四川大學

本科生毕业设计（学术论文）

题　　目　基于图卷积循环动态图嵌入的异常节点检测

学　　院　计算机学院

专　　业　计算机科学与技术

学生姓名　吴若婷

学　　号　201714150211　年级　2017

指导教师　段磊、陈亮（中山大学）

教务处制表

二〇二一年五月二十八日

基于图卷积循环动态图嵌入的异常节点检测

专业：计算机科学与技术

学生：吴若婷 指导老师：段磊 陈亮

[摘要] 数字加密货币在区块链、智能合约等技术的基础上诞生和发展，其交易平台也产生了诸多安全问题。对数字加密货币交易平台威胁最大的问题是网络钓鱼诈骗：网络钓鱼诈骗者构建虚拟平台，使用邮件等私人通讯诱导用户，最终暴露用户个人信息、窃取用户财产。为了给区块链生态系统创造良好的投资环境，需要对实施网络钓鱼的异常账户进行检测。

在数字加密货币的交易网络中，每笔交易都包括交易双方、金额和时间信息，账户和交易都随时间动态增加，因此交易网络可以被看作一个动态图结构。为了实现对钓鱼账户的检测，需要将所有的账户分为异常的钓鱼账户和正常账户。对图中的节点进行表示学习是图节点分类前的关键。当前针对节点表示学习的研究大多基于静态图，缺乏对图时序信息的利用。如何结合交易图中的空间拓扑信息和时序信息更好地表征节点，是目前面临的一大挑战。

为了解决动态图嵌入模型难以捕获节点拓扑信息的问题，同时保证模型学习到动态图的时序信息，结合图卷积神经网络和循环神经网络，论文提出了一种无监督学习的动态图节点嵌入的方法DynGCREM，能够同时捕捉图的结构信息和时序信息。首先图卷积网络层提取每个动态图快照的结构信息，再利用循环神经网络层中的长短期记忆自编码网络学习动态图的时序信息。该模型在以太坊的交易网络数据集进行了广泛的实验，实验验证了模型的有效性，证明了图卷积网络和循环神经网络相结合的确可以提升动态图表示学习的效果。

[关键词] 图卷积网络；动态图嵌入；异常检测；节点分类

Abnormal Node Detection based on Dynamic Graph Convolutional Recurrent Embedding Method

Computer Science and Technology

Student: WU Ruoting Adviser: DUAN Lei CHEN Liang

[Abstract] Cryptocurrency is created and developed based on blockchain, smart contracts, and other technologies, and its trading platforms also arise many security problems. The biggest threat to the cryptocurrency trading platform is phishing scams: Phishing fraudsters build a virtual platform, use e-mail and other private communications to induce users, and finally expose users' personal information and stealing users' property. To create a good investment environment for the blockchain ecosystem, it is necessary to detect abnormal phishing accounts.

In the cryptocurrency transaction network, each transaction includes the information of accounts on both sides, transaction amount, and transaction time. Accounts and transactions increase dynamically, so the transaction network can be regarded as a dynamic graph structure. All accounts need to be classified into 2 categories, abnormal accounts and normal accounts to detect phishing scams. Node representation learning is vital before node classification. Most of the current researches are node representation learning on static graphs, which lacks the use of graph temporal information. How to combine the spatial topology information and temporal information in a transaction network to better represent nodes is a major challenge.

To solve the problem that it is difficult for dynamic graph embedding model to capture node topology information while ensuring that the model can learn the temporal information of dynamic graph, this paper proposes an unsupervised learning method for dynamic graph node embedding based on Graph Convolutional Network and Recurrent Neural Network, which can capture the structural and temporal information of the graph at the same time. First, the graph convolution network layer extracts the structural information of each dynamic graph snapshot, and then the Long Short Term Memory autoencoder network in the recurrent neural network layer learns the temporal information of the dynamic graph. Experiments of the model are conducted in the Ethereum transaction network dataset. The results imply the effectiveness of the model, prove that the combination of Graph Convolution Network and Recurrent Neural Network can indeed improve the effect of dynamic graph representation learning.

[Key Words] Graph convolutional network; Dynamic graph embedding; Anomaly detection; Node classification

目 录

1 绪论

1.1 选题背景

区块链是一种去中心化共享账本，它以密码学的方式保证交易不被篡改和伪造。通过智能合约，区块链中的匿名用户能够在无中心授权的情况下进行交易。交易将按照时间顺序被高效、永久和可验证地记录在数据区块中，并连接形成链式数据结构。智能合约是一套公开透明的、不可更改的以数字形式定义的承诺，包括合约参与方执行的承诺[1]。智能合约在区块链交易中会自动执行，保证区块链中的数字货币交易过程被严格执行，并不可操纵。

基于区块链和智能合约技术，以太坊（Ethereum）于 2013 年被提出，它提供去中心化的虚拟机来处理智能合约。以太币（Ether，简称 ETH）是以太坊平台唯一允许交易的数字加密货币。以太币是继比特币的第二大数字加密货币，截至 2021 年 4 月，以太币的总市值约为3154亿美元[2]。但随着以太坊的发展，各种网络犯罪层出不穷。ChainAnalysis 的报告指出，在 2017 年上半年，共有 30287 名受害者遭受数字加密货币网络诈骗，损失金额达 2.25 亿美元。在所有的网络诈骗中，最为猖獗的是钓鱼诈骗，报告称 2017 年以来，所有网络犯罪收入的 50%以上来源于以太坊的钓鱼诈骗[3]，《加密犯罪报告》[4]也指出，在 2017 和 2018 年中，网络钓鱼诈骗行为占所有诈骗行为中的 63.7%。

在以太币的交易网络中，钓鱼诈骗者会构建出虚拟的平台、软件或网站，这些平台可能具有相关的恶意代码，用户在访问这些平台时，他们的敏感信息会被收集；同时钓鱼诈骗者会使用电子邮件等私人通讯给用户发送信息、诱导用户，最终暴露用户个人信息、窃取用户财产。一个较为出名的例子是 Bee Token，网络钓鱼诈骗者在 25 小时内从投资者中骗取了约 100 万美元[5]。

由此可见，区块链生态系统中的金融问题亟待解决。为了避免攻击者长期获得不法利益，避免诈骗行为对区块链生态系统交易安全的威胁，创造良好和安全的投资环境，需要运用有效的方法来检测以太坊中的网络钓鱼诈骗账户。

区块链中心化和去信任的特点使得它能够记录数字加密货币的所有交易信息，包括钓鱼诈骗者窃取受害者财产的交易。每笔交易都包括交易双方、金额和时间信息，账户和交易都随时间动态增加，因此交易网络可以被看作一个动态图结构。本论文利用深度学习技术，在图节点缺乏自身特征的情况下，结合动态图的空间拓扑信息和时序信息，对动态图的节点进行嵌入表示学习，从而提升动态图中节点分类的性能，更有效地检测出以太坊数字加密货币交易网络中的异常的钓鱼账户，及时排查异常账户，减少受害者的财产损失，保障以太坊等区块链数字货币交易平台的安全，有较强的实际意义。

