软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007654] [CSTR: 32375.14.jos.007654]

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

基于双向选择性状态空间模型的动态图表示学习*

何 萍, 汪雷达, 徐晓华, 严勇林

(扬州大学 信息与人工智能学院 (工业软件学院), 江苏 扬州 225127)

通信作者: 徐晓华, Email: arterx@gmail.com

摘 要: 动态图表示学习通过捕捉实体间随时间演化的拓扑结构与交互模式, 为链接预测等下游任务提供具有时空感知能力的嵌入表示, 从而揭示复杂系统的动态演化规律. 连续时间动态图因其具备丰富的细粒度时间信息, 为社交网络演化等复杂过程提供了更贴近真实世界的建模范式. 然而, 当前连续时间动态图表示学习至少面临 3 个挑战: (1) 在长交互序列中有效提取关键信息需具备强大的长期时间依赖建模能力; (2) 处理长交互序列时需控制计算复杂度以提升计算效率; (3) 传统按交互时间顺序捕获交互模式的方法难以揭示隐藏的非因果性关联. 针对上述问题, 提出一种连续时间动态图表示学习模型, 通过双向选择性状态空间编码机制, 既可捕捉交互序列的长期时间依赖, 又能通过反向路径引入后续交互信息, 打破时间单向性约束, 从而增强模型对全局上下文的理解能力. 大量实验结果表明, 该模型在不同领域真实世界数据集上表现出的预测性能均显著优于基线方法, 同时具有高效的计算效率, 使在有限计算资源下实现长时间双向依赖的建模成为可能.

关键词: 动态图表示学习; 连续时间动态图; 双向选择性状态空间模型; 长期双向依赖关系

中图法分类号: TP18

中文引用格式: 何萍, 汪雷达, 徐晓华, 严勇林. 基于双向选择性状态空间模型的动态图表示学习. 软件学报. http://www.jos.org.cn/1000-9825/7654.htm

英文引用格式: He P, Wang LD, Xu XH, Yan YL. Dynamic Graph Representation Learning Based on Bidirectional Selective State Space Model. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7654.htm

Dynamic Graph Representation Learning Based on Bidirectional Selective State Space Model

HE Ping, WANG Lei-Da, XU Xiao-Hua, YAN Yong-Lin

(College of Information and Artificial Intelligence (College of Industrial Software), Yangzhou University, Yangzhou 225127, China)

Abstract: Dynamic graph representation learning captures time-evolving topological structures and interaction patterns among entities, providing spatiotemporally aware embedding for downstream tasks such as link prediction, revealing the dynamic evolution patterns of complex systems. Continuous-time dynamic graphs, owing to their rich fine-grained temporal information, provide a more realistic modeling paradigm for complex processes such as social network evolution. However, current continuous-time dynamic graph representation learning faces at least three challenges. (1) The effective extraction of critical information from long historical interaction sequences requires robust modeling of long-term temporal dependencies. (2) Handling long interaction sequences requires controlling computational complexity to improve efficiency. (3) Traditional approaches that capture interaction patterns in chronological order struggle to reveal hidden non-causal associations. To address these challenges, this study proposes a novel continuous-time dynamic graph representation learning model. By leveraging a bidirectional selective state-space encoding mechanism, long-term temporal dependencies in interaction sequences can be captured, and subsequent interaction information can also be incorporated through backward pathways. The temporal unidirectional constraint is thus broken, enhancing the model's global context comprehension capability. Extensive experimental results demonstrate that the model consistently outperforms baseline methods in predictive performance on real-world datasets spanning diverse domains, while maintaining high computational efficiency, enabling the modeling of long-term bidirectional dependencies under

* 基金项目: 江苏省自然科学基金 (20201430); 江苏省高校优秀中青年教师和校长海外研究培训计划; 江苏省政府海外留学奖学金; 江苏省研究生科研与实践创新项目 (SJCX25_2293)
收稿时间: 2025-08-26; 修改时间: 2025-11-24, 2026-01-31; 采用时间: 2026-02-24; jos 在线出版时间: 2026-06-03

limited computational resources.

Key words: dynamic graph representation learning; continuous-time dynamic graph; bidirectional selective state space model; long-term bidirectional dependency

1 引 言

动态图通过将实体建模为节点、交互关系表示为时间戳标注的边, 为动态演化系统的分析提供了有效的建模工具[1]. 动态图表示学习旨在将动态图中的节点/边映射到低维向量空间, 捕捉节点间的依赖关系和交互规律, 从而为下游学习任务 (如动态链接预测) 提供动态自适应的嵌入表示, 提升预测的准确性. 近年来, 动态图表示学习已经成为当前的研究热点, 在社交媒体分析[2]、用户-物品交互系统[3]、交通网络[4]及推荐系统[5]等场景中得到广泛应用.

现有的动态图主要可分为两类: 离散时间动态图和连续时间动态图. 离散时间动态图通常将动态图划分为若干时间快照, 在每张快照中利用神经网络学习节点的空间表示, 并结合时序模型, 如循环神经网络 (recurrent neural network, RNN)、长短期记忆网络 (long short-term memory, LSTM) 捕捉节点的动态模式. 然而, 这类方法需要预先设定时间粒度以划分快照, 导致忽略单个快照内节点/边的细粒度时间顺序, 不可避免地造成信息丢失. 相较之下, 连续时间动态图将动态图中的边视为事件流, 通过实时编码节点间的局部交互信息及细粒度时间信息, 有效规避了离散表示中因时间粒度选择不当导致的信息丢失问题. 由于连续时间动态图能够更全面地保留时间信息, 具备更高的灵活性, 因此本文聚焦于连续时间动态图的表示学习研究.

近年来, 连续时间动态图的表示学习方法发展迅速, 目前主流的研究方法包括基于随机游走的方法[6,7]、基于时序点过程的方法[8,9]、基于 RNN 的方法[10,11]、基于注意力机制的方法[12,13]以及基于 Transformer 的方法[14−16]. 然而, 这些方法在捕捉长期时间依赖和计算效率方面仍存在显著局限: 一方面, 它们在建模长期依赖时往往面临稀疏事件的干扰和时间粒度模糊性的挑战. 例如, 在社交网络或用户行为分析中, 节点间的交互可能高度稀疏, 关键事件可能被大量无关事件淹没, 而传统模型因缺乏对事件重要性的区分能力, 难以有效捕捉这些长期依赖关系. 此外, 由于事件的时间戳分布不规则, 有的方法 (如基于循环神经网络的方法) 在处理长序列时容易陷入梯度消失或爆炸问题, 进一步削弱了其对长期模式的感知能力. 另一方面, 计算代价高昂的问题限制了已有方法在大规模动态图中的应用. 连续时间方法通常需要对每个时间戳的事件进行独立处理, 导致时间序列的冗余性显著增加, 模型的计算复杂度呈指数增长, 计算资源急剧增加. 尽管研究者们采取了一些举措来降低模型在处理序列长期时间依赖方面的计算复杂度, 如引入稀疏注意力机制、线性注意力近似等技术[17], 但计算效率提升仍然有限.

在此背景下, 选择性状态空间模型[18]为连续时间动态图表示学习提供了新的解决思路. 它采用动态过滤机制, 能够识别并保留关键的长期依赖事件, 同时抑制冗余信息, 有助于提升模型的长期模式感知能力. 同时, 它的参数化设计和灵活的状态转移机制又显著降低了计算复杂度, 使模型在大规模动态图中具备更高的扩展性与效率. 但是, 传统选择性状态空间模型通常按照时间顺序处理节点历史交互序列, 因此难以发现序列中隐藏的非因果关系模式. 例如, 在社交网络或用户行为分析中, 单向时序建模可能忽略时间序列中的双向关联, 导致对事件多因素依赖关系的片面理解, 从而降低模型对复杂动态行为的预测准确性与解释能力.

针对上述问题, 本文提出了一种基于双向选择性状态空间模型的动态图表示学习模型 (dynamic graph representation learning based on bidirectional selective state space model, DG-BS^{3}M), 旨在通过引入选择性状态空间模型和双向建模机制, 更高效地捕获序列的长期时间依赖关系和非因果关系模式. 本文主要贡献总结如下.

(1) 提出一种双向选择性状态空间编码机制, 同时捕捉动态图中节点正向的交互历史与反向的演进关联, 从而增强对长期依赖关系的建模能力.

(2) 引入选择性机制, 动态过滤冗余的时空信息, 保留关键的动态特征, 提升模型的效率与泛化性.

(3) 优化提取连续时间动态图的交互特征, 并利用选择性机制和状态空间模型的参数化设计, 降低处理历史交互序列的计算复杂度, 提高模型的可扩展性与实用性.

我们通过在多个公开的动态图数据集上与已有的连续时间动态图学习算法进行对比实验, 验证了 DG-BS^{3}M 在动态链接预测任务中的优越性能, 尤其在处理长时间序列时展现出更强的鲁棒性. 此外, 我们还通过消融性实

验, 验证了模型核心组件的合理性.

本文第 2 节回顾相关背景知识. 第 3 节详细阐述提出的基于双向选择性状态空间模型的连续时间动态图表示学习方法. 第 4 节通过大量实验验证所提方法的有效性. 第 5 节总结全文.

2 相关工作

2.1 动态图表示学习

根据动态图的不同类型, 动态图表示学习可以分为离散时间动态图表示学习[19−21]和连续时间动态图表示学习[10−12]. 离散时间动态图表示学习通过对每个时间点的静态图结构进行独立建模, 并借助时序模型关联不同时间点的信息, 从而有效捕捉时序依赖关系. 典型方法包括基于自编码器的方法[19]、基于深度生成的方法[20]以及基于图神经网络的方法[21]. 然而, 这类方法依赖离散时间动态图预设的时间间隔, 容易丢失快照间的细粒度动态. 相比之下, 连续时间动态图学习方法充分利用每条边或节点事件的精准时间戳, 通过连续域建模技术, 将时序交互序列编码为动态嵌入, 实现对图结构演化的连续性表示. 近年来, 随着连续时间动态图的广泛应用, 研究者们提出了多种不同类型的动态图表示学习方法.

(1) 基于随机游走的方法. 这类方法通常从图中的一个节点开始, 按照一定的概率随机选择相邻节点进行移动, 形成不同的节点序列, 通过多次随机游走充分探索图的结构信息, 训练模型以获得节点的嵌入表示进行下游任务. 例如, CTDNE[22]将随机游走应用于连续时间动态网络, 提出了时间随机游走方法, 解决了从连续时间动态网络中学习节点表示的问题. CAWN[6]使用因果匿名游走方法来学习网络动态, 避免了耗时的模式选择和计数过程. NeurTWs[7]结合了时间约束、结构属性和树遍历属性, 通过时空偏置的随机游走提取代表性模体 (Motifs), 从而有效地表示时间节点. PINT[23]基于时间随机游走计数构建相对位置特征, 通过消息传递捕获细粒度交互依赖. 但是, 这类方法的缺点是随着提取的随机游走序列长度的增加, 计算的复杂度显著提高.

(2) 基于时序点过程的方法. 这类方法通过条件强度函数量化事件在单位时间内发生的概率, 从而精确捕捉事件发生的时间间隔分布与历史依赖关系. 例如, GNPP[8]结合时间点过程和神经网络来捕捉动态图中的复杂交互模式和时间连续性. DyRep[9]采用双时间尺度深度时间点过程方法, 结合新颖时间注意力机制整合结构与时间组件, 捕获时序交错的非线性动态. TREND[2]基于霍克斯过程 (Hawkes process) 构建图神经网络, 融合事件与节点群体动态特征, 全局建模图链接形成过程, 捕捉系统时序演化规律. 然而, 随着输入交互序列长度的增加, 此类方法需要的计算资源急剧增加.

(3) 基于 RNN 的方法. 该方法通过引入 RNN 来捕捉网络在时间序列上的演化, 在事件或网络发生变化时更新交互节点的嵌入, 以保持其最新状态. 例如, TGN[10]和 JODIE[11]模型将节点嵌入到交互网络中, 利用 RNN 架构维护每个节点的嵌入. SDGNN[24]采用 RNN 维护节点隐状态, 由更新、传播组件构成: 更新组件更新交互节点隐状态, 传播组件将更新传递至相邻节点, 通过更新-传播步骤维持网络节点隐状态. 这类方法的弱点在于随着节点交互序列长度的增加, 它们会遇到梯度爆炸/消失的问题, 导致预测结果不尽如人意.

