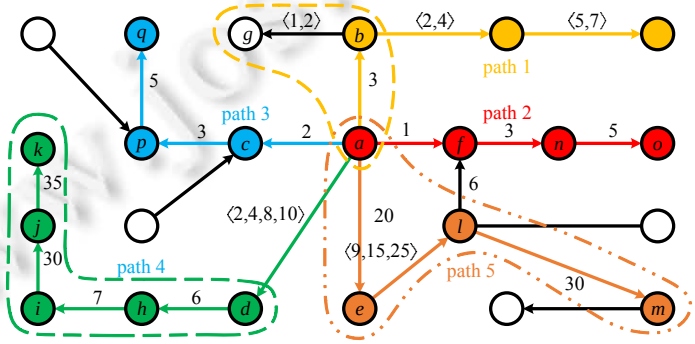


Fig.3 Random walk paths selection in temporal graph

图 3 时序图节点游走路径选择

- 限制 2:局部拓扑结构的动荡变化

有时在很小的时间跨度内,在一个图的局部范围内,节点间关系的动态变化有限,对单一节点 v_i 在不同的时刻游走会得到很多的重复路径.以图 3 中的顶点 a 和 d 为例,在 2 和 4 时刻,顶点 a 在路径 4 上走出的满足时序可达路径是相同的;而当时间的跨度相对较大,到达 8 时刻时,就可以得到一个不同的时序可达路径.当节点间的联系时刻突然发生了较大的变化时,有可能在和其相关的局部拓扑结构会发生了一些"动荡",这个时候可以在动荡之后的时刻进行重新游走,以得出一个新的路径,而在动荡之前,可以尽可能少地采样.

这种现象在实际的网络图中也是普遍存在的,以最典型的交通网络图为例,在不同的时刻(早高峰、晚高峰和平时),每个路段的通行时间是不同的,如此便导致了局部连通性的变化,等由始发地 A 行驶到目的地 B 时,在不同的时刻可能就需要选取不同的路径.

- 限制 3:不同类型的时序网络中动态变化率差异极大

在不同类型的时序网络中,各个节点间联系频率的差异是非常明显的.有的可能是以毫秒级来计算的(通信服务网络等),而有的可能是以分钟、小时,甚至以天来计算(邮件网络等).针对这种情况,对于如何设计一种自适应的节点采样策略,如此可以在不同类型的时序网络数据中解决限制二所面临的问题,便成了一个很大的挑战.

- 限制 4:采样路径的时间跨度问题

考虑到时间的变化对时序图中的拓扑结构和节点间的关系属性影响很大,所以一个节点受其相近邻居的影响也应该是局限在一定的时间范围之内的.正如在真实的一个网络中(脑网络、交通网络抑或邮件网络),脑网络中的一个神经元 a_1 在时刻 t_1 发送到另一个神经元的信息并不会永无休止地在这个网络中一直传递下去,经

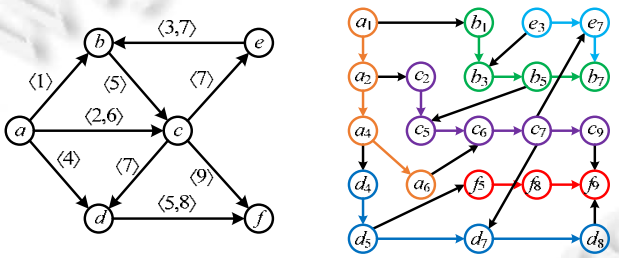
过若干个神经元后,在时刻 t_i 传递到神经元 a_i,而在经过若干个时刻后,有一个信息从神经元中 a_i 发出,而这个信息可能只是单纯的由神经元 a_i 发出,已经和初始神经元 a_1 无半点关系,所以在此时刻以后的时序路径应该归属于神经元 a_i 而不是 a_1.这种现象在时序网络中应该是一种很常见的现象,特别是在社交网络中,信息往往都是即时性的,这种现象应该更为常见.

所以当对初始节点 v_i 进行满足时序可达性的路径上进行采样时,随着采样路径中节点的增加,路径的时间跨度也就越大,这个时候应该检测路径尾部的节点和初始节点 v_i 虽然满足时序可达性,但是两者相互间的关联性是否是间接地借助中间节点完成的,且传递的性质是否已经发生了变化.

3 基本的时序节点嵌入策略

结合现有游走策略的嵌入方法和节点间的时序性,一种最简单的时序图嵌入策略就是在节点游走的过程中记录游走序列中最新节点的到达时刻,进而对下一个可以游走的节点的进行选择.

首先,需要将时序图 G_T 转化为一种更方便操作的静态图,如图 4 所示.图 4(a)是原始的时序图 G_T,图 4(b)是转化后所对应的静态图.在图 4(b)中,具有相同颜色的节点表示在不同时刻的同一个节点.以顶点 a 为例,在图 4(a)中,顶点 a 分别和顶点 b~d 在 1,2,4 和 6 这 4 个时刻存在联系,则在图 4(b)中,有 4 个与之对应颜色相同的顶点 a_1,a_2,a_4,a_6,同时按时间顺序由较早时刻的顶点依次指向较晚时刻的顶点,如图 4(b)中颜色和顶点相同的边所示.图 4(b)中顶点的下标表示顶点 u 到达邻居顶点 v 的时刻,或从邻居顶点 v 到达自身 u 的时刻,即顶点在不同时刻的出边和入边,如其中带箭头黑色边所示.



(a) Original temporal graph (b) Transformed static graph

Fig.4 Instance of temporal graph

图 4 时序图示例

在仅考虑节点间的时序可达的条件下,首先将时序图转化为静态图,图 4 所示;然后结合现有的节点随机游走策略,借助于自然语言处理中的 *skip-gram* 模型,设计出一种最基本的基于时序图的节点嵌入算法.

算法的基本思想:首先,在时序图中,受时间因素所限,节点会丧失游走的"自由"度,当初始节点 v 游走到节点 u 后,需要选择所要游走 u 的邻居节点 $\{u_1,u_2,\ldots,u_n\}$ 时,需要首先判断节点 u 的访问时刻 $Arr_{(v,u)}$ 是否小于等于和其邻居节点 u_i 的最大联系时刻 $\max(T_{(u,u_i)})$,只有满足的条件的节点 u_i 才可以被游走.而后可以从满足条件节点中的一个或若干个进行游走采样,当所有顶点都采样完毕得到游走序列时,便可以通过 *skip-gram* 模型得出节点带有时序关系的向量表达.

算法 1. Basic 时序嵌入.

输入:时序图 $G_T(V,E,T_E)$,*Skip-Gram* 模型,游走步长 L;

输出:时序图节点表达向量 Z.