1.2 国内外研究现状

1.2.1 钓鱼诈骗检测

针对钓鱼诈骗检测的相关方案主要可以分为三类：对钓鱼网站的分析、对钓鱼邮件内容的分析和对交易往来数据的检测。

对钓鱼网站的分析包括对网站本身的信息，如源代码[6]、网站 URL[7]和页面布局[8]等进行检测。钓鱼网站为了掩盖诈骗的目的，会尽可能地伪装成官方网站，降低用户的警惕性。因此可以通过分析网站检测出潜在的钓鱼网站。但近年来，诈骗者较少地使用钓鱼网站作为诈骗媒介，而是采用私人通信作为手段，而且被检测出的网站只能通过进一步的排查才能得知其是否实施了钓鱼诈骗行为，因此这种方法只能作为参考。

对钓鱼邮件内容的分析，最早用于欺诈和垃圾邮件检测的技术，如频繁模式挖掘[9]，行为监控[10]，监督学习[11]，较多地被运用在检测中。但这些技术主要依赖于预先给出的模式，缺乏灵活性的结构特征，而且没有考虑诈骗者与受害者之间的交易往来，无法从交易数据中提取交易特征。

对交易往来数据的检测，主要是利用交易网络中的交易特征进行检测，如 Trans2vec[12]。这一方案利用基于随机游走的方式构建节点表示向量，再使用支持向量机对节点进行分类，从而实现钓鱼节点检测。但这一方法仅能处理静态图数据，对动态的交易网络中的时序信息利用不足。

1.2.2 图异常节点检测

传统的图异常节点检测可以分为静态图的异常节点检测和动态图的异常节点检测。

在静态图中，单个节点的异常检测有多种解决方法。基于图表示学习的方法将图异常检测问题转化成目前较为普遍的异常点检测问题。基于图结构的方法通过计算图中节点的相近度来捕捉对象之间的简单自相关性，邻近的对象会被认为属于同一类。一种典型的方法是提出的一种基于特征的异常检测技术，称为 ODDBALL[13]。该技术提取基于 Egonet 的特征，并找出图中大多数 Egonet 对这些特征遵循的模式，可以发现异常的不遵循观察到的模式的 Egonet，从而检测出异常的节点。基于深度学习的方法多使用有监督深度学习框架，如 Chouiekh[14]等。静态图节点的异常检测对图的时序信息利用不足，难以迁移到不断变化的动态图结构中。

在动态图中，一些工作[15]使用 ARMA 模型对时间序列进行建模，但该模型无法考虑一段时间内的网络特征变化特点，因此长短期记忆网络[16]被引入到动态图异常节点检测的算法模型中。另外一些方法通过确定一个节点在整个时间内在社区中的移动水平，在社区结构网络中检测异常结点。DBMM[17]是一种动态行为混合模型，该模型利用特征基元的概念，将学习到的角色推广到未观察到的顶点。DBMM 关注的是时间异常，在每个

时间步中，都会发生一个确定节点角色的特征提取过程。另一个例子是 ECOutlier[18]，它在连续的时间戳中检测出能够反映异常社区成员进化的节点。根据在数据中发现的潜在社区，作者提出了一种将社区匹配和异常检测相结合的方法，问题被转化成通过对异常节点分配较低的权值忽略它们从而最小化社区匹配错误。对于异常节点的检测，以上的方法更多考虑时序信息，较少结合图的拓扑结构信息。

1.2.3 图表示学习

图是一种非欧几里得数据，它的空间结构不具备规则，对于图数据，除了节点和链接的特征信息，图还具有结构信息。在图机器学习任务中，图表示学习（Graph Representation Learning），又称图嵌入（Graph Embedding），是利用图的结构信息对图的节点进行特征工程。

图节点的嵌入可以被归纳成编码器-解码器（encoder-decoder）的框架：encoder 输入整个图，输出图上的每个节点映射到低维的向量空间的嵌入表示，decoder 输入低维的节点嵌入表示，然后利用它们重构原始图的邻居信息。进行图嵌入的重点在于对编码器的求解。编码器是一个函数，对图中任意一个节点$v \in V$，映射出嵌入表示向量$z_v \in R^d$，编码器可以被表示为$ENC: V \to R^d$，如图 1-1 所示：

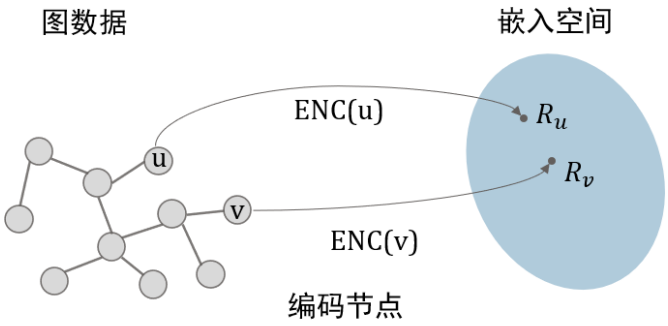


图 1-1 图节点嵌入原理图

1.2.3.1 静态图的节点表示学习

静态图的节点表示学习主要可以分为两种：一种是基于随机游走的方法，另一种是基于神经网络的方法。

基于随机游走的方法包括 DeepWalk[18]，LINE[19]和 Node2vec[20]，这些工作的不同之处在于对结构相似性的定义。DeepWalk 来源于 Word2Vec，选择一个特定的节点作为起始节点，使用随机游走来获得节点的有序序列，将序列作为一个句子，用 Word2Vec 学习得到节点的表示向量。LINE 定义了节点之间的一阶相似度和二阶相似度，一阶相似度由节点之间的边确定，当两个节点之间的边权重越大，它们的一阶相似度越高；二阶相似度指两个节点邻居网络结构的相似性。这一方法能够较好地运用在大型网络中。Node2vec

通过引入两个参数来平衡广度优先搜索和深度优先搜索改善随机游走。SDNE[21]利用自编码器学习节点表示。基于随机游走的方法由于其易传性，难以处理不可见节点，且它们通过对节点和邻域的序列化，将其转化为类似于文本的线性结构，主要集中在结构相似性的保留上，并没有直接处理图的结构。

基于神经网络的方法主要集中在图神经网络（Graph Neural Network）上，其主要思想是传播和聚合图中相邻节点的信息。受到计算机视觉领域卷积网络的启发，针对图数据的卷积网络也不断被提出， 2013 年，基于图论的图卷积神经网络[22]被提出，图卷积神经网络开始吸引大量学者关注。图卷积网络可以分为两大类：基于谱的和基于空间的。基于谱的图卷积神经网络包括 ChebyNet[23]、GCN[24]。基于空间的图卷积神经网络包括 GraphSAGE[25]，通过学习一个对邻居节点进行聚合表示的函数来产生中心节点的特征表示。GAT[26]引入注意力机制，给节点之间的边给予重要性，解决图神经网络聚合邻居节点的时候没有考虑到不同的邻居节点重要性不同的问题。

1.2.3.2 动态图的节点表示学习

对于关系随着时间而改变的动态图，如何对其的演变特征进行建模并支持增量更新模型参数仍然是待解决的问题。动态图可以被分为连续时间动态图（Continuous Time Dynamic Graph，简称 CTDG）和离散时间动态图（Discrete Time Dynamic Graph，简称 DTDG）。和连续时间动态图相比，离散时间动态图是从动态图中选取的快照序列，可能会损失一部分信息。目前用于 CTDG 的嵌入模型能够被 DTDG 使用，但是反之则不可。本论文主要研究的是针对离散时间动态图的嵌入方法，以下具体介绍离散时间动态图的节点表示学习方法。