(4) 基于注意力机制的方法. 这类方法能够有效建模节点之间的时空依赖关系, 通过计算节点之间的注意力权重, 自动学习节点之间的相互作用强度, 从而更好地捕捉节点在不同时间步之间的关联性. 例如, TGAT[12]提出时态图注意力层, 可以高效聚合时态拓扑邻域特征, 并学习时间特征的相互作用. CMOD[13]提出连续时间多层图表示框架, 以精细粒度捕获节点状态的连续演化并自适应地挖掘图中多尺度空间依赖. 然而, 这类方法主要专注于解决短序列依赖关系, 性能经常随着序列长度的增加而下降.

(5) 基于 Transformer 的方法. 这类方法通过利用 Transformer 架构的并行处理能力, 能够有效处理大规模动态图中的时序数据. 例如, SimpleDyG[14]采用轻量级节点表示更新机制, 依据时间远近区分历史交互影响力, 赋予近期关键交互更高权重, 高效捕获时序信息与节点交互模式, 兼顾效果与计算开销. DyGFormer[15]通过引入 Transformer 层和邻居共现编码, 显式建模节点间的相关性, 使得模型能够高效捕获动态图结构. TCL[16]通过结合 Transformer 层与对比学习策略, 精准捕捉节点时序结构关联. 但是, 随着序列长度的增加, Transformer 层的计算复杂度呈二次方增长, 导致计算代价高昂.

除此之外, 还有一些方法基于多层感知机[25]、常微分方程[26]或者哈希映射[27]来建模连续时间动态图, 但是这些方法依然存在无法高效捕获序列长期时间依赖的问题.

2.2 选择性状态空间模型

状态空间模型是一类用于描述系统动态行为的数学模型, 其特点是通过状态变量来刻画系统的内部状态, 并通过状态转移方程和观测方程来描述状态随时间和观测值之间的关系. 状态空间模型在处理具有时间依赖性和不确定性的复杂系统时表现尤其出色. 然而, 传统状态空间模型具有线性时间不变性, 即它们的参数既不依赖于输入, 也不随时间发生变化, 这限制了模型捕获序列数据时序信息的能力.

为了解决这一问题, 研究者们提出了很多不同的结构化状态空间模型[18,28−30], 其中最具代表性的是选择性状态空间模型[18]. 该模型通过引入一种选择性机制, 将时序输入向量 $\boldsymbol{z}(t)$ 经由隐藏状态 $\boldsymbol{\phi}(t)$, 线性映射为输出向量 $\boldsymbol{h}(t)$:

$$\boldsymbol{\phi}(t)=\tilde{\boldsymbol{A}}(t)\boldsymbol{\phi}(t-1)+\tilde{\boldsymbol{B}}(t)\boldsymbol{z}(t) \tag{1}$$

$$\boldsymbol{h}(t)=\boldsymbol{C}(t)\boldsymbol{\phi}(t) \tag{2}$$

其中, $\boldsymbol{C}(t)$ 为输出状态映射矩阵, 用于把状态映射到输出. $\tilde{\boldsymbol{B}}(t)$ 为离散化的输入状态映射矩阵, 用于把输入映射到状态. $\tilde{\boldsymbol{A}}(t)$ 为离散化的状态转移矩阵. 离散化的目的在于处理时间不连续的输入序列, 它们的计算公式为:

$$\tilde{\boldsymbol{A}}(t)=\mathrm{e}^{\Delta(t)\boldsymbol{A}} \tag{3}$$

$$\tilde{\boldsymbol{B}}(t)=(\Delta(t)\boldsymbol{A})^{-1}(\mathrm{e}^{\Delta(t)\boldsymbol{A}}-\boldsymbol{I})\cdot\Delta(t)\boldsymbol{B}(t) \tag{4}$$

公式 (3) 和 (4) 中, $\boldsymbol{A}$ 为初始的状态转移矩阵, 用于定义状态变化的基本规则, $\boldsymbol{I}$ 为单位矩阵, $\Delta(t)$ 为选择性因子, 也称为时间尺度参数, 用于控制每个时间步对应状态的更新程度. $\boldsymbol{B}(t)$ 为离散化之前的输入状态映射矩阵, 随着输入而发生变化.

$$\Delta(t)=\tau_{\Delta}(\mathrm{Linear}_{\Delta}(\boldsymbol{z}(t))),\ \boldsymbol{B}(t)=\mathrm{Linear}_{\boldsymbol{B}}(\boldsymbol{z}(t)),\ \boldsymbol{C}(t)=\mathrm{Linear}_{\boldsymbol{C}}(\boldsymbol{z}(t)) \tag{5}$$

其中, τ_{Δ} 为激活函数, Linear_{Δ}、$\mathrm{Linear}_{\boldsymbol{B}}$、$\mathrm{Linear}_{\boldsymbol{C}}$ 分别表示不同的线性映射函数.

由此可见, 选择性状态空间模型根据不同时刻的输入向量 $\boldsymbol{z}(t)$ 动态调整状态变量, 选择性地保留或忽略序列中的时序信息, 确保模型能够随着输入数据的顺序而不断演化, 因而适合对长序列时间依赖进行建模.

3 基于双向选择性状态空间模型的连续时间动态图表示学习

我们提出的模型 DG-BS^{3}M 主要由两部分组成: 第 1 部分为特征提取和融合, 即从节点历史交互序列中提取多种不同类型的嵌入表示, 包括节点嵌入表示、边嵌入表示、时间嵌入表示和邻居共现频率嵌入表示, 并对它们进行分块拼接和维度对齐. 第 2 部分为捕获序列长期双向依赖关系, 通过双向交互序列表达更新模块, 对节点的联合序列嵌入表示进行双向编码, 揭示潜在的非因果关系, 最后生成具有时序信息的节点表示, 用于动态链接预测. 图 1 演示了 DG-BS^{3}M 的具体架构组成.

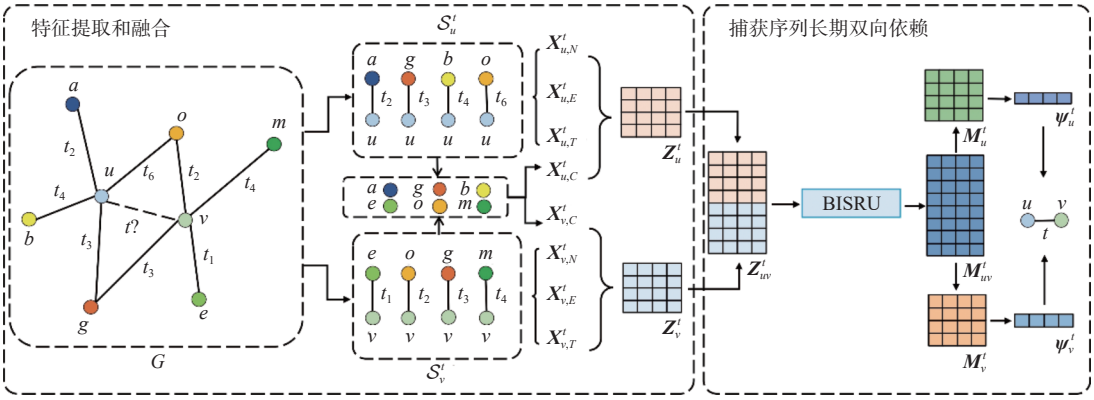

图 1 DG-BS^{3}M 的网络架构

3.1 问题描述

一个连续时间动态图$\mathcal{G}$可以表示为特定节点对之间相互作用的时间序列: $\mathcal{G}=[(u_0,v_0,t_0),(u_1,v_1,t_1),\ldots,(u_T,v_T,t_T)]$. 其中$t_i$ $(0\leqslant i\leqslant T)$表示时间戳, 时间戳的顺序为$0\leqslant t_0\leqslant t_1\leqslant\ldots\leqslant t_T$, $u_i,v_i\in\mathcal{N}$表示在时间戳t_i产生交互的源节点和目标节点, $\mathcal{N}$表示所有节点集合. 每个节点都有属于自己的节点特征$\boldsymbol{x}_{u_i},\boldsymbol{x}_{v_i}\in\mathbb{R}^{1\times d_N}$, 每次交互$(u_i,v_i,t_i)\in\mathcal{E}$生成的链接都有属于自己的边特征$\boldsymbol{e}^{t_i}_{u_i,v_i}\in\mathbb{R}^{1\times d_E}$, 其中$\mathcal{E}$表示所有边的集合, d_N和d_E分别表示节点和边的特征维度.

连续时间动态图表示学习旨在从节点u_i,v_i各自的历史交互信息中学习出能够反映它们动态演化特性的嵌入表示$\boldsymbol{h}_{u_i},\boldsymbol{h}_{v_i}\in\mathbb{R}^{1\times d}$, d表示统一的嵌入维度, 并将所学习到的节点表示应用于各种下游任务. 本文使用最常见的动态链接预测任务来验证所提方法的有效性.

3.2 特征提取和融合

3.2.1 节点、边、时间嵌入表示

在连续时间动态图中, 按照时间先后顺序依次采样源节点u和目标节点v在预测时刻t之前它们各自交互过的一阶邻居节点, 以此构建节点u和v的历史交互序列, 记为$\mathcal{S}^t_*=[(s^t_{*,1},t_1),(s^t_{*,2},t_2),\ldots,(s^t_{*,L},t_L)]$, 其中$s^t_{*,i}$ $(0\leqslant i\leqslant L)$表示节点$u$或$v$按照时间顺序交互过的第$i$个邻居节点, 每个节点都有给定的节点特征(或嵌入)$\boldsymbol{x}^t_{*,N,i}$和边特征(或嵌入)$\boldsymbol{x}^t_{*,E,i}$. 为了确保节点u和v的历史交互序列长度同为指定长度L, 规定$\mathcal{S}^t_*$缺少的交互节点都用缺省值填充, 其对应的节点嵌入和边嵌入表示用零向量填充. 我们将序列$\mathcal{S}^t_*$的节点和边嵌入表示分别记为$\boldsymbol{X}^t_{*,N}=[\boldsymbol{x}^t_{*,N,1};\boldsymbol{x}^t_{*,N,2};\ldots;\boldsymbol{x}^t_{*,N,L}]\in\mathbb{R}^{L\times d_N}$和$\boldsymbol{X}^t_{*,E}=[\boldsymbol{x}^t_{*,E,1};\boldsymbol{x}^t_{*,E,2};\ldots;\boldsymbol{x}^t_{*,E,L}]\in\mathbb{R}^{L\times d_E}$.

对于序列中第i次交互的时间戳t_i, 使用$\boldsymbol{x}^t_{*,T,i}=[\cos(w_1\Delta t_i),\sin(w_1\Delta t_i),\ldots,\cos(w_{d_T}\Delta t_i),\sin(w_{d_T}\Delta t_i)]/\sqrt{d_T}$进行编码, 以学习节点潜在的周期性的交互模式, 其中$\Delta t_i=t-t_i$表示当前交互的发生时间t_i相较于预测时刻t的间隔, $w_1,\ldots,w_{d_T}$表示可训练的权重参数, d_T表示序列时间嵌入表示的维度. 我们将序列$\mathcal{S}^t_*$的时间嵌入表示记为$\boldsymbol{X}^t_{*,T}=[\boldsymbol{x}^t_{*,T,1};\boldsymbol{x}^t_{*,T,2};\ldots;\boldsymbol{x}^t_{*,T,L}]\in\mathbb{R}^{L\times d_T}$.

3.2.2 邻居共现频率嵌入表示

为了反映节点u和v之间的相关性, 我们通过统计每个邻居在序列$\mathcal{S}^t_u$和$\mathcal{S}^t_v$中的出现次数来反映u和v之间的相关性. 如果节点u和v具有更多的共同历史邻居, 那么它们在未来发生相互作用的可能性就更大. 为此, 对于序列$\mathcal{S}^t_u$和$\mathcal{S}^t_v$中的每个邻居, 我们分别统计其在$\mathcal{S}^t_u$和$\mathcal{S}^t_v$中的出现频率.