1. $WL,ul=\varnothing$; //*WL*:所有节点游走序列集合,*ul*:节点 u 的游走序列
2. **FOR** node u in G_T
3. $ul=u$ //对节点 u 初始化其游走序列
4. **WHILE** $|ul|<L$ //控制游走长度

5. $u=ul[-1]$ //取出游走序列的尾部节点作为再次游走的起始点
6. **FOR** node v in N_u //N_u 表示 u 的邻居节点集合
7. Rand choose node v in $V\in N_u$ and $Visit_{(u)}\leqslant\max(T_{(u,v)})$ //有时序可达的节点存在
8. $Visit_{(v)}=t$ for t in $T_{(u,v)}$ and $t>Visit_{(u)}$ //记录节点 v 的游走到达时间
9. $ul=ul\cup v$ //更新节点 v 的游走序列 ul
10. **IF** $\forall_{v\text{ in }U}\max(T_{(u,v)})>Visit_{(u)}$
11. **BREAK** //不满足时序可达,游走终止
12. **END FOR**
13. **END WHILE**
14. $WL=WL\cup ul$ //更新游走序列 WL
15. **END FOR**
16. $z=Skip\text{-}Gram(WL)$ //返回所有节点的向量表达

在第 3 行,首先将节点 u 本身作为有序列的起始点;第 5 行表示从现有的游走序列中的尾部取出节点作为下一次游走的起始点;第 7 行~第 11 行表示当游走序列停留在节点 u 上时,从其邻居节点集合 N_u 中随机选择一个满足时序可达的节点 v 到 u 的游走序列 ul 中去,如果没有满足时序可达的邻居节点,则游走停止;第 14 行~第 16 行表示首先记录所有节点的游走序列(第 14 行),然后通过 *skip-gram* 模型得到各个节点的向量表达.

但是需要注意的是:这种基本的时序图嵌入策略无法很好地解决在第 2 节中所提到的各个限制,特别是无法自动识别出不同类型时序图的动态变化率和采样路径的时间跨度问题.基于此,进一步对基本算法进行改进,以更好地应对以上限制.

4 对基本时序节点嵌入策略的改进

由于不同类型时序图的动态变化率有很大的差异,相对的也就决定了在不同类型的时序图中采样路径的时间跨度也会有很大的不同.由于前一节所提出的 Basic 算法无法有效地解决这两种问题,所以在 Basic 算法的基础上,本节提出了一种自适应时序图动态变化的新型嵌入策略.

4.1 自适应时序节点嵌入

基于上一节中提出的基本嵌入策略所面临问题的基础上,本小节提出一种改进的自适应时序图节点采样策略 ATGEB.策略思想:网络的动态变化是因为信息在其中的流通,信息通过用户间建立联系进行传播.而信息同样也在随着时间的改变而发生变化,即信息在时序网络中是有“传播寿命”的.不同的信息 Inf_i 和 Inf_j 在网络中传播的时间跨度可能有完全重合、部分重合以及完全分离这 3 种情况.在全局的角度下,很难通过时间来区分这些不同的信息;但是一旦具体到各个单一节点 u 中去分析信息的传播时,便可能通过节点间的联系间接地保存这些信息特征.假设信息 Inf_i 和 Inf_j 的传播时间跨度都是 $[t_1,t_2]$,当信息分别传播到节点 u_i,u_j 时,便可以通过观察在 $[t_1,t_2]$ 间与两节点抱有联系的节点来间接地区分两种信息,因为不同的信息所作用的节点也可能是不同的.

通过这种思想,可以对节点 u_i 在不同的信息 Inf_i 下进行游走,这样尽可能保留节点 u_i 在不同类型信息传播下和其相邻节点的时序关系.但是需要注意的是:正常情况下,出于对用户隐私保护的需求,研究者是无法得到这些具体信息的传播途径的.结合信息自身传播的特性,可以从节点活跃时刻集合入手来进行问题的研究.

节点 u 与其邻居节点 N_u 的联系时刻 $T_u=\bigcup_{v\in N_u}T_{(u,v)}$,$T_u$ 同样包含了节点 u 的活跃时间跨度(time span):$TS_u=\max(T_u)-\min(T_u)$和活跃次数$|T_u|$及其活跃频率(activity frequency):$AF_u=|T_u|/TS_u$.在$[\min(T_u),\max(T_u)]$时间段中,由于传播信息的不同,集合 T_u 中的时刻 $t_1,t_2,...,t_{|T_u|}$ 分布是不均匀的,所以可以通过 DBSCAN 这种无监督方式对时刻 t_i 进行聚类这种间接的方式来区分节点 u 所传递的不同信息.将节点的平均活跃时间间隔设置为对象半径 $E=TS_u/(|T_u|-1)$.求解过程如下.

首先对 T_u 中的各个时刻进行排序,得 $T'_u=[t_1,...,t_{|T_u|}]$,则总的时间的间隔为 $\triangle_{T_u}=\sum_{i\in[1,|T_u|-1]}(t'_{i+1}-t'_i)=(t'_2-t'_1)+$

$(t'_3-t'_2)+...+(t'_{|T_u|}-t'_{|T_u|-1})=\max(T_u)-\min(T_u)=TS_u$,间隔个数为$|T_u|-1$.所以,$E=TS_u/(|T_u|)-1)$.

通过 DBSCAN 将 T_u 聚类后会得到若干各集合类 $C_1,...,C_k\left|\bigcup_{i\in[1,k]}C_i=T_u \text{ and } \bigcap_{i,j\in[1,k]i\neq j}(C_i,C_j)=\varnothing\right.$,其中,$C_i=[t^1_{C_i},...,t^{|C_i|}_{C_i}]$,即间接地看为若干个信息的活动范围,如$C_i$集合中的时间跨度为$[t^1_{C_i},t^{|C_i|}_{C_i}]$.然后,在每个已完成的聚类的时间段内对节点 u 进行游走,得出其各个时间段的游走序列,以此来保存在不同"信息"下的和其相邻节点的时序关系.具体的操作步骤在算法 2 和算法 3 中做出详细说明.

这种无监督的聚类方式可以很好地解决不同类型时序图中节点活动频率有差异的问题,因为将对象半径E设置好之后,便可以对各个时刻进行自动聚类.如果某一节点 u 非常规律地向其邻居节点发送信息,即 AF_u 保持不变,如此便间接说明了所传递"信息"的相似性甚至大概率表明一直都是在传递同一种信息,而通过这种自适应的聚类方式,便可以将 T_u 中的元素聚类到同一种类型中去,如此便可以在一定程度上减少了在不同时间段重复采样的可能,避免造成采集数据过分冗余.

算法 2. ATGEB.

输入:时序图 $G_T(V,E,T_E)$,*Skip-Gram* 模型;

输出:时序图节点表达向量 Z.

```
1.  WL,ul=∅;                          //WL:所有节点游走序列集合,ul:节点 u 的游走序列
2.  FOR node u in G_T
3.    T_u,ul=∅                        //初始化节点 u 的活跃(和相邻接点存有关联)时刻和游走序列
4.    FOR node v in N_u               //N_u 为节点 u 的邻居节点
5.      T_u=T_u∪(T_(u,v))
6.      T'_u = Sort(T_u), E = TS_u /(|T_u| −1)   //对 T_u 中的时刻进行排序并设置对象半径
7.      C_1,...,C_k = DBSCAN(T'_u, E)           //对排好序的 T'_u 在半径 E 内进行聚类
8.    FOR C in C_i
9.      ul=ul∪PathTree(u,C)           //汇总各个时间段的游走序列来更新 u 的游走序列
10.   END FOR
11.   END FOR
12.   WL=WL∪ul                        //更新游走序列 WL
13. END FOR
14. Z=Skip-Gram(WL)                   //返回所有节点的向量表达
```

针对在不同时间段 C_i 中节点 u 如何向其相邻节点进行游走的问题(第 9 行),在算法 3 中做出详细说明.其主要思路是:在节点 u 在其邻居节点 N_u 选取一个节点 v,节点 v 在时间段 C_i 内和 u 存在联系,且存在联系时刻 $T\in T_u$ 接近或等于时间段的起始值 $\min(C_i)$.然后由选取的节点 v 向外发散游走,"追踪"信息的传播路径.

算法 3. PathTree.