在离散时间动态图的嵌入方法中，将动态图按照时间进行聚合是一种较为简单的方法，聚合的函数包括直接求和[27]、对最近的快照给予更大的权重再进行求和、对图特征进行聚合等。除此之外， Liu 等人利用 DTDG 的时间作为正则化器[28]，在连续快照上对每个节点的嵌入施加平滑度约束，从而将表示学习问题转化成限制下的最优化问题。Sajjad 等人[29]提出了一种算法，用于在重用前一个快照中的有效随机游走的同时，为新快照生成无偏随机游动。NetWalk[30]采用相似的方法，在计算节点的向量表示时最小化每个随机游动中节点之间的成对距离，使得网络嵌入能够动态更新，实现实时检测。DynamicTriad[31]研究了三元闭包过程，通过三元闭合过程使模型能够捕捉网络的动态，在不同的时间学习嵌入向量。DynGEM[32]利用 SDNE 的模型架构，使用动态自编码模型，它能够使嵌入随着时间推移保持稳定，能处理不断增长的动态嵌入图。dyngraph2vec[33]利用通过自编码器结构重构下一个邻接矩阵以获取当前图的嵌入表达，EvolveGCN[34]使用 RNN 进化 GCN 参数来捕捉图序列的动态性,在时间维度上适应了图卷积神经网络模型，而不依赖于节点嵌入。

1.3 研究问题与挑战

论文主要研究如何对以太坊数字加密货币交易平台的网络钓鱼账户进行异常检测。在以太坊的数字加密货币交易网络中，账户的个人资料信息缺乏让钓鱼账户的检测成为了一个棘手的问题。交易网络数据的特性使得钓鱼节点检测面临以下的挑战：

1. 节点数据较大。交易网络的节点数量巨大，如果考虑整个图的拓扑结构，将面临时间和空间上的限制。

2. 节点无特征。交易网络中的账户和交易可以被看作节点和加权的有向边，但账户并没有任何具体信息，即图中节点并不存在自身特征，需要利用已知的图的结构信息对节点进行特征工程，提取出节点特征。

3. 正负样本不均衡。标记为异常钓鱼节点的数量远小于正常节点的数量，这增加了对钓鱼账户的识别难度。

为实现对网络钓鱼节点的异常检测，需要对数字加密货币交易网络中挖掘节点嵌入表示。现在的图嵌入的相关工作中，目前的静态网络图嵌入模型缺乏对时序信息的利用，而动态图嵌入模型通常在学习节点嵌入表示时仅考虑节点之间的一阶相似性，并未考虑更高阶的相似性，对图的空间拓扑信息利用不足。针对以上的挑战和相关工作的不足，论文将重点研究如何结合图数据中的高阶邻居信息和时序信息来完成动态图的嵌入表征。

1.4 论文工作

本论文为了实现更好的图节点嵌入，考虑把空间和时间两个维度上的信息进行结合。在空间上，利用图卷积神经网络，能够学习融合交易网络的高阶邻居信息和节点本身的特征；在时间上，由于交易图是一个不断变化的动态图，它的节点和链接会不断地增加，将时序信息引入图表示学习中也是尤为重要的。

本论文整体工作流程如图 1-2 所示，主要工作与贡献有：

1. 提出并实现了一种基于图卷积神经网络和循环神经网络结合的无监督的动态图表示学习算法，命名为基于图卷积循环的动态图嵌入模型（Dynamic Graph Convolutional Recurrent Embedding Method，简称 DynGCREM）。在图卷积神经网络层中，利用图卷积网络提取潜在的深层结构特征表示；在循环神经网络层，使用长短期记忆网络和自动编码的框架相结合，学习时序信息，最终学习图节点的向量表示，并用于下游任务。

2. 将基于图卷积神经网络的图嵌入方法引入以太坊的数字加密货币交易网络的场景中，图卷积网络能够更好地聚合以太坊的数字加密货币交易网络的节点特征和结构特征，为节点分类的任务提供了一种特征工程的方案。

3. 在以太坊的交易网络数据集中实现了图节点嵌入表征及分类的任务，并将 DynGCREM 模型与动态图嵌入算法进行对比分析；对模型进行消融实验，探究模型各组

件功能，实验验证了模型的有效性。最后对实验结果进行了讨论和解释。

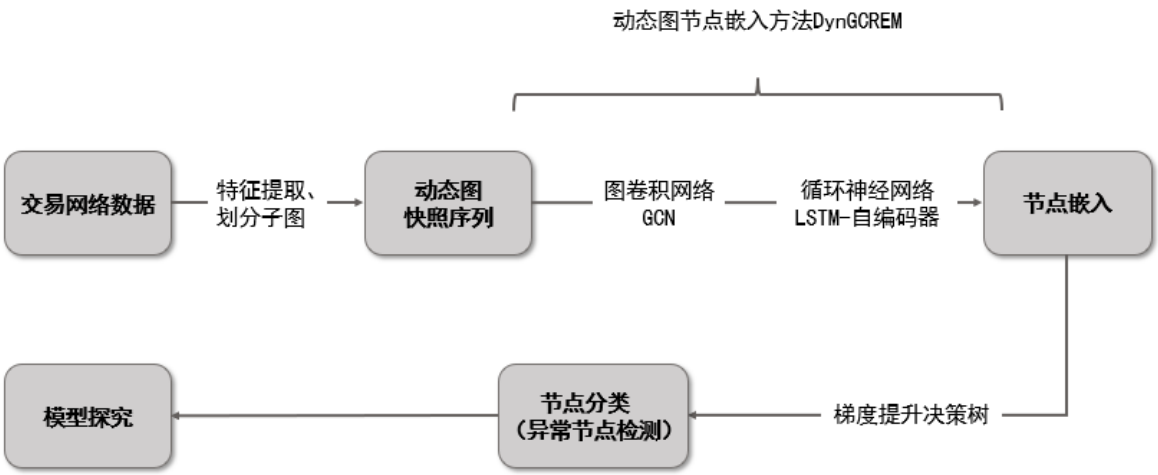


图 1-2 论文工作流程图

1.5 论文结构介绍

论文总共分为五个章节，具体内容结构如下：

第 1 章介绍了论文的选题背景、国内外研究现状，并描述了论文工作的动机、贡献和整体结构。

第 2 章阐述论文所使用的相关技术和算法，对相关工作进行进一步的补充和阐述。

第 3 章定义具体研究的问题，并详细介绍论文提出的模型 DynGCREM。

第 4 章介绍数据集的处理方法和实验设置，分析模型在数据集上与其他方法对比的结果，研究影响模型性能的因素。

第 5 章总结论文的主要工作，指出不足和未来可能的改进措施。

2 相关技术基础

论文提出的动态图嵌入方法使用了图卷积神经网络和循环神经网络，并利用自编码的思想加以改进，本章首先介绍这两种神经网络和自编码器的原理。为了评估模型对节点嵌入的性能，实现对异常钓鱼账户节点的检测，论文使用梯度提升决策树完成分类任务，本章将介绍梯度提升决策树的相关原理。

2.1 图卷积神经网络

卷积神经网络在图像和视频领域取得了较为显著的效果，它具有局部连接、权值共享等特点。与图像数据不同的是，图数据是非欧式结构数据，传统的卷积神经网络仅能处理排列整齐的矩阵或序列，即欧几里得结构数据，并不能直接使用卷积神经网络的卷积操作提取有效的特征。图卷积网络能够同时对节点特征信息和图的拓扑结构信息进行的学习，被证明是最高效的图神经网络之一。