如图2所示, 假设序列$\mathcal{S}^t_u$和$\mathcal{S}^t_v$对应的节点序列分别为$[a,g,b,o]$和$[e,o,g,m]$, 节点a、g、b、e、o、m在序列$\mathcal{S}^t_u$和$\mathcal{S}^t_v$中出现的频率分别为$\left[\frac{1}{4},0\right]$、$\left[\frac{1}{4},\frac{1}{4}\right]$、$\left[\frac{1}{4},0\right]$、$\left[0,\frac{1}{4}\right]$、$\left[\frac{1}{4},\frac{1}{4}\right]$、$\left[0,\frac{1}{4}\right]$, 则节点$u$和$v$的邻居共现频率分别表示为$\boldsymbol{C}^t_u=\left[\left[\frac{1}{4},0\right];\left[\frac{1}{4},\frac{1}{4}\right];\left[\frac{1}{4},0\right];\left[\frac{1}{4},\frac{1}{4}\right]\right]$, $\boldsymbol{C}^t_v=\left[\left[0,\frac{1}{4}\right];\left[\frac{1}{4},\frac{1}{4}\right];\left[\frac{1}{4},\frac{1}{4}\right];\left[0,\frac{1}{4}\right]\right]$. 然后, 应用一个函数$f:\mathbb{R}^{L\times 1}\to\mathbb{R}^{L\times d_C}$来编码邻居共现频率:

$$\boldsymbol{X}^t_{*,C}=f(\boldsymbol{C}^t_*[:,1])+f(\boldsymbol{C}^t_*[:,2]) \tag{6}$$

其中, $\boldsymbol{X}^t_{*,C}=[\boldsymbol{x}^t_{*,C,1};\boldsymbol{x}^t_{*,C,2};\ldots;\boldsymbol{x}^t_{*,C,L}]\in\mathbb{R}^{L\times d_C}$, $*$表示节点u或v, d_C为邻居共现频率嵌入表示的维度.

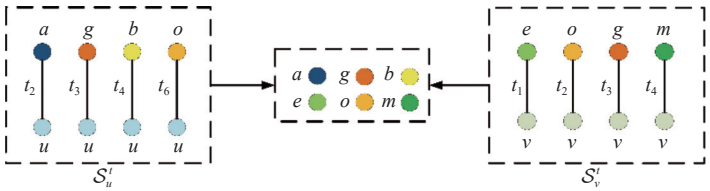

图2 节点交互序列中的历史邻居

3.2.3 分块拼接

对于长序列而言, 直接将整个序列嵌入表示输入模型会导致计算量巨大. 尽管选择性状态空间模型具有线性时间复杂度, 我们进一步采用分块技术来降低计算复杂度. 假设我们把序列分成P块, 每个小块由时间上邻近的节

点嵌入表示组成. 以 $\boldsymbol{X}_{u,N}^{t} \in \mathbb{R}^{L \times d_N}$ 的拼接为例, 将分割的小块横向拼接, 得到新的嵌入表示记为 $\boldsymbol{M}_{u,N}^{t} \in \mathbb{R}^{l_u^t \times d_N P}$, 其中 $l_u^t = l_v^t = L/P$. 由此我们可以得到序列 $\mathcal{S}_*^t$ 的分块拼接后的各种嵌入表示 $\boldsymbol{M}_{*,N}^{t} \in \mathbb{R}^{l_*^t \times d_N P}$, $\boldsymbol{M}_{*,E}^{t} \in \mathbb{R}^{l_*^t \times d_E P}$, $\boldsymbol{M}_{*,T}^{t} \in \mathbb{R}^{l_*^t \times d_T P}$ 和 $\boldsymbol{M}_{*,C}^{t} \in \mathbb{R}^{l_*^t \times d_C P}$, $* \in \{u, v\}$, 如图 3 所示.

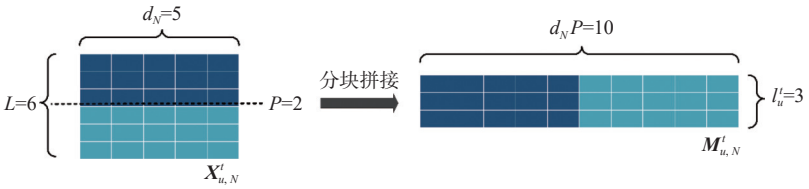

图 3 分块拼接示意

3.2.4 维度对齐

我们将 $\mathcal{S}_u^t$ 和 $\mathcal{S}_v^t$ 的各种嵌入表示对齐到指定的统一维度 d, 得到 $\boldsymbol{Z}_{u,\#}^{t} \in \mathbb{R}^{l_u^t \times d}$ 和 $\boldsymbol{Z}_{v,\#}^{t} \in \mathbb{R}^{l_v^t \times d}$. 其中 $\# \in \{E, T, C, N\}$. 对齐方式为:

$$\boldsymbol{Z}_{u,\#}^{t} = \boldsymbol{M}_{u,\#}^{t} \boldsymbol{W}_{\#} + \boldsymbol{b}_{\#},\ \boldsymbol{Z}_{v,\#}^{t} = \boldsymbol{M}_{v,\#}^{t} \boldsymbol{W}_{\#} + \boldsymbol{b}_{\#} \tag{7}$$

其中, $\boldsymbol{W}_{\#} \in \mathbb{R}^{d_{\#} P \times d}$ 和 $\boldsymbol{b}_{\#} \in \mathbb{R}^{1 \times d}$ 是可训练的权重.

然后我们将节点 u 和 v 对齐后的各种嵌入表示进行拼接, 得到 $\boldsymbol{Z}_u^t = \boldsymbol{Z}_{u,N}^{t} \| \boldsymbol{Z}_{u,E}^{t} \| \boldsymbol{Z}_{u,T}^{t} \| \boldsymbol{Z}_{u,C}^{t} \in \mathbb{R}^{l_u^t \times 4d}$, $\boldsymbol{Z}_v^t = \boldsymbol{Z}_{v,N}^{t} \| \boldsymbol{Z}_{v,E}^{t} \| \boldsymbol{Z}_{v,T}^{t} \| \boldsymbol{Z}_{v,C}^{t} \in \mathbb{R}^{l_v^t \times 4d}$.

3.3 捕获序列长期双向依赖关系

3.3.1 双向交互序列表达更新模块

捕获序列的长期双向依赖关系主要依赖于 DG-BS^{3}M 中的双向交互序列表达更新 (bidirectional interactive sequence representation update, BISRU) 模块, 其内部结构如图 4 所示.

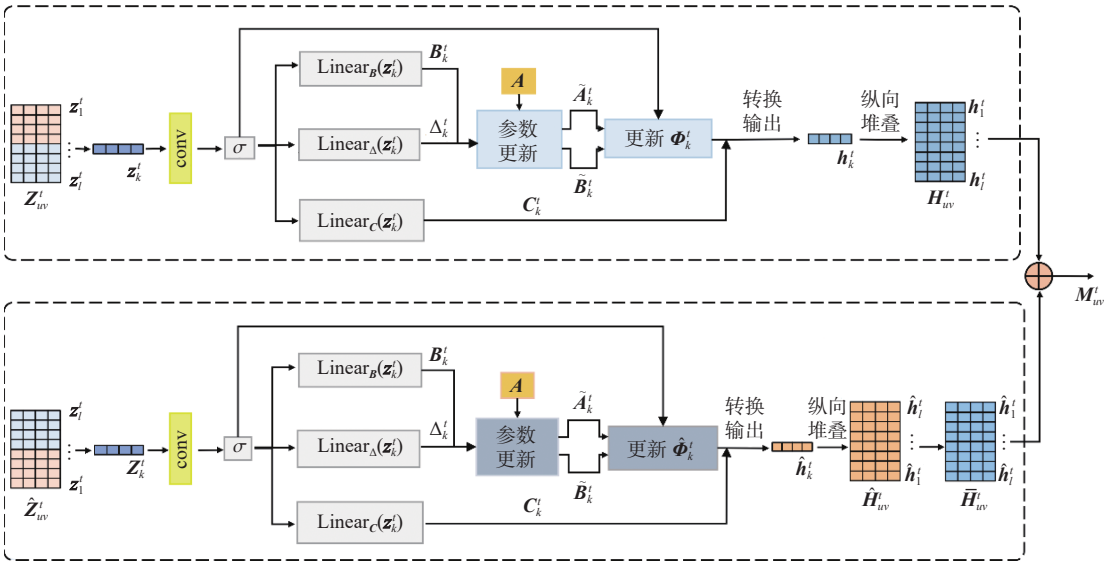

图 4 双向交互序列表达更新 (BISRU) 模块

首先, 我们对序列 $\mathcal{S}_u^t$ 和 $\mathcal{S}_v^t$ 的嵌入表示 $\boldsymbol{Z}_u^t$、$\boldsymbol{Z}_v^t$ 进行拼接操作, 得到序列 $\mathcal{S}_u^t$、$\mathcal{S}_v^t$ 的联合嵌入表示 $\boldsymbol{Z}_{uv}^t = [\boldsymbol{Z}_u^t; \boldsymbol{Z}_v^t] \in \mathbb{R}^{l \times 4d}$, 其中 $l = l_u^t + l_v^t$.

然后, 我们使用 BISRU 模块中的双向选择性状态空间模型对 $\boldsymbol{Z}_{uv}^t$ 进行编码, 以捕获节点 u 和 v 各自序列内部

及序列之间的长期双向依赖关系. 相比于传统方法使用时序网络仅对序列数据正向编码, 双向选择性状态空间模型使用选择性状态空间模型对序列联合嵌入表示 $\boldsymbol{Z}_{uv}^t$ 进行正向和反向编码并融合, 通过从两个角度提取序列特征, 减少单一视角的偏差, 从而增强模型对序列中隐藏的非因果关系的挖掘能力和对复杂时序关系的建模能力. 在正向编码过程中, 双向选择性状态空间模型按照时间顺序正向依次处理序列联合嵌入表示 $\boldsymbol{Z}_{uv}^t = [\boldsymbol{z}_1^t; \boldsymbol{z}_2^t; \ldots; \boldsymbol{z}_l^t]$, 令 $\boldsymbol{z}_k^t \in \mathbb{R}^{1\times 4d}$ $(1 \leqslant k \leqslant l)$ 表示 $\boldsymbol{Z}_{uv}^t$ 的第 k 行向量, 我们将 $\boldsymbol{z}_k^t$ 进行一维卷积预处理 (维度保持不变) 后按照 k 增序依次输入选择性状态空间模型, 生成包含长期正向时间依赖信息的状态表示 $\boldsymbol{h}_k^t \in \mathbb{R}^{1\times 4d}$:

$$\boldsymbol{\Phi}_k^t = \tilde{\boldsymbol{A}}_k^t \boldsymbol{\Phi}_{k-1}^t + \tilde{\boldsymbol{B}}_k^t \boldsymbol{z}_k^t \tag{8}$$

$$\boldsymbol{h}_k^t = \boldsymbol{C}_k^t \boldsymbol{\Phi}_k^t \tag{9}$$

$$\tilde{\boldsymbol{A}}_k^t = \mathrm{e}^{\Delta_k^t \boldsymbol{A}} \tag{10}$$

$$\tilde{\boldsymbol{B}}_k^t = (\Delta_k^t \boldsymbol{A})^{-1}(\mathrm{e}^{\Delta_k^t \boldsymbol{A}} - \boldsymbol{I}) \cdot \Delta_k^t \boldsymbol{B}_k^t \tag{11}$$

$$\boldsymbol{A}_k^t = \mathrm{Linear}_{\boldsymbol{A}}(\boldsymbol{z}_k^t),\ \boldsymbol{B}_k^t = \mathrm{Linear}_{\boldsymbol{B}}(\boldsymbol{z}_k^t),\ \Delta_k^t = \tau_\Delta(\mathrm{Linear}_\Delta(\boldsymbol{z}_k^t)) \tag{12}$$