输入:时序图 $G_T(V,E,T_E)$,节点 u,时间段 C_i;

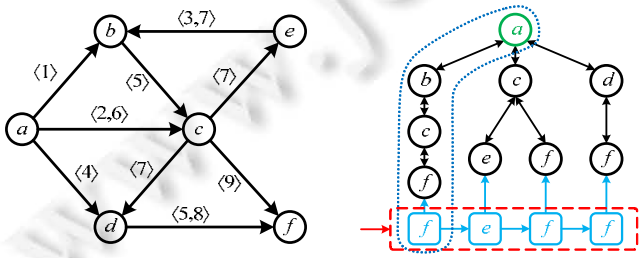
输出:节点 u 在时间段 C_i 内的游走序列 ul.

```
1.  u.prev,u.next=∅,u.name=u   //prev,next 分别保存前序、后续节点
2.  Q.pull(u),TL=∅   //u 作为树的根节点,TL 保存树的叶子节点地址
3.  WHILE Q is not empty   //构造游走树
4.    u←Q.push
5.    FOR each node v in N_u
6.      IF N_u=∅   //无邻居节点则将其记录为叶子结点并终止游走
7.        TL.pull(u)   //记录其为叶子结点
8.        BREAK
```

9. **ELIF** $T_{(u,v)}$ exist in [min(C_i),max(C_i)] and $u.Arr_u$<max(C_i) //判断是否在规定时间段内可达
10. $v.Arr_v=t$ for t in $T_{(u,v)}$ and $t>=Arr_u$ //记录游走到新节点的到达时间
11. $v.prev=u$, $v.name=v$; //指出前序节点方便游走序列的提取
12. $u.next.pull(v)$ //保存后续节点
13. $Q.pull(v)$
14. **ELIF** $T_{(u,v)}$ not exist in [min(C_i),max(C_i)]
15. $TL.pull(u)$ //在规定时间内不可达,则此路径终止将其设置为叶子结点
16. **END FOR**
17. **END WHILE**
18. **FOR** tree leaves tl in TF: //通过叶子节点查询多条游走路径
19. $list=\varnothing$ //初始化游走序列
20. **WHILE** $tl!=\varnothing$
21. $list=list\cup tl.name$ //获取节点名字
22. $tl=tf.prev$ //通过叶子节点向根节点进行节点序列的查询
23. **END WHILE**
24. $ul=ul\cup tl$ //得到一个关于节点 u 的序列集合
25. **END FOR**
26. **RETURN** ul //返回节点 u 在规定时间段内的游走序列

结合示例图 5 对算法 3 进行详细说明.首先需要说明:图 5 中的树是一个多叉时序可达树,即有根节点到叶子结点是满足时序可达的.在第 1 行中,对根节点的名字、后续节点集合和前序节点做初始化.需要注意的是:由于是多叉树,所以树的非叶子节点保留其各个后续节点的地址集合,而前序节点保存一个单一的地址.第 5 行~第 8 行:表示节点的邻居为空,则将此节点记录为叶子结点,将其地址记录到叶子结点集合 TL 中.第 9 行~第 13 行:如果有满足需求的邻居节点 v,即在时间段内 C_i 满足可达关系,则更新节点 v 的到达时间、前后序节点地址,记录节点名字,并将其放入树中.第 14 行、第 15 行表示节点 u 无满足可达性的邻居节点存在,则将 u 作为树的叶子节点,并将其地址记录到 TL 中去.

而后,第 18 行~第 24 行表示通过在 TL 所记录的叶子结点地址进行游走序列提取.如图 5(b)所示:首先,通过 f 以此向上直到 root 顶点 a,得到一个节点序列 ul_1='f,c,b,a'.需要注意的是:这个序列是和真实序列相反的,所以在算法实现时,需要将每次提出的节点名字放在序列最前面,这样便可以得到真实的序列 ul_1='a,b,c,f'.第 24 行中的 ul 表示节点的在这个时间段内的序列集合,以图 5(b)为例,即 ul=[[a,b,c,f],[a,c,e],[a,c,f],[a,d,f]].这样便可得到了一个节点 u 在时间段 C_i 内和信息 Inf_i 相关的游走序列集合.



(a) Temporal graph G_t (b) Path tree for node a

Fig.5 Random walk paths selection in temporal graph

图 5 时序图节点游走路径选择

4.2 嵌入准确性的简要分析

首先简要介绍 *skip-gram* 模型在节点嵌入中的运行原理,从节点的表达向量是如何得出的角度进行分析.如

图 6 所示,节点的 *one-hot* 向量由隐藏层加权后到输出层分类,通过计算在移动窗口 w 内节点和其余各个节点的概率值和 *softmax* 层的输出值的差异作为损失量,进行向后传递更新各层的权重值,然后通过移动窗口进行持续更新.最后,通过隐藏层 H 的权重值得到相应节点的向量表达 Z.

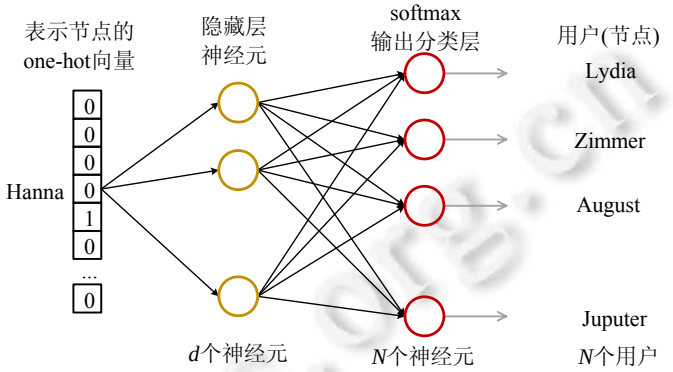


Fig.6 Instance for nodes embedding

图 6 节点向量嵌入示意图

假设时序网络中存在用户传递的信息 I 个:$Inf_1,Inf_2,\ldots,Inf_I$,同时存在的节点标签(类)L 个:$Lab_1,Lab_2,\ldots,Lab_L$.而不同标签的用户对不同信息的敏感度是不一样的,这也就说明标签决定了用户对不同信息的接收和传播的概率是不同的,即通过不同的信息用户"接触"到不同的用户.反之可以推断,若干种信息 $Inf_i,Inf_j,\ldots,Inf_k$ 可以间接决定用户的标签,$f(Inf_i,Inf_j,\ldots,Inf_k)=Lab_i$,所以通过用户所接收或传播的信息类别,可以更准确地判断出其所属的标签.

在节点分类的神经网络中,两个节点的向量相似度越高,也就意味着同属于一种类的可能性越高,即:

$$[sim(z_i,z_j)>sim(z_i,z_k)]\approx[P(u_i,u_j\in Lab_l)>P(u_i,u_k\in Lab_l)].$$

将在图 6 中的用户列表示为 U,*softmax* 层表示为输出层 O,隐藏层表示为 H,对于一个已经训练好的 *skip-gram* 模型,对于属于同 *Lab* 的用户节点 u 和 v,假设其经 *softmax* 层输出后的前 w 个类对应用户分别为(U^{u_1},U^{u_2},...,U^{u_w})和(U^{v_1},U^{v_2},...,U^{v_w}),而两者的嵌入向量 z_u,z_v,且已知隐藏层神经元的值 $H=W_H\times[0,0,0,\ldots,1,0]^T$ 直接决定了节点的向量和节点的经输出层后的分类 U,即 $H^u\to z_u,H^v\to z_v$,已知两节点 u,v 同 *Label*,则说明两者的嵌入向量相似:$z_u\approx z_v$,由此可知 $H^u\approx H^v$,而隐藏层同样决定了输出层,所以 $(U^{u_1},U^{u_2},...,U^{u_w})\approx(U^{v_1},U^{v_2},...,U^{v_w})$.