图卷积神经网络主要分为两类，一类基于空间域，另一类基于谱域。空域卷积神经网络直接对图进行卷积操作，类似图像处理中对像素点直接进行卷积。基于谱的图卷积神经网络从图信号处理的角度引入滤波器来定义图卷积，它将传统的卷积算法运用在图结构数据的处理上，关注以某一结点为中心，K 阶邻居之内的信息，主要方法包括 ChebyNet、GCN。ChebyNet 使用多项式展开近似计算图卷积，GCN 进一步简化了 ChebyNet 的运算，只覆盖一阶邻居节点，是一种更加简单的图卷积网络。论文主要运用的是基于谱域的图卷积神经网络，因此在下文将着重介绍 GCN。

一些图信号处理领域的学者定义了图的傅里叶变换，并进一步定义了在图上的卷积操作，并最终提出了图卷积网络（Graph Convolutional Networks，简称 GCN）。GCN 的原理是基于拉普拉斯矩阵的谱分解，通过图卷积运算可以同时聚合拓扑结构信息和节点属性信息。为了能够在图上进行卷积运算，传统的傅里叶变换和卷积操作需要在图上进行重新定义，在此首先引入图的拉普拉斯矩阵 L，L 是通过图的两个矩阵进行计算得到：

$$L = D - A \tag{2.1}$$

其中D表示度矩阵，A是图邻接矩阵。拉普拉斯矩阵是一个半正定矩阵，能够进行特征值分解，得到特征值Λ和特征向量矩阵U，如（2.2）式所示：

$$L = U\Lambda U^T \tag{2.2}$$

图的傅里叶变换（Graph Fourier Transform，简称 GFT）将图信号从空域的视角转向频谱视角，是图信号处理理论体系的基础，图上的傅里叶变换就是图的拉普拉斯矩阵和特征向量的乘积。对于图上的任意一个信号，图的傅里叶变换能找到一组正交基，这组正交基的线性组合能够表示x，并且基函数使用拉普拉斯矩阵的特征向量。因此对于任意一个在图 G 上的信号x，其图傅里叶变换为：

$$\hat{x}_k = \sum_{i=1}^{N} U_{ki}^T x_i \tag{2.3}$$

在介绍图的卷积之前，需要先引入卷积的数学定义：

$$(f * h)(t) = \int_{-\infty}^{+\infty} f(\tau)h(t-\tau)d\tau \tag{2.4}$$

在信号处理领域中，卷积定理说明函数卷积的傅里叶变换是函数傅里叶变换的乘积。因此时域中复杂的卷积计算可以被转化成频域中简单的相乘运算。如果用f表示图上的 N 维向量，h表示图卷积的卷积核，由卷积定理，两者的卷积可以通过其傅里叶变换的乘积再进行逆变换表示，即：

$$f * h = F^{-1}\left[\hat{f}(w)\hat{h}(w)\right] = \frac{1}{2\pi}\int \hat{f}(w)\hat{h}(w)e^{iwt}dw \tag{2.5}$$

f经过傅里叶变换后，由(2.3)式可得$\hat{f}$=$U^T f$，令h的傅里叶变换形式为$\hat{h}(\lambda)$，带入到(2.5)式后，可得：

$$(f * h) = U\hat{h}(\lambda)U^T f \tag{2.6}$$

其中$\hat{h}(\lambda)$是卷积核参数,是一个对角矩阵，Bruna[22]将$diag\left(\hat{h}(\lambda)\right)$设置为$diag(\theta)$。由此 GCN 的卷积层被定义为：

$$y = \sigma(Udiag(\theta)U^T x) \tag{2.7}$$

$diag(\theta)$可以被理解为图滤波器，设$g_\theta = diag(\theta)$,则图卷积层还能够被定义为信号x与滤波器g_θ相乘：

$$g_\theta * x = Ug_\theta U^T x \tag{2.8}$$

但这种方式计算复杂度较大，在图节点数量较多的情况下会达到$O(n^2)$。Defferrard[23]等提出了 K 阶 Chebyshev 多项式作为卷积核，Chebyshev 多项式被递归地定义为：

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \tag{2.9}$$

最初的两项$T_0(x)$和$T_1(x)$分别是 1 和 x，使用 k 阶的 Chebyshev 多项式T_k(x)近似$g_\theta(\Lambda)$:

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k\left(\widetilde{\Lambda}\right) \tag{2.10}$$

其中$\widetilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$, λ_{max}是L的最大的特征值。θ'是一个切比雪夫系数的向量。

Kipf 和 Welling[24]通过谱图卷积的局部一阶近似确定卷积网络结构，卷积公式为：

$$g_{\theta'} * x \approx \sum_{k=0}^{K} \theta'_k T_k\left(\tilde{\mathrm{L}}\right)x \tag{2.11}$$

其中$\tilde{\mathrm{L}} = \frac{2}{\lambda_{max}}\mathrm{L} - I_N$，此方程的复杂度为$O(|\varepsilon|)$，图卷积框架如图 2-1 所示。

图卷积神经网络由堆叠多层的卷积层形成，若卷积层数 k=1，则有：

$$g_{\theta'} * x \approx \theta'_0 T_0\left(\tilde{\mathrm{L}}\right)x + \theta'_1 T_1\left(\tilde{\mathrm{L}}\right)x = \theta'_0 x + \theta'_1 \tilde{\mathrm{L}}x \tag{2.12}$$

取近似$\lambda_{max}=2$，则有：

$$g_{\theta'}*x\approx\theta_0'x+\theta_1'(L-I_N)x=\theta_0'x-\theta_1'D^{-\frac{1}{2}}AD^{-\frac{1}{2}}x \tag{2.13}$$

为避免过拟合问题，约束参数的数量和最小化每一层的操作数，有：

$$g_{\theta'}*x\approx\theta(I_N+D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x \tag{2.14}$$

在（2.14）式中，$I_N+D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$的特征值在[0,2]范围内，如果重复操作运算该式，将会造成梯度消失或梯度爆炸的问题，因此，引入归一化技巧，用$\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$替代$I_N+D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$，其中$\tilde{A}=A+I_N$，即自连接和无向图邻接矩阵相加，$\widetilde{D}_{ii}=\sum_j\tilde{A}_{jj}$，表示$\tilde{A}$的度矩阵。

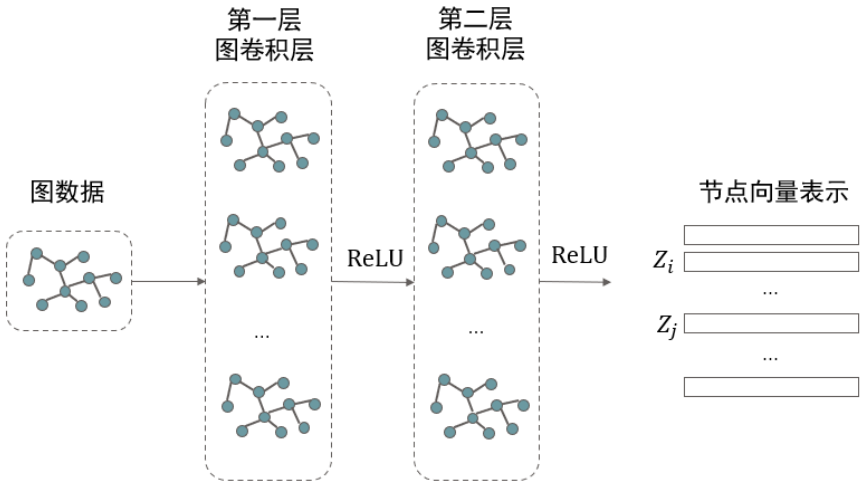


图 2-1 图卷积框架

2.2 循环神经网络

2.2.1 标准循环神经网络

循环神经网络（Recurrent Neural Network，简称 RNN）在 1986 年首次被提出，是一种对时序数据进行建模的神经网络。RNN 能够使用状态（记忆）来处理可变长度的序列，它能够挖掘数据中时序信息。