其中, $\boldsymbol{\Phi}_k^t \in \mathbb{R}^{r\times 4d}$ 表示 $\boldsymbol{z}_k^t$ 在正向序列中对应的隐藏状态矩阵, r 是隐藏状态的数量, 初始值 $\boldsymbol{\Phi}_0^t = \mathbf{O}$ ($\mathbf{O}$ 表示零矩阵). $\boldsymbol{B}_k^t \in \mathbb{R}^{r\times 1}$ 和 $\boldsymbol{C}_k^t \in \mathbb{R}^{1\times r}$ 分别为依赖于输入 $\boldsymbol{z}_k^t$ 的输入状态映射矩阵和输出状态映射矩阵. $\tilde{\boldsymbol{B}}_k^t \in \mathbb{R}^{r\times 1}$ 则是离散化的输入状态映射矩阵, 用于处理时间 t 不连续的输入. $\Delta_k^t \in \mathbb{R}$ 控制着模型对当前输入 $\boldsymbol{z}_k^t$ 的交互信息遗忘程度, 使得模型能够动态调整对不同交互信息的关注程度, 避免无关信息的干扰. $\tilde{\boldsymbol{A}}_k^t \in \mathbb{R}^{r\times r}$ 是 $\boldsymbol{z}_k^t$ 对应的离散化的状态转移矩阵, $\boldsymbol{A} \in \mathbb{R}^{r\times r}$ 则为初始的状态转移矩阵. 在设计 $\boldsymbol{A}$ 时, 我们基于 Gu 等人[31]提出的高阶多项式投影算子理论, 将其定义为一个 HiPPO 矩阵:

$$a_{ij} = \begin{cases} -\sqrt{(2i+1)(2j+1)}, & i > j \\ -(i+1), & i = j \\ 0, & i < j \end{cases} \tag{13}$$

这种设计的优点是通过正交多项式投影, 使得模型在更新隐藏状态的过程中无需对所有历史信息进行遍历处理, 从而避免了随着序列长度的增加导致计算量显著增加的难题. 通过纵向堆叠对应的输出向量 $\boldsymbol{h}_k^t$, 我们得到正向的序列状态表示 $\boldsymbol{H}_{uv}^t = [\boldsymbol{h}_1^t; \boldsymbol{h}_2^t; \ldots; \boldsymbol{h}_l^t] \in \mathbb{R}^{l\times 4d}$.

另一方面, 在逆向编码过程中, 双向选择性状态空间模型按照逆向时间顺序依次处理序列联合嵌入表示 $\hat{\boldsymbol{Z}}_{uv}^t = [\boldsymbol{z}_l^t; \boldsymbol{z}_{l-1}^t; \ldots; \boldsymbol{z}_1^t]$. 为此, 我们将 $\boldsymbol{z}_k^t$ 进行一维卷积预处理后按 k 降序依次输入选择性状态空间模型, 生成包含长期逆向时间依赖信息的状态表示 $\hat{\boldsymbol{h}}_k^t \in \mathbb{R}^{1\times 4d}$:

$$\hat{\boldsymbol{\Phi}}_k^t = \tilde{\boldsymbol{A}}_k^t \hat{\boldsymbol{\Phi}}_{k+1}^t + \tilde{\boldsymbol{B}}_k^t \boldsymbol{z}_k^t \tag{14}$$

$$\hat{\boldsymbol{h}}_k^t = \boldsymbol{C}_k^t \hat{\boldsymbol{\Phi}}_k^t \tag{15}$$

其中, $\hat{\boldsymbol{\Phi}}_k^t \in \mathbb{R}^{r\times 4d}$ 表示 $\boldsymbol{z}_k^t$ 在逆向序列中对应的隐藏状态矩阵, 初始值 $\hat{\boldsymbol{\Phi}}_{l+1}^t = \mathbf{O}$. 接着, 我们将得到的逆向序列嵌入表示 $\hat{\boldsymbol{H}}_{uv}^t = [\hat{\boldsymbol{h}}_l^t; \hat{\boldsymbol{h}}_{l-1}^t; \ldots; \hat{\boldsymbol{h}}_1^t] \in \mathbb{R}^{l\times 4d}$ 进行上下翻转, 得到其正序排列:

$$\overrightarrow{\boldsymbol{H}}_{uv}^t = \mathrm{Flip}(\hat{\boldsymbol{H}}_{uv}^t) \tag{16}$$

最后将双向序列嵌入表示 $\overrightarrow{\boldsymbol{H}}_{uv}^t$ 和 $\boldsymbol{H}_{uv}^t$ 相融合, 生成信息更丰富的序列双向嵌入表示 $\boldsymbol{M}_{uv}^t$:

$$\boldsymbol{M}_{uv}^t = \overrightarrow{\boldsymbol{H}}_{uv}^t + \boldsymbol{H}_{uv}^t \tag{17}$$

从而更全面地捕捉序列的长期双向依赖关系.

3.3.2 生成具有时序信息的节点表示

我们将 $\boldsymbol{M}_{uv}^t$ 重新划分为序列 $\mathcal{S}_u^t$ 和 $\mathcal{S}_v^t$ 各自的嵌入表示 $\boldsymbol{M}_u^t \in \mathbb{R}^{l_u^t\times 4d}$ 和 $\boldsymbol{M}_v^t \in \mathbb{R}^{l_v^t\times 4d}$:

$$\boldsymbol{M}_u^t = \boldsymbol{M}_{uv}^t[1 : l_u^t, :] \tag{18}$$

$$\boldsymbol{M}_v^t = \boldsymbol{M}_{uv}^t[l_u^t + 1 : l_u^t + l_v^t, :] \tag{19}$$

然后, 分别对序列 $\mathcal{S}_u^t$ 和 $\mathcal{S}_v^t$ 的嵌入表示 $\boldsymbol{M}_u^t$、$\boldsymbol{M}_v^t$ 进行平均聚合, 从而得到节点 u 和 v 最终的嵌入表示行向量

$\boldsymbol{\psi}_u^t \in \mathbb{R}^{d_{\text{out}}}$ 和 $\boldsymbol{\psi}_v^t \in \mathbb{R}^{d_{\text{out}}}$:

$$\boldsymbol{\psi}_u^t = Mean(\boldsymbol{M}_u^t)\boldsymbol{W}_{\text{out}} + \boldsymbol{b}_{\text{out}} \tag{20}$$

$$\boldsymbol{\psi}_v^t = Mean(\boldsymbol{M}_v^t)\boldsymbol{W}_{\text{out}} + \boldsymbol{b}_{\text{out}} \tag{21}$$

其中, $\boldsymbol{W}_{\text{out}} \in \mathbb{R}^{4d\times d_{\text{out}}}$ 为权重矩阵, $\boldsymbol{b}_{\text{out}} \in \mathbb{R}^{d_{\text{out}}}$ 为偏置行向量, $Mean$ 表示行向量平均聚合函数, d_{out} 表示节点 u 和 v 嵌入表示最终输出的维度.

3.4 动态链接预测

为了预测节点 u 和 v 在 t 时刻产生交互的概率, 我们把 $\boldsymbol{\psi}_u^t$ 和 $\boldsymbol{\psi}_v^t$ 进行联合, 然后使用多层感知机 (MLP) 计算如下公式:

$$\hat{y}_{uv} = Softmax(MLP(ReLU(MLP(\boldsymbol{\psi}_u^t \| \boldsymbol{\psi}_v^t)))) \tag{22}$$

其中, $\hat{y}_{uv}$ 为预测概率, $Softmax$ 和 $ReLU$ 为激活函数.

在训练阶段, 我们采用二元交叉熵损失函数用于衡量动态链接预测的损失:

$$\mathcal{L}_p = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log\hat{y}_i + (1-y_i)\cdot\log(1-\hat{y}_i)) \tag{23}$$

其中, n 为预测边数, $\hat{y}_i$ 为第 i 条边的预测链接概率, $y_i \in \{0,1\}$ 为真实标签, 0 表示未产生交互, 1 表示产生交互.

3.5 时间和空间复杂度分析

在时间复杂度方面, DG-BS^{3}M 在特征提取和融合阶段生成节点、边、时间和邻居共现频率嵌入表示的时间复杂度为 $O(nL(d_N+d_E+d_T+d_C))$, 分块拼接的时间复杂度为 $O(nP)$, 维度对齐的时间复杂度为 $O(nL(d_N+d_E+d_T+d_C)d)$. 由于 $d_N = d_E \gg d_T \gg d_C \gg P$, 则特征提取和融合的总时间复杂度为 $O(nLd_Nd)$. 在捕获序列长期双向依赖阶段, 对节点交互序列进行双向编码的时间复杂度为 $O(nL(rd^2+r^2d)$, 生成节点嵌入表示的时间复杂度为 $O(nLd^2d_{\text{out}})$. 因为 $d_{\text{out}} \gg r$ 且 $d \gg r$, 则 DG-BS^{3}M 在捕获序列长期双向依赖阶段的时间复杂度为 $O(nLd^2d_{\text{out}})$. 令 R 表示训练的轮数, 则 DG-BS^{3}M 在训练阶段的时间复杂度为 $O(RnLd(d_N+dd_{\text{out}}))$. 在测试阶段, 假设预测的边数为 n_t, DG-BS^{3}M 的时间复杂度为 $O(n_tLd(d_N+dd_{\text{out}}))$.

在空间复杂度方面, 假设单个批次处理的边数为 n_b, DG-BS^{3}M 在特征提取和融合阶段的空间复杂度为 $O(n_bL(d_N+d_E+d_T+d_C)) = O(n_bLd_N)$. 在捕获序列长期双向依赖阶段, 双向交互序列表达更新模块的空间复杂度为 $O(n_b(rd+r^2+Ld)) = O(n_bLd)$, 生成节点嵌入的空间复杂度为 $O(dd_{\text{out}})$. 因为 $d_N \gg d$, 所以 DG-BS^{3}M 的总空间复杂度为 $O(n_bLd_N)$.

已知当前经典连续时间动态图表示学习方法[10,12,14,15]需要的时间复杂度和空间复杂度都与序列长度 L 的平方成正比. 与它们相比, 我们的方法 DG-BS^{3}M 将时间复杂度和空间复杂度都降低至与序列长度线性相关, 因而在对较长的序列进行建模时, 具有计算效率更高, 占用计算资源更少的优势.

3.6 收敛性分析

由定理 1 可知, DG-BS^{3}M 的双向选择性状态空间机制具备指数收敛特性.

定理 1. 双向选择性状态空间模型 (DG-BS^{3}M) 生成的状态序列满足指数收敛.

证明: 假设状态空间 $\mathcal{X} \subseteq \mathbb{R}^{r\times 4d}$ 是完备度量空间, 度量 $d(\boldsymbol{\Phi}_1,\boldsymbol{\Phi}_2) = \|\boldsymbol{\Phi}_1-\boldsymbol{\Phi}_2\|_2$, 且 $\exists M>0$, 对 $\forall\boldsymbol{\Phi}\in\mathcal{X}$, 有 $\|\boldsymbol{\Phi}\|_2 \leqslant M$. 定义算子 $T\colon \mathcal{X}\to\mathcal{X}$ 为:

$$T(\boldsymbol{\Phi}) = \tilde{\boldsymbol{A}}\boldsymbol{\Phi} + \tilde{\boldsymbol{B}}\boldsymbol{z} \tag{24}$$

其中, $\tilde{\boldsymbol{A}} = \mathrm{e}^{\Delta\boldsymbol{A}}$, $\boldsymbol{A}$ 为 HiPPO 矩阵 (见公式 (13)). 两个状态经 T 映射后的距离为:

$$d(T(\boldsymbol{\Phi}),T(\boldsymbol{\Phi}')) = \left\|(\tilde{\boldsymbol{A}}\boldsymbol{\Phi}+\tilde{\boldsymbol{B}}\boldsymbol{z})-(\tilde{\boldsymbol{A}}\boldsymbol{\Phi}'+\tilde{\boldsymbol{B}}\boldsymbol{z})\right\|_2 = \left\|\tilde{\boldsymbol{A}}(\boldsymbol{\Phi}-\boldsymbol{\Phi}')\right\|_2 \leqslant \left\|\tilde{\boldsymbol{A}}\right\|_2\cdot\|\boldsymbol{\Phi}-\boldsymbol{\Phi}'\|_2 \tag{25}$$

已知 HiPPO 矩阵 $\boldsymbol{A}$ 的特征值为其对角线元素, 即 $-1,-2,\ldots,-r$, 因此 $\tilde{\boldsymbol{A}}$ 的特征值 $\mathrm{e}^{-k\Delta}$ ($k=1,2,\ldots,r$) 在 $(0,1)$ 区间内. 故存在 $\lambda\in(0,1)$, 使得 $\left\|\tilde{\boldsymbol{A}}\right\|_2 \leqslant \lambda$, 即有:

$$d(T(\boldsymbol{\Phi}),T(\boldsymbol{\Phi}'))\leqslant\lambda\cdot d(\boldsymbol{\Phi},\boldsymbol{\Phi}') \tag{26}$$

故 T 是压缩映射.