上述主要描述了在进行节点嵌入网络中的向前传播机制,基于此,进一步说明隐藏层的更新方式.假设节点 u 在一个游走序列 *ul* 中,以 u 为中心所构成的一个大小为 w 的窗口 $Win_u=\{u_1,\ldots,u,\ldots,u_w\}$,则在此窗口内的网络训练误差 $\Delta E=\sum_{v\in Win}(y'-P(u,v))^{1/2}$,其中,$y'=Onehot_u\times W_H\times W_O$.由于 Win_u 的规模有限,不可能将所有节点放入进去,所以对除 u 以外的节点,其在窗口 Win_u 的可能性越小,则 $P(u,v)$的值也就越小,则随着训练的增加,由输出层 *softmax* 的特性可知,对 u 的 *One-hot* 向量 $Onehot_u$ 在 *skip-gram* 网络中的分类,也就越倾向于在 Win_u 出现频率高的节点.由此可知,节点 u 的窗口 Win_u 中的节点出现频率也就影响了其在 *skip-gram* 中的分类,进而由上段中的分析可知,也就影响了其隐藏层中神经元的值 H^u 来影响 z_u.

所以,通过以上的分析可知:对于属于同 *Label* 的节点 u 和 v,要使得两者的嵌入向量 z_u,z_v 相似,则需要使得在所有游走序列 *ul* 中的分别以 u 和 v 为中心的移动窗口 Win_u,Win_v 中出现的高频率节点集合尽可能高地重合,且同属相同的 *Label* 的节点应该有着相似的重合度.因为如果不同的 *Label* 的节点有了相似的重合度,便也意味着可以得到相似的表达向量 z,而这显然与两者的 *Label* 不同这一前提是矛盾的.

对于同 *Label* 两节点 u 和 v,设其可能的游走节点的集合分别为 S_{ul}^u 和 S_{ul}^v,在算法 2 中可能的游走集合分别为 T_{ul}^u 和 T_{ul}^v,且易知 $T_{ul}^u\subseteq S_{ul}^u,T_{ul}^v\subseteq S_{ul}^v$,则对于游走窗口 $Win_u=\{u_1,\ldots,u,\ldots,u_w\}$和 $Win_v=\{v_1,\ldots,v,\ldots,v_w\}$,若两节点的 *Label* 与 Inf_i,Inf_j 相关,设受两信息影响而活跃的节点集合为 S_{Inf},$S_{Inf}\in T_{ul}^v\cap u,v\in S_{Inf}$,而 S_{Inf}中的节点也就间接地

表明他们是在 Inf_i,Inf_j 的影响下活动的.所以相较于随机游走的策略,在本文的游走策略下,在含有两节点 u 和 v 的游走路径 ul 中,在节点 $u(v)$之前和之后的 $1/2(w-1)$个节点(即窗口 Win_u 或 Win_v 内的节点)也都更倾向于都包含在 S_{Inf} 中,即窗口内的节点和 $u(v)$同 *Label* 的概率更大,所以表明以两节点为中心的窗口内出现共同节点的可能性也就越大(相较于随机游走的方法).如前分析,共同高频率节点越多,也就越可以得到更为相似的节点向量.

5 图的重要节点采样

在图数据结构中,由于其复杂的拓扑结构,通常都会有大量的节点在很短的距离内聚集在一个社区中.而通过节点嵌入算法的思想可以看出:在一个密集的社区内,其中大量的节点会游走出很相似的序列,进而也会产生相似的向量表达.而这些非常相似向量并属于同一种类 C_i 的话,那么在进行节点分类的任务工作中,如果应用要在尽可能短的时间内,且可以承受模型有一定误差的这种需求的话,是否可以通过采样同一类型中具有相似向量节点中的若干个节点向量进行神经网络中参数的训练呢?这样便可以在尽可能短的时间内训练出神经网络中的参数权重.

对于节点 u_i,u_j 和 u_k,其所对应的游走序列分别为 u_il,u_jl 和 u_kl,如果 u_i 和 u_j 的距离 $d_{(u_i,u_j)}>d_{(u_j,u_k)}$, 说明节点 u_i 相较于节点 u_k 更接近于 u_j.则在一般情况下,两节点的共同邻居同样也满足 $N_{u_i}\cap N_{u_j}>N_{u_j}\cap N_{u_k}$. 如此,对于节点 v,在经过 w(假设其大小和 *skip-gram* 模型窗口一致)长度游走的范围内,当其游走序列 vl 在经过节点 u_j 后,经过节点 u_i 的概率同样大于经过 u_k 的概率,即 $P(vl\rightarrow u_j|vl\rightarrow u_i)>P(vl\rightarrow u_j|vl\rightarrow u_k)$;反之,序列由节点 u_i 到节点 u_j 的概率同样也大于 u_k.由此可得:对于任意节点的游走序列 $wl\in WL$,在一个窗口大小为 w 的范围内,其同时包含 u_i,u_j 的概率大于同时包含的 u_j,u_k 概率,即 $P(u_j,u_i\in wl)>P(u_j,u_k\in wl)$,在全局的角度来看,$\sum_{wl\in WL}P(u_i,u_j\in wl)>\sum_{wl\in WL}P(u_j,u_k\in wl)$,所以 u_i,u_j 共同在窗口 w 范围内出现在 WL 中的次数要大于 u_j,u_k.而在 *skip-gram* 模型中,节点 u 向量是通过在序列 wl 中和 u 共同在窗口 w 中的其余节点进行更新的.由此可知,对于三节点所对应的向量 z_{u_i},z_{u_j},z_{u_k},则前两者应该具有更大的相似性,即 $sim(z_{u_i},z_{u_j})>sim(z_{u_j},z_{u_k})$.

根据以上分析可知:可以选取图中的若干个稠密子图 $g_1,g_2,\ldots,g_m$ 作为密集社区来选取重要节点,稠密子图 g_i 需要满足对于任意两节点 $v_i,v_j\in g_i$,两节点的距离 $d_{(v_i,v_j)}$ 应该尽可能地小.首先最容易想到的是通过 *k-core* 来进行稠密社区的挖掘,虽然 *k-core* 可以通过提高 k 值的方式得到稠密的子图 g_i,但是这种方式得到的子图并不满足使得 $d_{(v_i,v_j)}$ 尽可能地小.以图 7 为例,去掉节点 e,f,g,剩余的节点便可以得到一个完整的 2-*core* 子图,但是通过节点 c 和 d 可以发现,$d_{(c,d)}$=6,而通过小世界原理可知,这在图数据网络中已经可以看成一个很长的距离.所以对社区的挖掘,仅仅通过稠密性是难以达到目标需求的,还需要限制社区中节点的相互距离,而在文献[28]中所提出的 *kr-Clique* 算法思想可以满足这种需求,其中:k 指的是节点度数;r 指的是节点跳数,即在社区内任意两节点可以在 r 跳内可达.