图 2-2 展示了一个通用的 RNN 模型，图 2-3 展示了每一个 RNN 单元的内部结构。每个 RNN 单元都是一个简化的前馈神经网络，并向下一个单元传递信息。神经网络的激活函数通常使用 Sigmoid 函数。

基础的 RNN 仍然存在问题，由于 BP 算法和长时间依赖，可能会出现梯度消失或者梯度爆炸的问题。因此出现了一系列改进的算法，较为经典的有长短期记忆网络和循环门控单元。

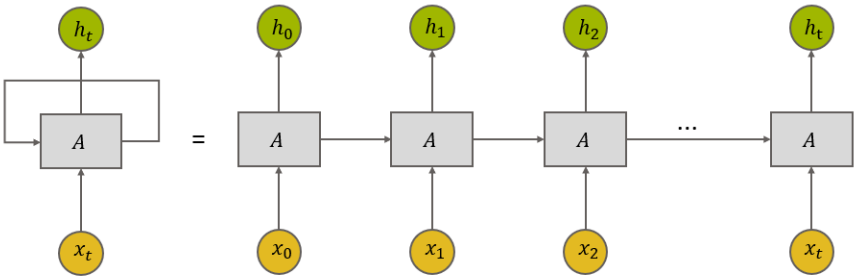


图 2-2 循环神经网络结构图

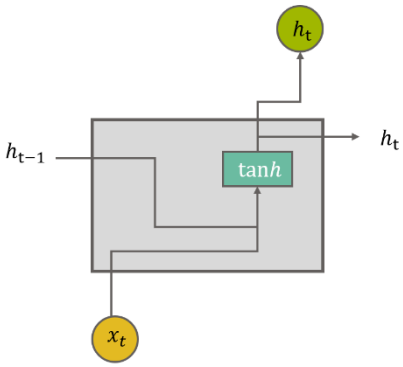


图 2-3 RNN 单元结构图

2.2.2 长短期记忆网络

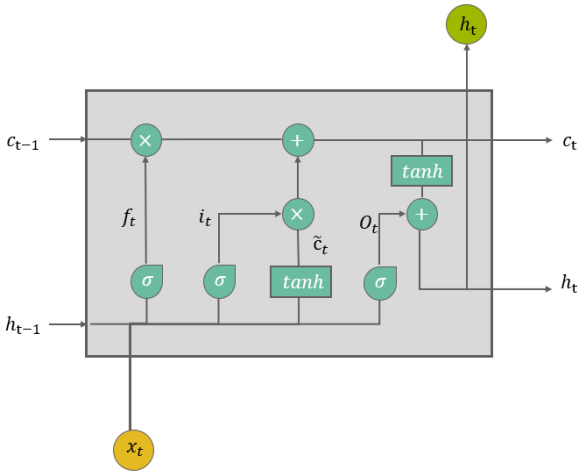


图 2-4 LSTM 单元内部结构图

长短期记忆网络（Long Short-Term Memory Network，简称 LSTM）不仅考虑前一时刻的输入，而且赋予了网络的记忆功能，它的核心在于细胞的状态，它能够通过门结构来添加或移除细胞状态信息。它加入了 Sigmoid 层，该层输出 0 至 1 之间的概率值，描述每个部分有多少量可以通过。其中 0 表示不允许任务变量通过，1 表示允许所有变量通过。LSTM 可用于在各种通用序列建模任务中建模长期依赖性。

图 2-4 展示了一个 LSTM 单元的内部结构，操作⊗ 和⊕ 分别表示点乘和加法。一个LSTM单元有输入门、输出门和遗忘门三个门控，控制着进出细胞的信息流，建立了内部状态的自循环。输入门能够控制当前时间步是否输入并决定哪些信息保存进长期记忆，通过前一个时间步的系统状态对内部状态进行更新。输出门能够确定哪一部分需要被输出。遗忘门能够确定哪些信息被遗忘，通过前一个时间步的内部状态更新当前时间步的内部状态。

LSTM 的公式为：

$$\begin{cases} i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ \widetilde{c_t} = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \\ c_t = f_t \otimes c_{t-1} + i_t \otimes \widetilde{c_t} \\ o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t = o_t \otimes tanh(c_t) \end{cases} \tag{2.15}$$

其中x_t是 LSTM 单元的输入向量，i_t表示输入门，它会决定有多少信息加入记忆信息流，f_t表示遗忘门，能决定循环神经网络丢弃之前的信息。$\widetilde{c_t}$是记忆状态的在 t 时刻生成的新的候选信息，c_t相当于循环神经网络的记忆，它的更新由输入门和遗忘门共同决定。o_t是输出门，它决定有多少记忆被用于下一阶段的更新，它的取值在 0 到 1 之间。

2.3 自编码器

自编码器（Autoencoder，简称 AE），是一种无监督学习的神经网络，它使输出值等于输入值，输出值相当于输入值的重构。输入值将会被压缩成潜在空间表征，输出通过潜在空间表征来进行重构。自编码器的结构如图 2-5。

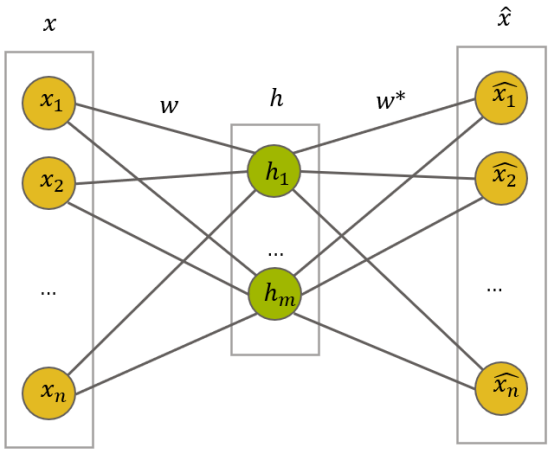


图 2-5 自编码结构图

编码器部分可以表示为编码函数$h(x)$，函数的输出即为潜在空间的表征：

$$h(x) = f(W^T x + b) \tag{2.16}$$

其中x是输入层，f是激活函数，b是偏置。h表示隐藏层。

解码器部分可以表示为解码函数$g(h)$，通过解码器能重构来自潜在空间表征的输入。其中$\hat{x}$是解码器重构的输出，它的结构和编码器的输入层一致，可表示为：

$$\hat{x} = f((W^*)^T g(x) + c) \tag{2.17}$$

其中$W^* = W^T$。

自编码器结构能够无监督地将输入层数据x转换成隐藏层中的嵌入表示形式h，它通过计算重构x的误差，尽可能地让输出更接近输入，因此自编码器的训练目标就是最小化损失函数$L_{loss} = \frac{1}{2}\sum_{i=1}^{N}(x_1 - \widehat{x_1})^2$。通过模型的训练，能够得到（2.16）和（2.17）式中最优的W、b、c。