由 Banach 不动点定理可知: 存在唯一 $\boldsymbol{\Phi}^*\in\mathcal{X}$, 使得 $T(\boldsymbol{\Phi}^*)=\boldsymbol{\Phi}^*$, 即存在唯一稳态. 对任意 $k>0$, 由数学归纳法易得:

$$\left\|\boldsymbol{\Phi}_k^t-\boldsymbol{\Phi}^*\right\|_2=\left\|T(\boldsymbol{\Phi}_{k-1}^t)-T(\boldsymbol{\Phi}^*)\right\|_2\leqslant\lambda^k\left\|\boldsymbol{\Phi}_0^t-\boldsymbol{\Phi}^*\right\|_2 \tag{27}$$

因 $\lambda\in(0,1)$, 当 $k\to\infty$ 时, $\lambda^k\to 0$, 故 $\boldsymbol{\Phi}_k^t\to\boldsymbol{\Phi}^*$, 说明序列 $(\boldsymbol{\Phi}_0^t,\boldsymbol{\Phi}_1^t,\ldots,\boldsymbol{\Phi}_k^t)$ 满足指数收敛. 同理可证, 反向序列 $(\hat{\boldsymbol{\Phi}}_k^t,\hat{\boldsymbol{\Phi}}_{k-1}^t,\ldots,\hat{\boldsymbol{\Phi}}_0^t)$ 亦满足指数收敛. 综上所述, DG-BS^{3}M 生成的状态序列满足指数收敛. 证毕.

4 实验分析

在本节中, 我们使用了 8 个公开的真实世界的数据集, 将提出的 DG-BS^{3}M 方法与其他基线方法在动态链接预测任务进行性能比较, 从而验证模型的有效性.

4.1 数据集

我们使用的 8 个真实世界数据集[32], 具体信息如下.

(1) Wikipedia: 一个二分交互网络, 包含了一个月内维基百科页面上的编辑信息. 节点表示用户和页面, 链接表示带有时间戳的编辑行为. 每个链接都具有 172 维的语言查询和词数统计 (LIWC) 特征.

(2) Reddit: 一个二分的交互网络, 记录了用户在 subreddits 中一个月的发帖情况. 用户和帖子是节点, 链接是带有时间戳的发帖请求. 每个链接具有 172 维的 LIWC 特征.

(3) MOOC: 一个在线的二分交互网络, 其中节点为学生和课程内容单元 (例如视频和问题集). 每个链接表示学生对特定内容单元的访问行为, 并具有 4 维特征.

(4) LastFM: 一个二分的交互网络, 由用户在一个月内收听的歌曲的信息组成. 用户和歌曲是节点, 链接表示用户的收听行为.

(5) Enron: 记录了 ENRON 能源公司员工之间 3 年多的电子邮件通信.

(6) Social Evo: 一个移动电话近场网络, 其对所有本科生宿舍的日常活动进行了为期 8 个月的监控, 其中每个链接都具有 2 维特征.

(7) UCI: 一个在线通信网络, 其中节点是高校学生, 链接是学生发布的消息.

(8) Can Parl: 一个动态的政治网络, 记录了 2006–2019 年加拿大议会中议员之间的互动. 每个节点代表一个选区的议员, 当两个议员都对一个议案投赞成票时, 就创建一个链接. 每个链接的权重是一个议员在一年中对另一个议员投赞成票的次数.

表 1 列举了所有数据集的统计信息. 其中包含了每个数据集的节点数、链接边数、采样持续时间、时间步数量和时间粒度. 我们将这些数据集按照 70%:15%:15% 的比例划分为训练集、验证集和测试集.

表 1 数据集详细信息

数据集	节点数	链接数	持续时间	时间步	时间粒度
Wikipedia	9227	157474	1 month	152757	Unix timestamps
Reddit	10984	672447	1 month	669065	Unix timestamps
MOOC	7144	411749	17 months	345600	Unix timestamps
LastFM	1980	1293103	1 month	1283614	Unix timestamps
Enron	184	125235	3 years	22632	Unix timestamps
Social Evo	74	2099519	8 months	565932	Unix timestamps
UCI	1899	59835	196 days	58911	Unix timestamps
Can Parl	734	74478	14 years	14	years

4.2 基线方法

(1) DyRep[9]: 一种基于 RNN 的方法, 在每次交互时更新节点状态, 包含一个时间注意力聚合模块, 以考虑动态

图中随时间演化的结构信息.

(2) TGAT[12]: 一种基于自注意力机制的方法, 通过聚合每个节点的时间拓扑邻居的特征来计算节点表示. 其还具有时间编码功能, 用于捕获时间模式.

(3) CAWN[6]: 一种基于时间随机游走的方法, 通过使用时间随机游走提取时间网络模体来表示网络动态, 将模体输入 RNN 中编码单个游走表示, 再通过自注意力机制聚合多游走表示为单向量, 用于下游任务.

(4) TGN[10]: 一种基于 RNN 和自注意力机制的混合方法. 它利用存储模块存储和更新每个节点的内存状态, 同时通过嵌入模块生成节点的时序表示.

(5) TCL[16]: 一种基于 Transformer 的方法, 先对时序依赖交互子图做广度优先搜索生成节点交互序列, 再用融合图拓扑与时间信息的图转换器学习节点表示, 还通过交叉注意力操作建模交互节点间的相互依赖关系.

(6) GraphMixer[25]: 一种简单的基于图卷积的架构, 使用固定时间的编码函数, 并将其纳入基于 MLP-Mixer 的链接编码器中, 从时序链接中学习, 并使用基于均值池化的节点编码器来聚合节点特征.

(7) CTAN[26]: 一种基于非耗散常微分方程 (ordinary differential equation, ODE) 的架构, 通过反对称权重矩阵, 并结合相对时间差编码对邻居消息进行时间加权求和, 保证信息在长程时空传播中不衰减.

(8) NAT[27]: 一种基于哈希映射的邻域感知方法, 通过固定大小的哈希表存储节点邻域关联信息, 并结合最短时间游走编码节点成对关系, 保证动态图中节点关联信息的高效存储和快速解码.

(9) PINT[23]: 一种基于时间随机游走的注入式架构, 通过计数多阶时间随机游走路径构建节点相对位置特征, 并累积游走信息对节点交互依赖关系进行量化表征, 从而精准捕获动态图中节点的细粒度时空动态.

4.3 超参数设置

表 2 列出了不同对比算法的核心超参数搜索范围, 其余参数均采用文献或原始代码的默认设置. 考虑到 CTAN 算法的内存需求显著高于其他算法, 为避免内存溢出, 我们将其采样邻居数量设为文献建议值 5. 同时, 为最大化 CTAN 在捕获长期时间依赖性方面的性能, 我们将其图卷积层数设置为允许的最大值 5. 虽然 DyRep、TGAT 等算法中的“采样邻居数量”与 CAWN 算法的“随机游走长度”在命名上存在差异, 但这些参数与本文采用的“交互序列长度”参数具有相同的本质含义, 即指定节点历史交互的一跳或多跳邻居序列长度.

表 2 比较方法的超参数搜索范围

超参数	范围	比较方法
采样邻居数量	[8, 16, 32, 64, 128, 256, 512, 1 024]	DyRep、TGAT、TGN、TCL、GraphMixer、NAT
随机游走长度	[8, 16, 32, 64, 128, 256]	CAWN、PINT
交互序列长度&分块数	[32&1, 64&2, 128&4, 256&8, 512&16, 1 024&32, 2 048&64, 4 096&128]	DG-BS^{3}M

4.4 评价指标和预测任务

我们采用平均精度 (average precision, AP) 和受试者工作特征曲线下面积 (area under the receiver operating characteristic curve, AUC) 作为动态链接预测的评价指标. 我们考虑两种不同类型的动态链接预测任务: (1) 直推式预测, 其目标是预测在训练过程中观察到的节点之间的未来链接; (2) 归纳式预测, 其目标是预测在训练过程中没观察到的节点之间的未来链接. 为了进行全面的评估, 我们采用随机、历史和归纳这 3 种负采样策略来产生负样本, 并与正样本一起用于评估算法的链接预测性能. 随机负采样策略是指从预测时间点上所有不存在的链接中随机选择指定数目的负样本; 历史负采样策略是指选择在训练数据中曾经出现过, 但在预测时间点上却不存在的链接作为负样本; 归纳式负采样策略则是选择在训练数据中未曾出现过, 但在测试数据中出现, 并在预测时间点上不存在的链接作为负样本. 相比随机负采样, 历史和归纳负采样策略更加困难和具有挑战性.

所有模型共训练 5 次, 均采用提前停止策略, 若连续 20 轮没有改进就提前终止, 最多训练 100 轮, 选择在验证集上达到最优性能的模型参数进行测试. 对于所有模型, 我们使用 Adam 优化器, 并将学习率和批处理大小分别设置为 0.000 1 和 200. 所有实验均在 NVIDIA GeForce RTX 4070 12 GB GPU 上完成.

4.5 实验结果分析

我们将 DG-BS^{3}M 与基线方法在直推式和归纳式两种链接预测任务上进行了比较. 表 3–表 5 分别展示了 3 种负采样策略下, 10 种方法在直推式链接预测任务中的表现对比, 评价指标为 AP. 表 6–表 8 则分别展示了在 3 种负采样策略下, 10 种方法在归纳式链接预测任务中的表现对比, 评价指标为 AUC. 表中"/"表示内存溢出, AR (average rank) 表示各个方法在不同数据集上的平均预测性能排名, 排名愈靠前, 性能表现愈佳. 最佳表现和排名通过加粗来强调, 次佳表现通过下划线来强调.