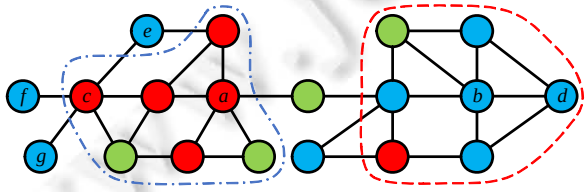


Fig.7 *k*-core communities in graph

图 7 图的 *k*-core 划分

文献[28]中的 kr-Clique 算法旨在找出所有满足需求的社区,但是在本文中这是没有必要的,重要节点的采样旨在选取相对于节点类 C_i 更具有代表性的节点向量作为训练样本集进行训练,而无需选取所有的节点.在这种思想的指导下,可以在图中选取若干个度数较大的节点 $v_{c_1},v_{c_2},\ldots,v_{c_n}$ 作为中心节点,然后以这些中心节点选取其邻居的度数大于 k 的节点 $u\,|\,(u\in N_{v_{c_1}}\cap d(u)>k)$,然后通过节点 u 向外进行发散,选取节点 u 满足条件的邻居

节点,循环 r 次,即可以得到一个关于中心节点 v_{c_2} 的粗略的 *kr-Clique*,这已经可以满足需求.当一个中心节点 v_{c_1} 的 *kr-Clique* 社区选取完毕后,对其中的节点按类进行聚类,得到节点集合 S_{C_1},S_{C_2},...,S_{C_c},然后可以按照一个很小的比例(如 10%)从各个聚类中随机选取节点向量作为训练集,而非直接将图中的 60%节点作为训练集,因为这60%中的节点很多都是属于相同的类别,且向量具有较高的相似性.然后可以进行关于下一个中心节点 v_{c_2} 的重要节点选取.

对于中心节点和重要节点的选取工作,本小节通过一种类似于影响力最大问题中基于节点度的启发式算法[29]进行选取,详见算法 4 所示.

算法 4. INS(important nodes sampling)算法.

输入:时序图 $G_T(V,E,T_E)$,节点类,C_1,C_2,…,C_c,中心节点个数 N;

输出:重要节点的向量集合 Z^{IN}.

1. $krC=\varnothing, i=0, Z^{IN}=\varnothing$ //*krC* 表示已选取的社区节点集合
2. **WHILE** ($i<N$)
3. **FOR** node u in $W=G_T \cap krC$
4. Chose u_i as central nodes if $d(u_i)>d(u_j)|(u_j \in W)$ //选择节点度数最大的节点作为中心节点
5. $krC_{u_i}=u_i, BN_1=u_i, BN_2=\varnothing$ //BN_1,BN_2 存储社区的边界点
6. **FOR** j in *range*(r) //社区节点间的最大跳跃数
7. **WHILE** $BN_1!=\varnothing$
8. $v=BN_1.pop$ //通过边界点选取新的节点进入社区
9. $BN_2.pull(v_i), krC_{u_i}.pull(v_i)$ for v_i in N_v and $d(v_i)>k$
10. **END WHILE**
11. $BN_1=BN_2, BN_2=\varnothing$ //通过 BN_2 更新边界点 BN_1
12. **EBD FOR**
13. **END FOR**
14. Chose important nodes S in krC_{u_i} in a certain proportion //选取重要节点集合 S
15. $Z^{IN}=Z^{IN} \cup S, krC=krC \cup krC_{u_i}$
16. **END WHILE**
17. **RETURN** Z^{IN} //返回重要节点向量集合

第 1 行的 *krC* 表示已经选取的社区节点集合,i 控制社区中心节点选取的数量.第 3 行、第 4 行表示在选取新的中心节点 u_i 时,使得 u_i 不在已经选取的社区中,即与现存的所有的中心节点的距离都要至少大于等于 2,且 u_i 的度数在剩余节点中保持最大,这样可以在图中尽可能广的范围内选取中心节点,而不是集中在图的某一个局部地区.以图 7 为例,首先选取 a 为中心节点选取一个(3,3)-*Clique*,然后在剩余节点中,将 b 作为中心节点.第 5 行~第 12 行表示以节点 u_i 为中心的 krC_{u_i} 社区的形成,首先通过 BN_1 寻找满足条件的邻居节点,将社区 krC_{u_i} 进行向外扩散,并将边界点记录到 BN_2 中,第 1 轮完毕后,将 BN_2 中的边界点赋值到 BN_1 进行新一轮的扩散.第 14 行为重要节点选取,具体细节已在上段中说明.

而以上的重要节点的选取都是在稠密社区的基础上完成的,但是在实际的图数据中,有很多的节点或者新的连通结构体都是游离在这些社区之外.为了使得选取的节点不仅仅具有代表性,还应更具有统一性,所以最后还需要在这些“游离”的节点中按照同样的比例随机选取节点添加到训练样本集合 Z^{IN} 中去.

6 实验和评估

在算法实验的设计和验证上,为了充分验证算法在不同类型数据上的表现,本文选取了 4 种在节点(边)规模上、时序边分布、稠密度等图结构方向都有明显不同的真实数据集作为输入数据,并在节点聚类、链接预测、

节点分类和节点时序可达检测这 4 个方向对算法进行验证.

6.1 实验数据和参数设置

(1) 实验数据

数据集 D0[30]和 D1[31]分别取自两个名为 Bitcoin OTC 和 Bitcoin Alpha 的比特币交易平台,主要注意的是:这两个数据集为用户的金融交易数据,非普通的社交网络,用户与其邻居只存在一次联系,从数据中的时序边等于静态边可以看出.数据集 D2[32]和 D3[32]分别取自名为 Math Overflow 和 Super User 的堆交换网站中的时序网络数据.各个数据集节点规模、静态边、时序边以及节点的活跃频率的分布情况详见表 2.并且从表 2 的数据可以看出:各个数据集在不同的指标上有很大的差异,满足了实验需求.

Table 2 Info of data sets

表 2 实验数据集信息

数据集	节点数	时序边	平均时序边	静态边	平均静态边	时间跨度(天)	活跃频率
D0	6k	36k	6.05	36k	6.05	1 903	18.91
D1	4k	24k	6.39	24k	6.39	1 901	12.62
D2	25k	507k	20.41	240k	9.67	2 350	215.74
D3	194k	1443k	7.44	925k	4.77	2 773	520.38

(2) 对比算法

本文除了在数据集上完成了本文所提出的算法,同样对以下 3 种算法在数据集上进行了复现作为对比.由于其中某些算法不支持时序性可达的游走策略,在数据集上进行复现选择将数据作为静态图看待,即忽略边上的时间戳:

- DeepWalk.首先,将时序图 G_T 中的时间戳删除,将其转化为静态有向图;然后,在静态图的基础上通过 DeepWalk 算法对节点进行向量化表达;
- Node2vec.和 DeepWalk 算法相比,node2vec 算法的不同之处在于:当游走经过节点 t 到达 u 后,向邻居节点 v 进行游走时,首先需要计算 u 的每个邻居节点的游走概率 $\alpha_{pq}(t,v)$,然后借助于 Alias 采样方法选择下一步游走的节点 v.游走的概率计算公式见公式(1):

$$\alpha_{pq}(t,v)=\begin{cases}1/p, & \text{if } d_{tv}=0\\ 1, & \text{if } d_{tv}=1\\ 1/q, & \text{if } d_{tv}=2\end{cases} \tag{1}$$

  其中,d_{tv} 表示两节点路径长度.在本次实验中,p 设置为 0.25,q 的值设置为 4,然后通过 Alias 算法进行游走采样;
- CTDNE[33].这种算法的思想与本文的 Basic 和 DeepWalk 算法一致,只是节点在游走的过程中,作者设置一个节点对时间相关的概率,即有偏采样,以此来左右节点对邻居节点的选择.

(3) 实验环境

64 位操作系统:Windows10,CPU:i5-8400@2.80Hz,内存为 24G,硬盘 500GB,编程环境为 Python 3.7.