由于自编码器结构能够学习到 x 在隐藏层 h 的编码表示，根据输入层和隐藏层的大小，自编码网络可以被分为两种不同的类型：欠完备自编码器（undercomplete autoencoder）和过完备自编码器（overcomplete autoencoder）。欠完备自编码器的隐藏层神经元个数比输入层的少，这个过程相当于一种非线性的降维操作，在尽可能地保留原始信息的前提下以更小的维度来表征原始信息。过完备自编码器的隐藏层神经元个数大于输入层，通常用于获取稀疏的特征表示。

2.4 梯度提升决策树

提升（Boosting）是一种将弱学习算法提升为强学习算法的方法，在训练时，它会反复修改训练数据的权值分布，先构建多个弱分类器，然后再将弱分类器线性组合，构成强分类器，在不断的迭代中进行优化。

Boosting 的方法主要有两种，包括 AdaBoost 算法和梯度提升（Gradient Boost）算法。AdaBoost 按照当前样本的误差大小来调整权重，Gradient Boosting 调整模型拟合的目标为当前的样本残差。

决策树是应用在数据挖掘领域的分类任务的一种基本方法。梯度提升决策树（Gradient Boosting Decision Tree，简称 GBDT）是将梯度提升的方法运用于决策树中，即梯度提升过程的每一个子模型都是决策树。每一棵决策树学习之前所有树的结论和残差，拟合得到当前的残差和决策树。具体流程描述如下：首先在初始训练阶段，训练样本建立弱分类器。通过利用梯度计算公式计算出残差减小的方向，以此更新决策树，不断迭代该过程，直到迭代次数上限，最终将多个弱分类器转换为强分类器。

相较于需要复杂调参的神经网络，GBDT 的参数数量少、便于调参，而且它对输入数据的要求较低，不限制模型输入特征向量的值。实现 GBDT 的工具包括 XGBoost、LightGBM 等。

3 基于图卷积循环的动态图嵌入模型

3.1 问题构建

在描述论文的模型方法之前，先对论文研究的问题进行具体的定义，阐明论文目标。

3.1.1 基本定义

定义 3.1（动态图）：动态图G可被表示为一组快照（snapshot），用一组序列进行表示：$S = G_0, G_1 \ldots G_T$，G_t表示t时刻的动态图的状态镜像，即一个图快照。t时刻的图快照包含该时刻图的节点信息和边信息，用三元组(V, E_t, A_t)表示。V表示图的节点集合，E_t表示t时刻的边的集合，A_t表示t时刻的节点属性（如果节点属性不存在则不存在该集合）。

定义3.2（动态图表示学习）：动态图中的快照序列中，设节点为v，则利用算法学习一系列节点嵌入向量：R_{v1}、$R_{v2}\ldots R_{vT}$。R_{vt}表示节点v在t时刻的低维稠密向量表示，即节点嵌入。动态图表示学习的目标是学习嵌入函数$f(t)$，将图中的节点映射到潜在空间中，获得节点嵌入向量。嵌入向量包含了原始动态图中的结构特征和时间特征。

定义 3.3（节点分类）：图$G = (V, E)$中，V表示图的节点集合，E表示边的集合。图 G 可以是静态图或动态图的任一图快照。图节点有两种信息：特征X和类别（标签）Y。V可被分为有标签的集合和无标签的集合，节点分类的目标是使用有标签集合训练分类器f，来推断无标签节点的类别，在无标签的集合进行测试。

3.1.2 目标

模型的目标是给定动态图G，获取图中节点的低维节点嵌入表示。通过映射获取到的节点嵌入表示向量R_T将用于节点分类的任务中验证其有效性。

3.2 DynGCREM 模型

本节将详细阐述 DynGCREM 模型的具体结构和原理。DynGCREM 的整体框架如图 3-1 所示，可以被分为两个组件：图卷积网络层和循环神经网络层。图卷积网络层从动态图中负责学习节点之间的空间拓扑信息和高阶相似性特征，循环神经网络层负责学习动态图的演化信息。对每一个节点k，模型能够学习在每一个快照的嵌入表示R_{k_1}、$R_{k_2}\ldots\ldots R_{k_T}$。$R_{k_t}$表示节点$k$在时刻$t$的节点向量表示，它包含$t$时刻之前的动态图演化信息。

模型的图卷积网络层中，GCN 会通过重构邻接矩阵，得到每一个时刻每个节点的嵌入向量，节点k在时刻t的节点向量表示Z_{k_t}，即输出一个形状为（采样节点数，时间间隔，嵌入向量长度）的张量。模型的循环神经网络层中，输入为节点表示向量和原始特征数据的合并，通过重构输出，LSTM 自编码器能够计算出图中节点的嵌入向量R_{k_T}。