表 3 随机负采样策略下直推式链接预测 AP 性能对比

方法	AP (%)								AR
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo	UCI	Can Parl	
DyRep	94.28±0.16	98.16±0.06	79.17±1.52	71.42±1.27	76.22±4.95	88.87±0.30	67.49±0.80	64.53±0.46	9.00
TGAT	96.96±0.15	98.18±0.09	85.58±0.54	73.42±0.21	70.08±1.17	93.16±0.17	79.53±0.02	64.39±0.79	7.75
TGN	96.69±0.01	98.64±0.04	**90.73±0.15**	77.07±3.97	85.80±1.97	93.57±0.17	93.18±0.85	67.06±1.05	5.38
CAWN	**99.03±0.01**	**99.15±0.01**	86.16±0.47	91.12±0.17	91.62±0.09	84.96±0.09	**96.21±0.02**	70.08±4.60	3.13
TCL	97.73±0.02	97.89±0.01	83.00±0.27	67.27±2.16	85.70±1.04	93.13±0.16	84.06±0.65	69.09±4.35	6.88
GraphMixer	96.92±0.06	97.02±0.01	82.67±0.16	75.61±0.24	81.60±0.33	93.37±0.07	93.20±0.23	67.77±2.21	7.13
CTAN	96.61±0.79	97.21±0.84	84.71±2.85	86.44±0.80	92.52±1.20	/	76.64±4.11	67.85±2.36	6.75
NAT	98.03±0.07	99.10±0.05	85.88±0.55	88.02±1.94	90.60±0.66	88.92±3.45	93.40±0.26	79.72±1.76	4.25
PINT	98.45±0.04	99.05±0.02	87.12±0.36	89.66±1.81	92.20±0.15	94.42±0.03	95.37±0.48	68.36±1.43	3.25
DG-BS^{3}M	98.96±0.13	99.11±0.10	87.23±0.35	**93.22±0.08**	**92.53±0.10**	**94.69±0.03**	95.76±0.02	**98.97±1.07**	**1.50**

表 4 历史负采样策略下直推式链接预测 AP 性能对比

方法	AP (%)								AR
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo	UCI	Can Parl	
DyRep	79.49±0.33	79.44±0.25	79.89±0.93	74.34±0.54	67.84±6.20	93.29±0.43	55.72±0.60	59.69±2.32	8.25
TGAT	87.33±0.30	79.74±0.22	83.00±0.56	71.59±0.24	64.38±0.59	95.01±0.44	67.03±0.61	59.26±6.22	7.25
TGN	90.99±0.07	80.46±0.31	**87.16±0.21**	76.87±4.64	74.24±0.52	94.45±0.56	79.74±0.59	56.46±4.74	5.38
CAWN	84.42±0.91	80.58±0.07	80.04±2.34	82.83±0.37	78.45±0.16	85.53±0.38	77.48±0.99	73.37±3.48	5.00
TCL	86.41±4.32	81.57±0.19	77.41±0.70	59.30±2.31	74.18±0.85	94.74±0.31	69.25±0.12	67.09±8.14	6.50
GraphMixer	90.68±0.66	77.47±0.13	76.24±2.07	72.47±0.49	79.74±0.86	94.93±0.31	**83.85±0.41**	70.79±7.89	5.25
CTAN	**95.91±0.10**	**89.77±2.28**	67.73±2.08	82.29±0.94	77.24±1.53	/	76.62±0.33	66.46±3.56	5.13
NAT	68.36±4.47	80.58±0.91	85.42±3.26	78.75±1.63	72.90±1.38	91.29±4.89	75.38±1.28	77.72±1.78	6.00
PINT	91.33±0.76	83.93±1.11	84.80±1.57	**84.95±0.43**	**85.41±0.84**	96.73±0.29	81.47±0.20	63.84±5.36	**2.75**
DG-BS^{3}M	84.24±1.08	81.23±2.01	85.88± 1.61	81.96±0.19	76.12±1.69	**97.12±0.14**	82.23±1.15	**98.69±1.07**	3.38

表 5 归纳负采样策略下直推式链接预测 AP 性能对比

方法	AP (%)								AR
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo	UCI	Can Parl	
DyRep	70.69±1.72	86.29±0.37	64.16±0.27	64.41±2.70	68.57±1.86	93.28±0.48	56.02±2.16	55.59±1.05	8.63
TGAT	86.77±0.12	87.80±0.20	76.76±0.14	71.13±0.17	64.66±0.11	94.84±0.44	67.87±0.23	56.03±4.77	5.75
TGN	87.33±0.01	87.61±0.45	78.29±1.06	65.95±5.98	71.26±2.89	95.13±0.56	70.40±1.67	53.90±4.51	5.38
CAWN	85.33±1.29	91.15±0.03	**81.36±1.54**	69.96±0.40	75.75±0.30	88.32±0.27	70.92±1.99	75.75±0.30	3.88
TCL	73.58±10.61	77.97±0.09	75.59±0.63	58.21±0.89	72.80±0.37	94.90±0.36	69.10±0.06	66.16±6.93	6.88
GraphMixer	87.88±0.17	84.58±0.06	73.38±1.25	68.12±0.33	76.08±0.91	94.72±0.33	**79.18±0.25**	64.63±5.61	4.88
CTAN	**94.15±0.08**	90.99±2.19	64.93±3.31	**80.06±0.85**	72.02±2.64	/	66.25±0.51	69.45±2.56	5.25
NAT	53.85±6.25	81.71±1.03	77.28±1.94	64.02±1.48	70.34±1.73	92.12±4.53	57.55±0.81	73.08±1.27	7.50
PINT	82.39±2.70	87.59±0.59	75.70±1.43	70.79±0.47	74.69±1.37	97.15±0.25	73.61±0.23	59.15±6.20	4.88
DG-BS^{3}M	84.33±2.75	**91.38±0.21**	81.17±1.82	71.87±1.04	**76.14±4.58**	**97.54±0.07**	73.62±1.13	**97.51±0.97**	**2.00**

表 6 随机负采样策略下归纳式链接预测 AUC 性能对比

方法	AUC (%)								AR
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo	UCI	Can Parl	
DyRep	90.84±0.47	95.94±0.08	82.97±0.42	82.24±1.51	70.54±5.88	91.18±0.30	61.26±1.40	48.95±2.71	8.25
TGAT	95.97±0.22	96.65±0.19	86.57±0.28	76.99±0.29	63.49±2.59	93.41±0.17	77.42±0.07	58.67±1.19	6.38
TGN	95.77±0.03	97.41±0.13	**91.37±0.92**	82.61±3.15	78.39±1.31	93.43±0.17	88.31±1.28	53.42±0.71	5.38
CAWN	**98.55±0.01**	98.59±0.01	85.48±0.12	92.13±0.33	88.00±0.05	84.73±0.09	**92.59±0.01**	56.16±1.80	3.50
TCL	96.96±0.02	96.03±0.04	81.86±0.18	70.84±0.85	81.78±0.93	93.71±0.16	79.76±0.22	57.47±0.29	6.00
GraphMixer	96.03±0.02	94.52±0.01	81.27±0.37	80.37±0.18	76.00±0.41	94.09±0.07	89.60±0.22	53.67±3.92	6.38
CTAN	92.59±0.70	82.35±4.03	66.38±1.59	61.49±2.78	75.23±2.24	/	48.58±6.02	55.27±2.45	9.13
NAT	95.82±0.18	98.00±0.04	84.72±1.31	91.60±1.31	87.95±0.58	88.15±6.36	83.78±0.37	62.70±2.91	5.13
PINT	97.76±0.05	98.38±0.07	90.27±0.96	92.15±0.68	87.97±0.61	94.78±0.37	92.46±0.17	49.64±0.69	3.38
DG-BS^{3}M	98.50±0.08	**98.66±0.02**	87.07±0.54	**94.29±0.04**	**90.64±0.42**	**95.32±0.03**	92.55±0.11	**92.62±2.70**	**1.50**

表 7 历史负采样策略下归纳式链接预测 AUC 性能对比

方法	AUC (%)								AR
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo	UCI	Can Parl	
DyRep	56.39±1.35	61.06±0.76	63.90±0.59	68.79±1.08	57.54±2.73	87.93±0.05	54.31±2.02	47.93±3.75	8.25
TGAT	78.33±0.37	64.48±0.46	74.97±0.32	69.89±0.28	56.84±1.60	91.87±0.72	61.37±0.12	59.45±0.83	5.25
TGN	82.79±0.21	64.97±0.67	77.17±1.42	66.99±5.62	62.15±1.58	92.10±1.22	64.89±0.72	50.86±1.52	4.63
CAWN	72.36±0.77	63.73±0.17	73.99±3.34	68.24±0.18	69.85±0.10	83.54±0.24	62.52±1.29	59.19±0.45	5.63
TCL	71.67±5.40	62.41±0.04	70.05±0.72	55.88±1.85	68.97±0.46	93.28±0.60	61.49±0.17	58.06±2.39	6.50
GraphMixer	82.70±0.36	64.17±0.16	71.66±1.24	70.07±0.20	**70.31±0.78**	93.62±0.35	**75.69±0.04**	53.39±4.54	3.50
CTAN	**91.12±0.13**	**81.70±4.71**	58.06±0.89	57.85±3.66	68.70±1.83	/	51.68±2.61	60.36±3.57	6.00
NAT	40.95±4.99	58.32±0.63	67.99±1.68	64.18±0.65	61.69±0.68	84.89±4.72	42.59±0.96	61.72±2.76	8.00
PINT	73.29±2.33	64.02±0.46	75.92±2.48	**75.02±0.66**	68.90±1.29	96.29±0.56	65.34±0.29	48.93±2.35	4.13
DG-BS^{3}M	71.97±1.24	64.52±1.06	**79.67±1.59**	69.58±0.56	64.77±1.97	**96.89±0.09**	66.42±1.08	**92.02±1.85**	**3.13**

表 8 归纳负采样策略下归纳式链接预测 AUC 性能对比

方法	AUC (%)								AR
	Wikipedia	Reddit	MOOC	LastFM	Enron	Social Evo	UCI	Can Parl	
DyRep	56.40±1.34	61.07±0.77	63.91±0.59	68.78±1.08	61.54±0.72	87.94±0.05	54.32±2.03	45.46±4.13	8.25
TGAT	78.29±0.20	64.74±0.19	75.08±0.41	69.90±0.28	58.97±0.47	91.88±0.72	61.35±0.09	56.72±1.46	5.63
TGN	82.80±0.22	64.96±0.67	77.16±1.43	67.00±5.61	62.16±1.58	92.11±1.22	64.93±0.73	50.54±1.57	4.88
CAWN	72.37±0.78	63.75±0.15	77.28±1.31	68.25±0.18	70.17±0.20	83.55±0.23	62.56±1.29	70.17±0.20	4.88
TCL	71.68±5.39	62.42±0.04	70.06±0.72	55.89±1.83	68.98±0.45	93.29±0.60	61.53±0.18	58.17±2.49	6.38
GraphMixer	82.71±0.36	64.19±0.16	72.78±1.37	70.06±0.20	**70.32±0.78**	93.63±0.35	**75.73±0.04**	53.32±4.72	3.50
CTAN	**91.13±0.11**	**81.71±4.73**	58.07±0.89	57.86±3.67	68.71±1.83	/	51.67±2.61	60.35±3.55	6.13
NAT	40.96±4.96	58.31±0.61	68.00±1.69	64.19±0.66	61.70±0.68	84.88±4.72	42.63±0.97	61.83±2.92	8.13
PINT	73.30±2.34	64.04±0.46	75.93±2.48	**75.01±0.66**	68.91±1.28	96.29±0.56	65.35±0.29	48.94±2.34	4.25
DG-BS^{3}M	71.98±1.24	65.04±0.88	**79.80±1.59**	69.59±0.56	64.45±3.39	**96.90±0.11**	66.45±1.08	**92.04±1.77**	**3.00**

从表 3–表 8 中可以看到, 无论是直推式还是归纳式链接预测任务, DG-BS^{3}M 都比其他基线方法表现出更优秀的性能. 尤其在历史负采样和归纳负采样策略下, DG-BS^{3}M 在 Can Parl 等数据集上表现出较大的优势, 即比次优预测结果分别高出 32% 和 22%. 这主要是因为以下两个原因: 第一, 双向交互序列表达更新模块通过选择性状态空间模型对节点交互序列进行双向编码, 除了能够充分捕捉节点之间的长期时间依赖关系, 还能够挖掘出序列背后隐藏的非因果关系模式, 这对于全面理解节点之间的复杂交互模式具有重要意义. 第二, 多种不同类型的特征提取和融合能够有效捕获节点之间的相关性, 为预测节点间的未来链接也提供了有效信息. 同时, 我们发现 DG-BS^{3}M 对不同时间尺度的动态图也具有良好的适应性. 无论是细粒度的秒级交互网络 (如 MOOC), 还是粗粒度的

年度交互网络 (如 Can Parl), DG-BS^{3}M 均表现出优越的性能. 这是因为其双向编码机制通过前向建模 (短期关联) 与反向建模 (长期关联) 的互补, 既能捕捉细粒度短期依赖, 又能建模粗粒度长期稳定关系, 从而实现从微观到宏观时间尺度的灵活适配. 此外, 我们也观察到 DG-BS^{3}M 在个别数据集上表现并不突出, 这可能是因为它采用了简单相加的方式融合正反向嵌入表示. 这种方式默认双向信息权重均等, 无法根据数据集的特性自适应分配权重, 使得模型在需要精细区分双向信息价值的场景中, 难以充分挖掘出节点间的有效依赖信息.