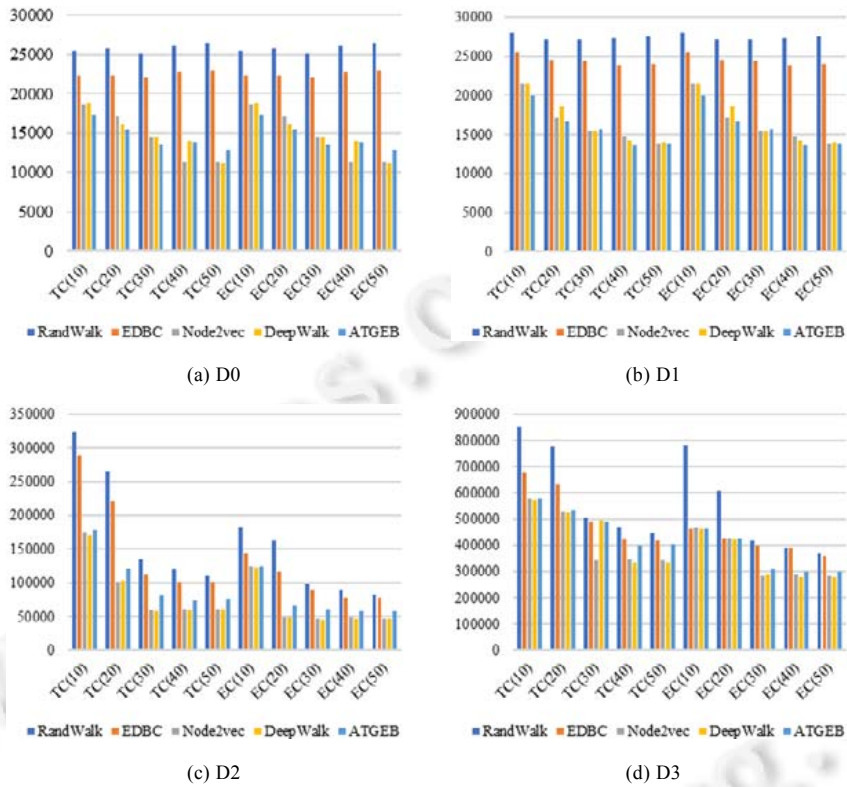
(4) 参数设置:

在通过 *skip-gram* 生成节点向量时,窗口大小设置为 5;节点的游走步长 L 分别设置为 10,20,30,40 和 50;节点 u 生成的向量 z_u 的维度大小 d=100;学习率 r=0.01,在分类任务中的损失函数为交叉熵损失函数.

6.2 节点聚类

在节点聚类的实验中,通过两种方式来对各个算法的聚类效果进行评测:跨类的静态边 EC(edges crossing clusters)和时序边 TC(temporal edges crossing clusters)的数量.假设通过节点的表达向量将节点聚类为 N 类:C_1, C_2,…,C_N,用 C^v 表示节点 v 所属的类.则 $EC=\bigcup_{v\in G_T}(v,u)\,|\,S\in N_v\cap(\forall_{u\in S}C^u\neq C^v)$,即:当节点 v 的邻居 u 和 v 属于不同的类时,边(v,u)便可以看为跨越不同的聚类.$TC=\bigcup_{e\in EC}T_e$,e 表示在 EC 中的边,T_e 表示边 e 上的时刻集合.

为了尽可能公允地分析实验的结果,在实际的实验中,分别将节点聚类成 4 类和 5 类,然后在计算出各种算法在不同聚类的情况下的表现.详情如图 8 和图 9 所示,其中,*TC*(*i*)和 *EC*(*i*)中的 *i* 表示的是节点的游走长度:由 10~50.

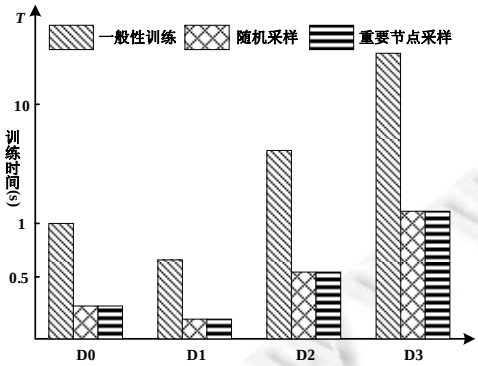


(a) D0 (b) D1

(c) D2 (d) D3

Fig.8 Node clustering in temporal graph

图 8 时序图节点聚类



数据集 / 算法	D0		D1		D2		D3	
Basic	43.2	46.7	50.1	52.1	49.7	51.2	47.5	50.6
Deepwalk	46.5	49.4	52.4	54.5	54.9	56.3	53.8	55.9
Node2vec	47.3	49.6	52.5	54.3	55.6	57.1	52.9	55.7
CTDNE	45.8	47.6	51.0	53.1	51.4	54.2	50.1	52.6
ATGEB	46.3	49.7	51.3	53.5	53.2	55.1	51.7	54.8

Fig.9 Analysis for important nodes sampling

图 9 重要节点采样分析

图 8 中的各个子图的纵坐标表示数量,横坐标表示在 *TC* 和 *EC* 下的节点游走的长度 *L*:10~50.由图 8(a)和图 8(b)中的数据可以看出:本文的方法在起始阶段会更快地收敛,但是随着游走长度的增加逐渐趋于平稳.这是因为受采样策略所限,因为一旦采样策略受时间所限,则其游走长度到达一定的长度后便会自动终止.可以看出,

本文的方法基本上会在 30 步以后趋于平稳.由此可知,游走的序列长度大部分应该在 30 步以内.在图 8(a)和图 8(b)中可以看出:ATGEB 在前期有一定的优势,但是随着游走长度的增加,便和 DeepWalk 和 node2vec 保持基本一致.但是在图 8(c)和图 8(d)中可以看出,后两种方法在在数据集 D2 和 D3 中更有优势.通过此实验,旨在表明对于传统的节点聚类问题上,基于时序节点嵌入的策略对数据集的属性较为敏感,即对某些特定类型的数据集有一定的优势,但不如传统的 DeepWalk 和 node2vec 这类算法可以较好地应用在各类型数据中.

6.3 连接预测和可达性检测

在此次实验中,首先在链接预测的任务中,分别对每个节点 v 选取样本向量,选取节点 v 的 $1/2|N_v|$个邻居节点作为正样本,$z^{(u,v)}=z^u+z^v$,其中,'+'表示拼接,相应的类为 1,表示两节点有边,然后在图中随机选取同样数目节点 $u_1,u_2,...,u_{1/2|N_v|}$,然后判断节点 v 和 u_i 是否存在边:如有边,则向量 $z^{(v,u_i)}$ 对应的类为 1;否则为 0.而在时序可达性检测的实验中,通过计算两节点是否满足时序可达,进行分配向量的类别.在对所有节点完成样本采集后,取其中的 20%为训练集,其余的 80%为测试集,测试结构见表 3,其中的数据值都是取自不同游走长度下所得到结果的最高值.