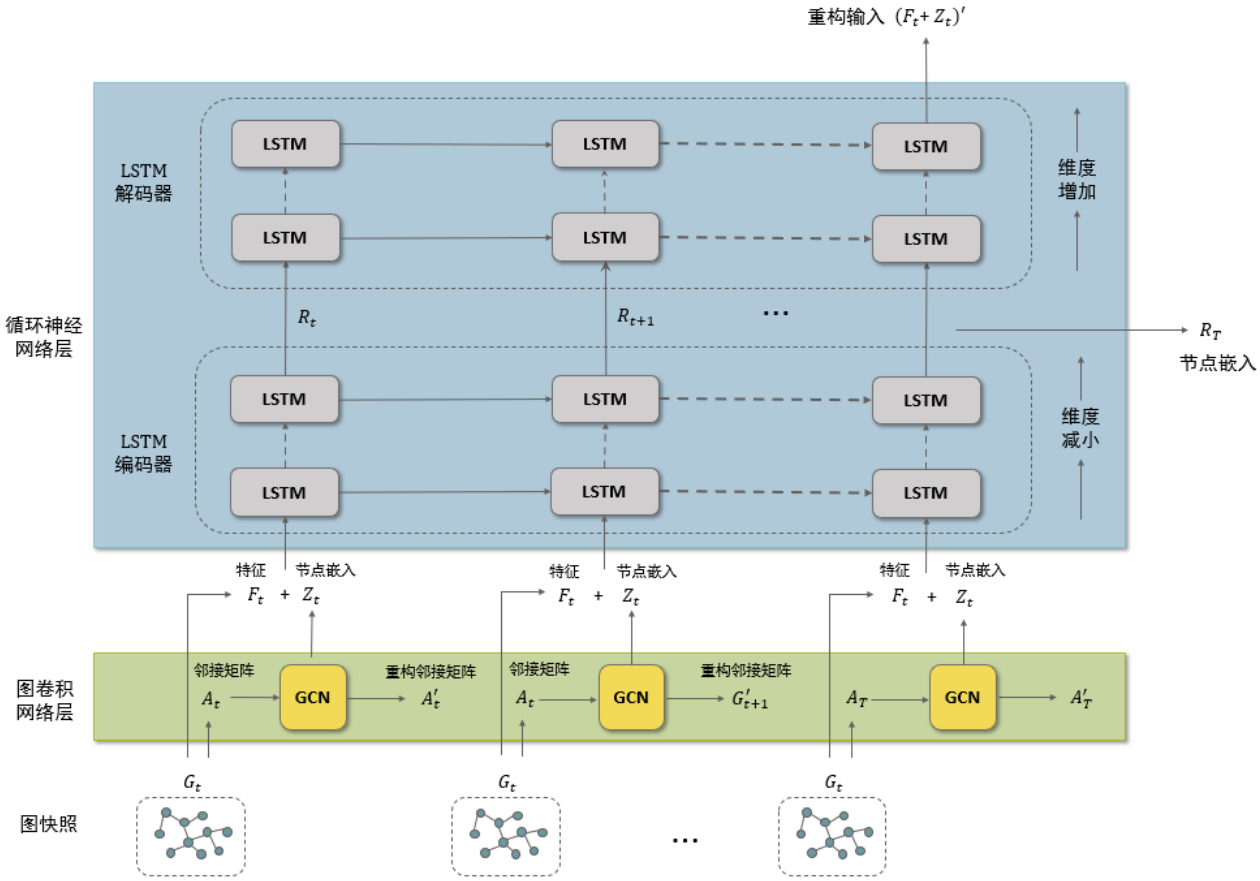


图 3-1 DynGCREM 模型整体框架图

3.2.1 图卷积网络层

在图卷积网络层中，每个时刻的图快照都会被用于训练图卷积网络，通过无监督的方式，图卷积神经网络 GCN 学习节点的空间拓扑信息。

GCN 有如下的每一层的传播公式：

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{3.1}$$

其中，$\tilde{A} = A + I_N$，A是无向图的邻接矩阵，I_N是单位矩阵，$\tilde{D}$是$\tilde{A}$的度矩阵。图的拉普拉斯矩阵为 L，满足：

$$L = I_N - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} = U\Lambda U^T \tag{3.2}$$

通过特征值分解能够得到特征向量矩阵U和特征值Λ。模型的第一层$H^{(0)}$是输入 X，获取 1 阶邻域的节点信息。第 K 层 GCN 模型的卷积层能够采样 K 阶邻域的所有信息节点。

如果使用单层的卷积层，图卷积层的输出 Z 如下：

$$Z = \sigma\left(\tilde{A}XW^{(0)}\right) \tag{3.3}$$

其中X是输入的节点特征，Z 就是节点的一阶嵌入表示，使用 *tanh* 函数作为激活函数。为了使整个过程无监督，利用重构邻接矩阵的误差作为损失函数，令$\tilde{A} = sigmoid(ZZ^T)$，表示邻接矩阵的估计值，损失函数定义为：

$$L_{GCN} = \frac{\| A - \tilde{A} \|_F^2}{n} \tag{3.4}$$

$\| \|_F^2$表示二范数的计算，ZZ^T是通过 Z 重构出来的图结构，也就是一个 decoder 的过程。因此整个过程是一个无监督的过程。如果通过 decoder 重构出来的邻接矩阵和原始邻接矩阵的相似度越高，也就意味着图节点的嵌入保留了更多原始的图的结构信息，图嵌入的表示效果就更好。

经过图卷积层，模型将获取图快照中节点的嵌入向量，这一向量将作为循环神经网络层的输入。

3.2.2 循环神经网络层

图神经网络层输出的嵌入向量 Z 和原始特征合并后的向量将作为循环神经网络层的输入，对应新的嵌入向量为该部分的输出。循环神经网络层生成嵌入时，采用 LSTM 自编码器的结构，具体原理如图 3-2 所示：

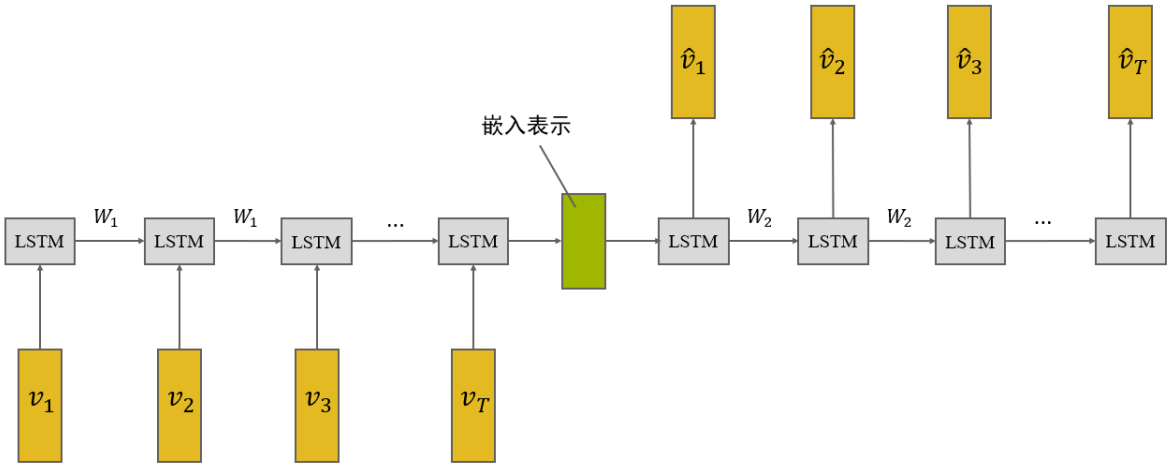


图 3-2 LSTM-自编码器原理图

LSTM 自编码器由对称的 LSTM 单元组成，在编码器阶段，LSTM 会输入每个图快照中由 GCN 组件生成的节点的嵌入向量与原始特征向量的合并，节点v在G_t快照的节点嵌入为Z_{v_t}，与原始特征向量合并后的新特征记为$Z_{v_t}^1$。将$Z_{v_t}^1$输入编码器，将会输出学习到的嵌入向量，该嵌入向量是解码器的输入，解码器再输出$\widehat{Z_{v_t}^1}$。同时训练解码器和编码器，最后将编码器保留，编码器的输出作为节点最终的嵌入向量R_{v_t}。每一个G_t的$Z_{v_t}^1$都会作为 LSTM 自编码器的输入，模型的隐藏层就是嵌入向量R_{v_t}。该嵌入向量就是动态图嵌入模型中对动态图节点嵌入表示。

节点v在 t 时刻时，第一层 LSTM 层的定义如下：

输入门为：

$$i_{v_t}^{(1)} = \sigma\left(W_i^{(1)} \cdot \left[h_{t-1}, Z_{v_t}^1\right] + b_i^{(1)}\right) \tag{3.4}$$

遗忘门为：

$$f_{v_t}^{(1)} = \sigma\left(W_f^{(1)} \cdot \left[h_{t-1}, Z_{v_t}^1\right] + b_f^{(1)}\right) \tag{3.5}$$

c_t表示 LSTM 的细胞状态，有：

$$\widetilde{c_{v_t}}^{(1)} = tanh\left(W_c^{(1)} \cdot \left[h_{t-1}, Z_{v_t}^1\right] + b_c^{(1)}\right) \tag{3.6}$$

$\widetilde{c_t}$是记忆状态的在 t 时刻生成的新的候选信息，有：

$$c_{v_t}{}^{(1)} = f_{v_t}^{(1)} \otimes c_{v_{t-1}}{}^{(1)} + i_{vt}^{(1)} \otimes \widetilde{c_{v_t}}^{(1)} \tag{3.7}$$