4.6 长序列建模分析

为了验证 DG-BS^{3}M 在捕获序列长期依赖方面的优越性, 我们测试了不同方法随着交互序列长度的增加, 在多个不同规模的数据集上产生的动态链接预测性能. CTAN 因为内存要求远超其他算法, 无法处理长交互序列 ($L \geqslant 8$), 因此未纳入比较. 图 5 和图 6 分别展示了随机采样策略下各个方法在相同内存下能够捕获最长序列时间依赖以及它们在两种链接预测任务中的性能表现.

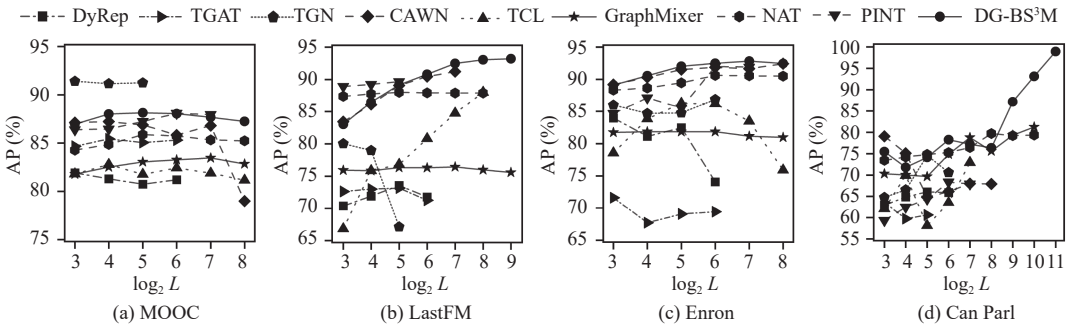

图 5 不同序列长度下直推式链接预测任务的表现对比

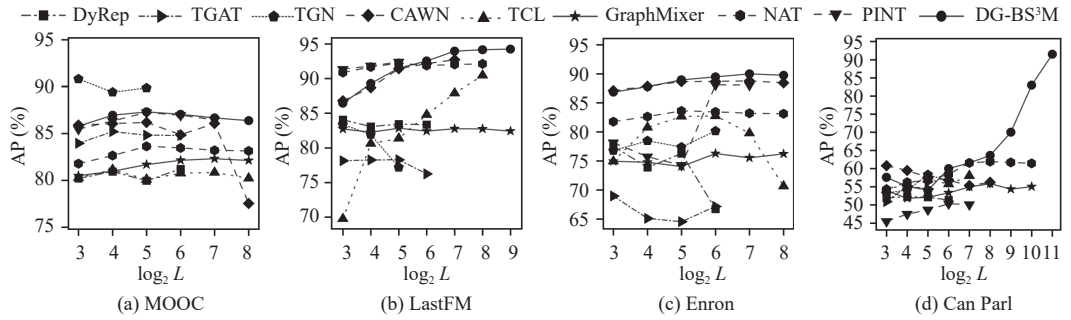

图 6 不同序列长度下归纳式链接预测任务的表现对比

从图 5 和图 6 中可以看出, 随着序列长度的增加, DG-BS^{3}M 的链接预测性能在不同数据集上逐步提升, 然后趋于平稳. 在相同的内存条件下, 有的方法 (如 CAWN、GraphMixer、NAT 和 DG-BS^{3}M) 能够捕获较长的序列时间依赖关系, 而有的一些方法 (如 DyRep、TGAT、TGN 和 PINT) 则只能捕获一些较短的序列时间依赖关系, 因为在处理长序列时会导致内存溢出. 尽管 CAWN、GraphMixer 和 NAT 方法跟我们提出的 DG-BS^{3}M 一样能在有限的内存中处理节点长序列历史交互信息, 但由于其潜在的梯度爆炸或消失问题, 它们的性能没有从节点更长历史交互序列中受益. 例如, 在 Can Parl 数据集上, CAWN、GraphMixer 和 NAT 等方法的预测性能随着序列长度的增加逐渐趋于平缓, 甚至出现下降趋势. 相比之下, DG-BS^{3}M 的预测性能稳步提升, 始终保持领先优势. 这充分验证了我们的方法在捕捉序列长期时间依赖关系方面的优势, 在较低的时间和空间复杂度下表现出更优越的预测性能.

表 9 进一步对比了不同方法随序列长度增加时单轮的训练时间. 实验结果表明, 随着序列长度的增加, 大部分基线方法的训练时间呈显著上升趋势, 这反映出它们较高的计算时间复杂度. 有的方法 (如 GraphMixer、NAT) 虽

然在处理长序列时耗时较少, 但其性能却表现欠佳 (图 5 和图 6). 相比之下, DG-BS^{3}M 在保持优异性能的同时, 仍能维持相对较低的计算时间.

表 9 不同序列长度下不同方法的训练时间

数据集	序列长度	DyRep	TGAT	TGN	CAWN	TCL	GraphMixer	NAT	PINT	DG-BS^{3}M
MOOC	32	4 min	6 min	14 min	4 min	2 min	2 min	3 min	8 min	11 min
	64	15 min	9 min	—	14 min	2 min	2 min	5 min	14 min	12 min
	128	—	—	—	68 min	5 min	3 min	8 min	17 min	14 min
	256	—	—	—	95 min	10 min	5 min	12 min	—	16 min
LastFM	32	14 min	21 min	48 min	15 min	4 min	3 min	8 min	26 min	13 min
	64	18 min	28 min	—	65 min	7 min	5 min	12 min	—	15 min
	128	—	—	—	92 min	73 min	7 min	14 min	—	16 min
	256	—	—	—	—	80 min	14 min	17 min	—	18 min
Enron	32	14 min	22 min	48 min	15 min	4 min	4 min	3 min	3 min	10 min
	64	18 min	28 min	—	64 min	7 min	4 min	4 min	6 min	11 min
	128	—	—	—	78 min	64 min	7 min	7 min	10 min	13 min
	256	—	—	—	82 min	78 min	14 min	10 min	—	15 min
Can Parl	32	4 min	3 min	4 min	1 min	16 s	13 s	8 s	3 min	34 s
	64	7 min	—	7 min	2 min	27 s	18 s	10 s	6 min	43 s
	128	—	—	—	4 min	1 min	27 s	15 s	8 min	1 min
	256	—	—	—	7 min	—	1 min	20 s	—	1 min

为更直观地呈现 DG-BS^{3}M 的时间与空间复杂度特性, 图 7 展示了其在不同数据集上随序列长度变化的单轮训练耗时 (min) 及最大内存消耗 (GB) 的效率曲线. 从图 7 中可见, 随着序列长度的增加, 模型的时空消耗呈线性增长趋势. 这一特性表明 DG-BS^{3}M 具有较低的时空复杂度特征, 使其能够在有限计算资源下高效建模序列的长期时间依赖关系.

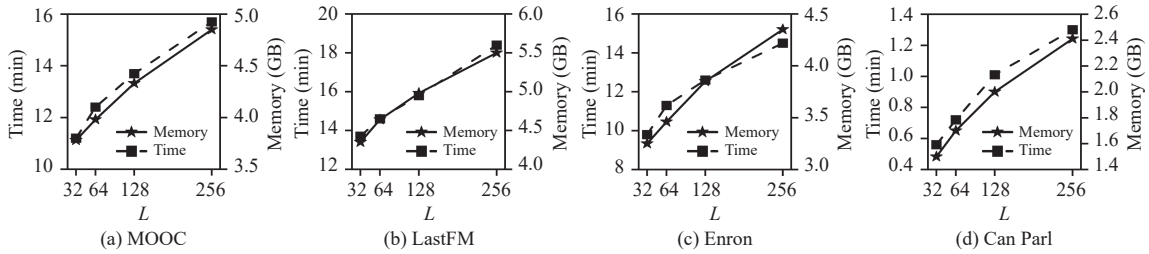

图 7 不同序列长度下 DG-BS^{3}M 的效率曲线

4.7 消融性分析

为了验证 DG-BS^{3}M 中核心模块的有效性, 我们进行如下消融性实验: (1) 消除双向交互序列表达更新模块, 记为 w/o BISRU; (2) 消除邻居共现频率嵌入表示模块, 记为 w/o CNE; (3) 消除时间嵌入表示模块, 记为 w/o TE; (4) 消除 BISRU 模块中的选择性机制, 使用传统的结构化状态空间模型 S4[29], 记为 DG-S4. 我们将以上 4 种算法变种与完整的 DG-BS^{3}M 进行动态链接预测性能的比较. 图 8 和图 9 分别演示了随机采样策略下 5 种方法在直推式和归纳式链接预测任务中的表现对比. 从图 8 和图 9 中我们可以看到, BISRU 模块对 DG-BS^{3}M 的链接预测性能具有最显著的提升作用, 邻居共现频率嵌入表示和时间嵌入表示的提取也在一定程度上提升了模型的预测能力. 相比之下, DG-S4 的预测效果出现较明显的下降, 这是因为结构化状态空间模型具有固定的状态转移参数, 无法根据不同时刻的输入上下文动态调整参数权重, 从而导致冗余信息干扰模型进行特征编码. 这也进一步印证了 DG-BS^{3}M 的选择性机制能够自适应筛选出节点历史交互中有效信息的重要性.

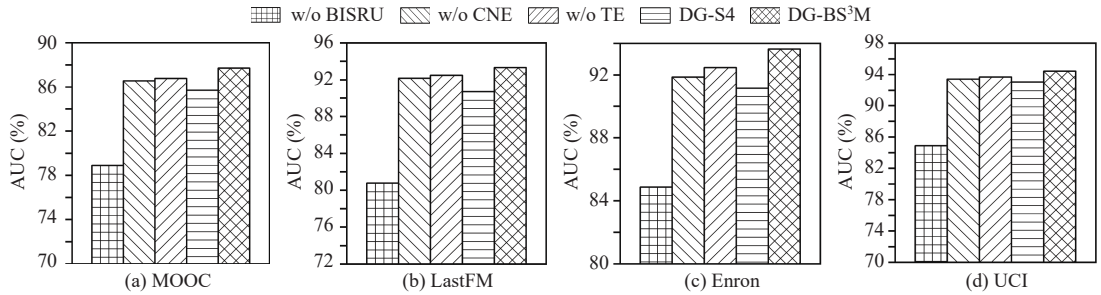

图 8　直推式链接预测任务中的消融性分析

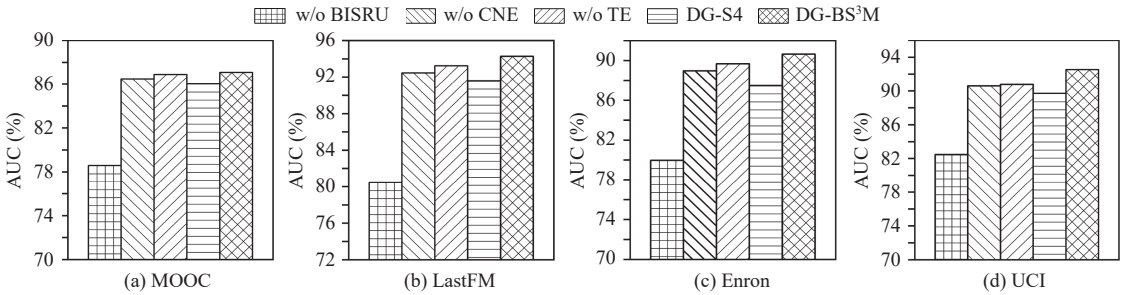

图 9　归纳式链接预测任务中的消融性分析

为了进一步研究 BISRU 模块中双向编码的重要性, 我们实现了仅使用正向和逆向选择性状态空间编码的动态图表示学习方法, 分别记为 DG-US^{3}M-F (dynamic graph representation learning based on unidirectional selective state space model-forward) 和 DG-US^{3}M-R (dynamic graph representation learning based on unidirectional selective state space model-reverse), 并将它们与采用双向序列编码的 DG-BS^{3}M 进行比较. 图 10 和图 11 分别展示了 3 种负采样策略这 3 种算法在直推式和归纳式链接预测任务中的性能对比.