Table 3 Accuracy of link prediction and temporal achievability (%)

表 3 链接预测和时序可达性检测准确性 (%)

问题 / 数据集 / 算法	链接预测				时序可达性检测			
	D0	D1	D2	D3	D0	D1	D2	D3
Basic	64.2	61.0	60.9	56.6	65.3	63.2	62.9	58.8
Deepwalk	79.3	**82.5**	**71.9**	80.8	76.3	78.6	70.8	78.3
Node2vec	**81.9**	81.8	70.2	79.7	79.4	79.5	68.1	77.9
CTDNE	67.9	64.7	65.9	59.4	81.7	80.7	71.9	80.3
ATGEB	78.4	76.4	71.6	**81.5**	**85.3**	**82.9**	**72.8**	**82.7**

首先需要说明的是:时序图节点的嵌入在链接预测的问题上受限于其游走策略的影响,其结果有很大的可能性弱于普通游走策略的,特别是在节点间联系时刻很少的情况下.正如表 3 所示:在数据集 D0 和 D1 中,基于时序采样的方法在链接预测上有明显的弱势.因为从 D0 和 D1 中的数据信息可以看出,节点间的联系是非常单一的.由于是货币交易网络,用户间在全局范围内只有一次联系,即时序边和静态边相同,这也就导致任何基于时序游走策略的方法的游走长度都会更大程度地受时间因素所限制.但是当随着用户联系次数增加,本文所提出的算法和 DeepWalk 和 Node2vec 的差距也就越小,在数据集 D2 中只有微小的差距,而在 D3 数据中生成了更好的实验结果.

在节点间的时序可达性检测问题上,形势则会发生改变.在此问题上,传统游走策略的准确度会有明显的下降.因为传统的游走策略是完全没有考虑节点间的时序关系的,即节点的表达向量中并没有充分保存节点间的时序特性,而基于时序图游走的策略在采样过程中遵循时序可达这一条件的.并且从中可以看出:ATGEB 在数据集 D0 和 D1 中有着明显的优势,而在 D2 和 D3 中不如前者更明显.这是因为在后两种的数据集中,节点的活跃频率更高,节点间的联系更频繁,节点的路径所满足时序可达性的几率也就越大(相较于前两种数据集).

6.4 节点分类和重要节点采样

在本小节的实验测试中,主要集中在节点的分类和重要节点的采样.在重要节点的采样中,对比了对于不同训练集且选取方式不同下的训练时间和结果.其中的训练集的选取方式有 3 种:一是正常取 20%的数据集 S 作为训练集;一种随机取十分之一 S 规模的数据集作为训练集;最后一种是在第 4 节中所介绍的重要节点采样的方式,同样选取和第 2 种同等数量的数据集作为训练集.然后对比在完全相同的神经网络中的训练时间和测结果的变化情况,来判断重要节点的采样是否可以通过在尽可能少的误差内用更短的时间训练数据.

在测试的时序图数据集中并没有对节点所属的类进行标注,所以在此次实验中,通过两种方式对节点进行标注 3 种类型来进行模拟实验验证.因为在实际的数据集中,属于相同类的节点相对而言都有更短的跳数距离,基于此,通过以下两种方式对节点进行模拟标注分类.

- 第 1 种方式是选取若干个距离尽可能远的节点 $u_1,\ldots,u_k$,分别标注为 C_1,C_2,C_3,然后以这些节点为中心,取得这些节点相近邻居的集合 S_{u_1},将这些集合中的节点 60%的节点标注和中心节点相同的类,然后剩余的节点进行随机标注;
- 第 2 种方式是中心节点在选取临近节点时需要满足时序可达,得到集合 $S_{u_1}^T$,后续的标注类的方法同第 1 种.

通过这两种方式的对比,来尽可能真实、客观地评价本文方法的实验结果.而之所以没有使用对节点进行随机标注 3 种类型的方式,因为随机标注的策略会导致本来非常相似的两个向量赋予了不同的类,导致最后只能识别出一种类型,经过实验而放弃了这种策略.尽管如此,这种模拟的方法还是难免出现上述问题,只能做到尽可能的准确.

实验的结果见表 4,从中可以看出:在考虑节点间的时序性时,传统的采样方法在结果上都会呈现出不同程度的下降;基于时序游走策略的方法可以更好地保留节点间的时序关系,从而得出更好地实验结果.在图 9 中,注意:由于有的数据集规模较小训练时间较短,为了更方便看清实验的结果,左子图的横坐标并不是按比例绘制的.同时,以第 1 种方式(不考虑时序性)的采样策略进行重要节点采样的结果对比.其中,在各个对应数据下方,左边的数据集为随机选取的结果,相邻右边的数据集为通过重要节点策略选取的节点的实验结果.结合左右两个子图可以看出,重要节点采样的策略可以在一定的误差损失范围内(本实验中可以控制在 2%以内)且在更短的时间完成神经网络模型的训练.

Table 4 Accuracy of nodes classification (%)

表 4 节点分类准确性 (%)

问题 / 数据集 / 算法	不考虑时序性				考虑时序性			
	D0	D1	D2	D3	D0	D1	D2	D3
Basic	47.8	53.1	53.8	52.3	47.4	53.7	54.9	53.1
Deepwalk	49.7	55.6	57.6	**56.6**	48.2	54.3	55.8	54.3
Node2vec	50.2	**55.9**	**57.9**	56.4	48.6	54.5	56.5	55.2
CTDNE	48.6	54.3	55.7	54.7	49.2	54.9	54.9	53.8
ATGEB	**50.4**	54.8	56.3	56.2	**51.1**	**55.8**	**58.6**	**57.6**

7 相关工作

和静态图不同,时序图中的边会随时间呈现出两种相互转化的状态:活跃和非活跃.时序图中的顶点只有在边处于活跃状态下是存在联系的.现实中有很多时序网络的例子,如:

(1) 通信网络:电邮和手机通讯、短信等都是非常典型的点对点时序网络;

(2) 一对多的信息传播网络:这种时序网络特点是单一用户向其余多个用户传播信息;

(3) 生物信息网络:如脑神经元网络、代谢网络和蛋白质互作用网络等.

如今,在数据库领域,面向时序图的研究工作主要集中在可达性查询、最短时序路径和到达时间预测[25−27]等方向.

图节点嵌入的工作最早引起人们的注意始于 DeepWalk[2]算法的提出,DeepWalk 算法开创性地将图的随机游走策略和自然语言处理中的 *skip-gram*[3]文字向量化表达模型相结合,通过这种极具创新性的方法完成了图节点的向量化表达工作.而后的研究者们[4−7]在 DeepWalk 算法的基础上,希望通过一种有偏采样的方式对这种方法进行改进,进而设计出了不同的节点采样策略,如 node2vec,LINE,PTE 以及 stru2vec 等节点嵌入方法.虽然这些算法在相应的图数据集上,在链接预测和节点分类等问题上都宣称取得了更好的实验效果,但是相较于 DeepWalk 算法并没有质的提升,并且不同的采样策略在不同的数据集上效果有时会出现明显的差异.由此给后续研究者们在研究节点嵌入表达问题上提供了启发,即:在现有神经网络框架训练能力有限的今天,图节点嵌入的向量化表达的研究需要着眼于研究者所想要解决的具体问题和图数据所属的拓扑结构特征.如在异质图上做节点嵌入和在知识图谱上的研究方式是不同的,而当所要解决的问题不同时,如做节点分类和商品推荐,也会

因所研究问题的不同而做出符合问题实际需要的研究策略.

社区发现[28,34,35]是一个在图数据挖掘中经久不衰的研究方向.在问题的起始阶段,研究者们主要致力于将拓扑距离相近的网络用户聚集在一起,形成一个社区.而随着问题的深入研究和现实一些实际的应用需求,研究者们更倾向于将“兴趣点”一致的用户聚集在一起,形成一个基于共同“兴趣点”的社区.在这个社区中,各个用户通常有着共同的兴趣或类似的属性,如此可以通过这些兴趣点,更精准地向用户推荐他们感兴趣的信息或者所需商品.

8 展 望

本文的研究工作主要集中在面向无属性的时序图节点嵌入问题上,并进行了相关实验,在各个方面验证了所提出算法的有效性和可扩展性.在属性缺失的时序图上是无法使用图卷积的思想对节点进行表达学习的,所以本文中的嵌入策略无法有效地将节点的属性信息嵌入到节点向量中去.在未来的研究工作中:(1) 将考虑基于属性时序图的节点表达工作的研究,拟在现有图卷积思想的基础上研究节点属性和节点时序关系的关联,进而提出一种可以在属性时序图上进行时序图卷积的图表达学习的策略;(2) 本文也尝试了面向图数据的重点采样工作,在后续的研究工作中,会对样本节点做出更充分的考虑,不仅仅只是局限于节点的拓扑结构,同样也将考虑节点间的属性关系,借助于图数据相关挖掘算法进行重要节点的选取,力求使得所选取的节点向量更具有代表性和统一性,从而能更好、更快地在神经网络中训练出所需的权重参数.

References:

[1] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans. on Neural Networks, 2009,20(1):61−80.

[2] Perozzi B, Al-Rfou R, Skiena S. DeepWalk: Online learning of social representations. In: Proc. of the 20th ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining. 2014. 701−710.

[3] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781v3.

[4] Tang J, Qu M, Wang MZ, Zhang M, Yan J, Mei QZ. LINE: Large-scale information network embedding. In: Proc. of the 24th Int’l Conf. on World Wide Web. 2015. 1067−1077.

[5] Tang J, Qu M, Mei QZ. PTE: Predictive text embedding through large-scale heterogeneous text networks. In: Proc. of the 21st ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining. 2015. 1165−1174.

[6] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proc. of the 22nd ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining. 2016. 855−864.

[7] Leonardo FRR, Pedro HPS, Daniel RF. struc2vec: Learning node representations from structural identity. In: Proc. of the 23rd ACM SIGKDD Int’l Conf. on Knowledge Discovery and Data Mining. 2017. 385−394.

[8] Qiu JZ, Dong YX, Ma H, Li J, Wang KS, Tang J. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In: Proc. of the 11th ACM Int’l Conf. on Web Search and Data Mining. 2018. 459–467.

[9] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems. 2017. 1024−1034.

[10] Kipf TN, Welling M. Semi-Supervised classification with graph convolutional networks. In: Proc. of the ICLR (Poster). 2017.

[11] Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In: Proc. of the ICLR (Poster). 2018.

[12] Wang YS, Yuan Y, Ma YL, Wang GR. Time-dependent graphs: Definitions, applications, and algorithms. Data Science and Engineering, 2019,4(4):352−366.

[13] Takaguchi T, Yano Y, Yoshida Y. Coverage centralities for temporal networks. European Physical Journal B, 2016,89(2):35.

[14] Frand D, Masoud TO, Jörg-Rüdiger S. Shortest paths in FIFO time-dependent networks. Algorithmica, 2012,62(1-2):416–435.

[15] Rossi L, Musolesi M, Torsello A. On the *k*-anonymization of time-varying and multi-layer social graphs. In: Proc. of the 9th Int’l Conf. on Web and Social Media. 2015. 377−386.

[16] Przytycka TM, Singh M, Slonim DK. Toward the dynamic interactome: It’s about time. Briefings in Bioinformatics, 2010,11(1): 15–29.

[17] Han JD, Bertin N, Hao T, *et al*. Evidence for dynamically organized modularity in the yeast protein-protein interaction network. Nature, 2004,430(6995):88−93.

[18] Lèbre S, Becq J, Devaux F, *et al*. Statistical inference of the time-varying structure of gene-regulation networks. BMC Systems Biology, 2010,4(1):1−16.

[19] Wu H, Cheng J, Ke Y, *et al*. Efficient algorithms for temporal path computation. IEEE Trans. on Knowledge & Data Engineering, 2016,28(11):2927−2942.

[20] Li J, Han ZC, Cheng H, Su J, Wang PY, Zhang JF, Pan LJ. Predicting path failure in time-evolving graphs. In: Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. 2019. 1279−1289.

[21] Hu JL, Yang B, Guo CJ, Jensen CS, Xiong H. Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks. In: Proc. of the IEEE Int'l Conf. on Data Engineering. 2020. 1417−1428.

[22] Kumar S, Hamilton WL, Leskovec J, Jurafsky D. Community interaction and conflict on the Web. In: Proc. of the World Wide Web Conf. 2018. 933−943.

[23] Panzarasa P, Opsahl T, Carley KM. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. Journal of the American Society for Information Science and Technology, 2009,60(5):911−932.

[24] Bai C, Kumar S, Leskovec J, Metzger M, Nunamaker JF, Subrahmanian VS. Predicting visual focus of attention in multi-person discussion videos. In: Proc. of the Int'l Joint Conf. on Artificial Intelligence. 2019. 4504−4510.

[25] Wu HH, Huang YZ, Cheng J, Li JF, Ke YP. Reachability and time-based path queries in temporal graphs. In: Proc. of the IEEE Int'l Conf. on Data Engineering. 2016. 145−156.

[26] Yuan Y, Lian X, Wang GR, Ma YL, Wang YS. Constrained shortest path query in a large time-dependent graph. Proc. of the VLDB Endow, 2019,12(10):1058−1070.

[27] Yuan Y, Lian X, Wang GR, Chen L, Ma YL, Wang YS. Weight-Constrained route planning over time-dependent graphs. In: Proc. of the IEEE Int'l Conf. on Data Engineering. 2019. 914−925.

[28] Bron C, Kerbosch J. Finding all cliques of an undirected graph (algorithm 457). Commun. ACM, 1973,16(9):575−576.

[29] Chen W, Wang YJ, Yang SY. Efficient influence maximization in social networks. In: Proc. of the 15th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2009. 199−207.

[30] Kumar S, Spezzano F, Subrahmanian VS, Faloutsos C. Edge weight prediction in weighted signed networks. In: Proc. of the IEEE Int'l Conf. on Data Mining. 2016. 221−230.

[31] Kumar S, Hooi B, Makhija D, Kumar M, Subrahmanian VS, Faloutsos C. REV2: Fraudulent user prediction in rating platforms. In: Proc. of the ACM Int'l Conf. on Web Search and Data Mining. 2018. 333−341.

[32] Paranjape A, Benson AR, Leskovec J. Motifs in temporal networks. In: Proc. of the 10th ACM Int'l Conf. on Web Search and Data Mining. 2017. 601−610.

[33] Nguyen GH, Lee JB, rossi RA, Ahmed NK, Koh E, Kim S. Continuous-Time dynamic network embeddings. In: Companion Proc. of the Web Conf. 2018. 2018. 969−976.

[34] Wang Y, Jian X, Yang ZH. Query optimal *k*-plex based community in graphs. Data Science and Engineering, 2017,2(4):257−273.

[35] Fan WF, Hu CM. Big graph analyses: From queries to dependencies and association rules. Data Science and Engineering, 2017,2(1): 36−55.

吴安彪(1993－),男,博士生,CCF 学生会员,主要研究领域为图数据库,图神经网络.

马玉亮(1990－),男,博士,主要研究领域为图数据库,基于位置的社交网络(LBSN)挖掘.

袁野(1981－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据管理,数据库理论与系统.

王国仁(1966－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为不确定数据管理,数据密集型计算,可视媒体数据管理与分析,非结构化数据管理,分布式查询处理与优化技术,生物信息学.