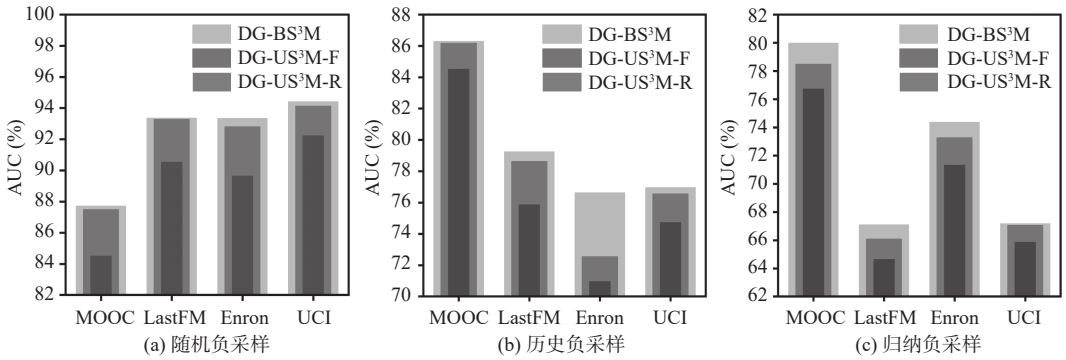

图 10　3 种负采样策略下单双向序列编码的直推式链接预测性能对比

从图 10 和图 11 中可以看出, 仅使用正向或反向交互序列编码使得模型预测性能在各个数据集上都出现不同程度的下降, 其中仅使用反向交互序列编码导致的性能下降幅度更大. 这是因为节点交互通常遵循因果逻辑, 而反向交互序列编码倒置时序关系, 仅凭借捕获的非因果依赖关系难以反映节点间真实的关联度. 相较之下, 使用双向交互序列编码则在各个数据集上均展现出更优的预测性能, 因为它能够更全面地融合动态图中事件间的因果链与非因果性交互模式, 从而提升预测的鲁棒性与泛化能力.

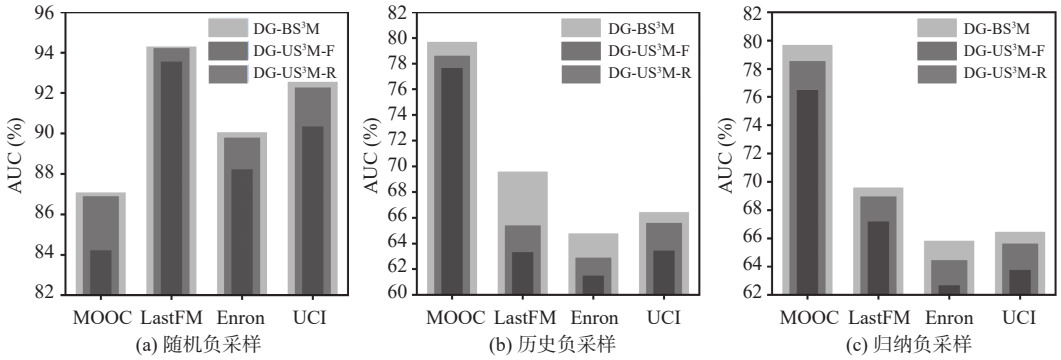

图 11　3 种负采样策略下单双向序列编码的归纳式链接预测性能对比

5 总　结

在本文中, 我们提出了一种基于双向选择性状态空间模型的连续时间动态图表示学习方法 DG-BS^{3}M. 该方法通过双向选择性状态空间模型对节点交互序列同时进行正向和反向联合编码, 不仅在计算资源有限的情况下能够高效捕捉交互序列的长期依赖关系, 而且还能深度挖掘出序列中隐藏的非因果关系模式. 这种双重编码机制显著增强了连续时间动态图中节点的嵌入表示学习能力, 使模型在动态链接预测等任务中展现出更优的性能. 在未来的工作中, 我们将一方面通过增强对双向信息价值差异的适配能力, 进一步提升模型的动态图表示学习能力; 另一方面通过对不同类型的点和边进行编码, 将模型拓展至异构图、带属性边、多关系图等更复杂的动态图上.

References

[1] Kazemi SM, Goel R, Jain K, Kobyzev I, Sethi A, Forsyth P, Poupart P. Representation learning for dynamic graphs: A survey. The Journal of Machine Learning Research, 2020, 21(1): 70.

[2] Wen ZH, Fang Y. TREND: TempoRal event and node dynamics for graph representation learning. In: Proc. of the 2022 ACM Web Conf. Lyon: ACM, 2022. 1159–1169. [doi: 10.1145/3485447.3512164]

[3] Zhang SZ, Chen LY, Wang C, Li SL, Xiong H. Temporal graph contrastive learning for sequential recommendation. In: Proc. of the 38th AAAI Conf. on Artificial Intelligence. Vancouver: AAAI, 2024. 9359–9367. [doi: 10.1609/aaai.v38i8.28789]

[4] Zhao WZ, Yuan G, Zhang YM, Qiao SJ, Wang SZ, Zhang L. Multi-view fused spatial-temporal dynamic GCN for urban traffic flow prediction. Ruan Jian Xue Bao/Journal of Software, 2024, 35(4): 1751–1773. http://www.jos.org.cn/1000-9825/7018.htm [doi: 10.13328/j.cnki.jos.007018]

[5] Besta M, Catarino AC, Gianinazzi L, Blach N, Nyczyk P, Niewiadomski H, Hoefler T. HOT: Higher-order dynamic graph representation learning with efficient Transformers. In: Proc. of the 2nd Learning on Graphs Conf. PMLR, 2024. 231.

[6] Wang YB, Chang YY, Liu YY, Leskovec J, Li P. Inductive representation learning in temporal networks via causal anonymous walks. In: Proc. of the 9th Int'l Conf. on Learning Representations. OpenReview.net, 2021.

[7] Jin M, Li YF, Pan SR. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 1445.

[8] Xia WW, Li YC, Li SH. Graph neural point process for temporal interaction prediction. IEEE Trans. on Knowledge and Data Engineering, 2023, 35(5): 4867–4879. [doi: 10.1109/TKDE.2022.3149927]

[9] Trivedi RS, Farajtabar M, Biswal P, Zha HY. DyRep: Learning representations over dynamic graphs. In: Proc. of the 7th Int'l Conf. on Learning Representations. New Orleans: OpenReview.net, 2019.

[10] Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, Bronstein M. Temporal graph networks for deep learning on dynamic graphs. arXiv:2006.10637, 2020.

[11] Kumar S, Zhang XK, Leskovec J. Predicting dynamic embedding trajectory in temporal interaction networks. In: Proc. of the 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining. Anchorage: ACM, 2019. 1269–1278. [doi: 10.1145/3292500.3330895]

[12] Xu D, Ruan CW, Körpeoglu E, Kumar S, Achan K. Inductive representation learning on temporal graphs. In: Proc. of the 8th Int'l Conf. on Learning Representations. Addis Ababa: OpenReview.net, 2020.

[13] Han LZ, Ma XJ, Sun LL, Du BW, Fu YJ, Lv WF, Xiong H. Continuous-time and multi-level graph representation learning for origin-destination demand prediction. In: Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Washington:

ACM, 2022. 516–524. [doi: 10.1145/3534678.3539273]
[14] Wu YX, Fang Y, Liao LZ. On the feasibility of simple Transformer for dynamic graph modeling. In: Proc. of the 2024 ACM Web Conf. Singapore: ACM, 2024. 870–880. [doi: 10.1145/3589334.3645622]
[15] Yu L, Sun LL, Du BW, Lv WF. Towards better dynamic graph learning: New architecture and unified library. In: Proc. of the 37th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2023. 2960.
[16] Wang L, Chang XF, Li S, Chu YF, Li H, Zhang W, He XF, Song L, Zhou JR, Yang HX. TCL: Transformer-based dynamic graph modelling via contrastive learning. arXiv:2105.07944, 2021.
[17] Li D, Huang T, Hong J, Hong YL, Wang JQ, Wang Z, Zhang X. Event sparse net: Sparse dynamic graph multi-representation learning with temporal attention for event-based data. In: Proc. of the 6th Chinese Conf. on Pattern Recognition and Computer Vision. Xiamen: Springer, 2024. 208–219. [doi: 10.1007/978-981-99-8546-3_17]
[18] Gu A, Dao T. Linear-time sequence modeling with selective state spaces. In: Proc. of the 1st Conf. on Language Modeling (COLM), 2024. 752–765. [doi: 10.48550/arXiv.2312.00752]
[19] Liu J, Shang XQ, Han XL, Zhang WT, Yin HZ. Spatial-temporal memories enhanced graph autoencoder for anomaly detection in dynamic graphs. arXiv:2403.09039, 2024.
[20] Vignac C, Krawczuk I, Siraudin A, Wang BH, Cevher V, Frossard P. DiGress: Discrete denoising diffusion for graph generation. In: Proc. of the 11th Int'l Conf. on Learning Representations. Kigali: OpenReview.net, 2023.
[21] Pareja A, Domeniconi G, Chen J, Ma TF, Suzumura T, Kanezashi H, Kaler T, Schardl T, Leiserson C. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In: Proc. of the 34th AAAI Conf. on Artificial Intelligence. New York: AAAI, 2020. 5363–5370. [doi: 10.1609/aaai.v34i04.5984]
[22] Xu Y, Han LZ, Sun LL, Du BW, Liu CR, Xiong H. Bootstrapping on continuous-time dynamic graphs for crowd flow modeling. IEEE Trans. on Knowledge and Data Engineering, 2024, 36(10): 5039–5052. [doi: 10.1109/TKDE.2024.3383663]
[23] Souza AH, Mesquita D, Kaski S, Garg V. Provably expressive temporal graph networks. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 2337.
[24] Tian S, Xiong T, Shi LL. Streaming dynamic graph neural networks for continuous-time temporal graph modeling. In: Proc. of the 2021 IEEE Int'l Conf. on Data Mining (ICDM). Auckland: IEEE, 2021. 1361–1366. [doi: 10.1109/ICDM51629.2021.00171]
[25] Cong WL, Zhang S, Kang J, Yuan BC, Wu H, Zhou X, Tong HH, Mahdavi M. Do we really need complicated model architectures for temporal networks? In: Proc. of the 11th Int'l Conf. on Learning Representations. Kigali: OpenReview.net, 2023.
[26] Gravina A, Lovisotto G, Gallicchio C. Long range propagation on continuous-time dynamic graphs. In: Proc. of the 41st Int'l Conf. on Machine Learning. Vienna: PMLR, 2024. 16206–16225.
[27] Luo YH, Li P. Neighborhood-aware scalable temporal network representation learning. In: Proc. of the 1st Learning on Graphs Conf. PMLR, 2022. 1.
[28] Gupta A, Gu A, Berant J. Diagonal state spaces are as effective as structured state spaces. In: Proc. of the 36th Int'l Conf. on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 1670.
[29] Gu A, Goel K, Ré C. Efficiently modeling long sequences with structured state spaces. In: Proc. of the 10th Int'l Conf. on Learning Representations. OpenReview.net, 2022.
[30] Smith J T H, Warrington A, Linderman S W. Simplified state space layers for sequence modeling. In: Proc. of the 11th Int'l Conf. on Learning Representations. OpenReview.net, 2023.
[31] Gu A, Dao T, Ermon S, Rudra A, Ré C. HiPPO: Recurrent memory with optimal polynomial projections. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 125.
[32] Kumar S, Zhang X, Leskovec J. Predicting dynamic embedding trajectory in temporal interaction networks. 2019. https://zenodo.org/records/7213796

附中文参考文献

[4] 赵文竹, 袁冠, 张艳梅, 乔少杰, 王森章, 张雷. 多视角融合的时空动态 GCN 城市交通流量预测. 软件学报, 2024, 35(4): 1751–1773. http://www.jos.org.cn/1000-9825/7018.htm [doi: 10.13328/j.cnki.jos.007018]

作者简介

何萍, 博士, 副教授, 主要研究领域为机器学习, 数据挖掘.
汪雷达, 硕士生, 主要研究领域为机器学习, 数据挖掘.
徐晓华, 博士, 副教授, 主要研究领域为机器学习, 数据挖掘, 并行计算.
严勇林, 硕士生, 主要研究领域为机器学习, 数据挖掘.