输出门为：

$$o_{v_t}{}^{(1)} = \sigma\left(W_o^{(1)} \cdot \left[h_{t-1}, Z_{v_t}^1\right] + b_o^{(1)}\right) \tag{3.8}$$

隐藏状态为：

$$h_{v_t}^{(1)} = o_{v_t}^{(1)} \otimes tanh(c_{v_t}{}^{(1)}) \tag{3.9}$$

在编码器中第一层的 LSTM 单元状态和隐藏层表示通过 LSTM 的网络链传递至下一层，因此可以将节点 v 在 t 时刻里第 k 层 LSTM 层定义如（3.10）：

$$\begin{cases} h_{v_t}^{(k)} = o_{v_t}^{(k)} \otimes tanh\left(c_{v_t}{}^{(k)}\right) \\ o_{v_t}{}^{(k)} = \sigma\left(W_o^{(k)} \cdot \left[h_{v_{t-1}}^{(k)}, h_{v_t}^{(k-1)}\right] + b_o^{(k)}\right) \end{cases} \tag{3.10}$$

LSTM 自编码网络的损失函数可以被定义为输入向量的重构误差，损失函数使用均方误差（Mean Square Error）进行计算：

$$L_{LSTM-auotoencoder} = \frac{1}{2}\sum_{i=1}^{N}\left(Z_{v_t}^1 - \widehat{Z_{v_t}^1}\right)^2 \tag{3.11}$$

3.2.3 Mini-batch 学习

模型在实验中使用 Mini-batch 的方法训练学习模型的各项参数。Mini-batch 方法需要在训练前先设置参数批处理大小（batch size），在每次训练中随机选取大小等于批处理大小的样本进行训练，按照批来进行梯度下降，更新参数。本模型最后应用了 Adam 优化器计算目标函数。最小批量的学习能够减少梯度下降的随机性，所需时间更短。

3.3 小结

本章定义了论文具体研究的问题：在给定的动态图中获取图中节点的低维节点嵌入表示，针对该问题提出了一种基于图神经网络和循环神经网络的动态图嵌入方法 DynGCREM，本章详细对模型的组成成分进行介绍。

4 实验与结果分析

4.1 数据处理

4.1.1 数据集介绍

论文使用的数据集来源于以太坊公共交易平台[1]。在该平台上能获取到被标记的钓鱼诈骗节点，根据节点进行两层广度优先搜索，可获得包含钓鱼诈骗交易网络的数据集。数据集的时间跨度为 2015 年 8 月 7 日至 2019 年 1 月 19 日，包含 2,973,489 个账户节点和 13,551,303 条交易记录。数据集中包括 1,165 个被标记为钓鱼账户的异常结点。交易网络数据集中，两个相同的账户可能有多个交易，因此节点之间可能有多条有向的带权边，表示节点之间的交易，边的方向表示汇款的方向，权重包含数字货币具体交易的信息：数字货币的交易金额和交易时间。

4.1.2 节点采样

算法 1 随机游走节点采样

输入：无向图G，采样节点数L，起始节点v_{src}

输出：采样的节点集合S

$S \leftarrow set(v_{src})$ //将v_{src}加入S

$U \leftarrow v_{src}$

$\boldsymbol{while} len(S) < L \boldsymbol{do}$

    $v_{cur} \leftarrow U$

    $\boldsymbol{if} len(v_{cur}) > 0 \boldsymbol{then}$//$v_{cur}$有邻居节点

        $v_{next} \leftarrow v_{cur}$的一个随机邻居节点

        $S.add(v_{next})$ //将v_{next}加入S

        $U \leftarrow v_{next}$

    $\boldsymbol{end}$

    $\boldsymbol{else}$

        $\boldsymbol{break}$

    $\boldsymbol{end}$

$\boldsymbol{end}$

$\boldsymbol{return}\ S$

由于数据集的规模很大，需要对原始数据集采样出较小的数据集作为实验对象。为

[1] https://ethereum.org/en/

了保证采样后的数据集与原始数据具有相同的分布，采用随机游走邻域关系的方法获得采样子图。由于交易行为（即图中的边）是有向的，而邻居关系是无向的，因此在采样时，整个交易网络会被视为一个无向图，在这一阶段，有向边将会直接转换为无向边。随机游走的算法如算法 1 所示。

根据该算法，将采样的节点数分别设置为 30000，40000 和 50000，通过随机游走采样出三个节点集合，节点集合以及它们之间的边形成了三个子图，分别记为 S1、S2、S3。具体子图的统计信息如表 4-1 所示：

表 4-1 采样子图统计结果

子图名称	节点个数	钓鱼节点个数	边数	平均度数
S1	30000	113	1,140,091	76.0061
S2	40000	134	1,292,279	64.6140
S3	50000	172	1,418,893	56.7557

4.1.3 特征提取

在特征提取阶段，将采样得到的子图进行特征工程，提取节点的基本特征。由于数据集中并没有针对每个节点本身的具体特征，因此需要根据节点之间的关系提取出代表节点的特征。在特征提取过程中，使用 4.1.2 节中提取出的子图的有向图进行特征工程。

具体采用的特征如下：

FT1：入度。表示当前节点接收到的交易总数。

FT2：出度。表示当前节点发送给其他节点的交易总数。

FT3：总度数。表示当前节点的所有边，即总交易次数。

FT4：汇入总量。表示发送到当前节点的 ETH 总量，通过对以当前节点为终点的所有的边的权重求和进行计算。

FT5：汇出总量。表示当前节点发送到相邻所有节点的 ETH 总量。通过对以当前节点为起点的所有的边的权重求和进行计算。

FT6：交易总量。表示当前节点参与的所有交易的 ETH 总量。通过 FT4 和 FT5 求和计算。

FT7：邻居节点数。表示当前节点的邻居节点数量。邻居节点数和度数（FT3）在图中并不相等，因为两个账户之间可能有不止一次交易，因此度数将会大于等于邻居节点数。

FT8：平均交易频率的倒数。交易频率越高时，该值越小。计算方式为当前节点最早和最晚的交易之间的时间差除以 FT3。

以节点数为 40000 的子图 S2 为例，对上述的特征进行统计，得到表 4-2。

在表 4-2 中，负样本是未标记为钓鱼账户的节点。正样本是被标记为钓鱼账户的节点。

从表中可以看出，平均而言，负样本的 FT1（入度）较高，而正样本的 FT1 较均匀。正样本的 FT2 值（出度）相比其 FT1 值更大。钓鱼账户通常会大量收到受害者的汇款，该项的统计结果比较违反直觉，一种可能的假设是钓鱼账户向辅助账户分散资金来减小钓鱼账户的可疑度。负样本的 FT3（总度数）非常不均匀，这可能是因为未标记的节点中构成可能较为复杂，负样本除了可能是正常节点，还有可能是一些用于智能合约和交换的节点，由于角色的不同使得负样本的总度数不均匀。

表 4-2 节点特征统计

a. FT1-FT3 统计信息

特征		FT1	FT2	FT3
均值	负样本	35.49	35.45	70.95
	正样本	30.96	43.60	74.57
50%	负样本	0	2	3
	正样本	10	3	31
75%	负样本	2	9	15
	正样本	17	52	76
最大值	负样本	77,330	84,085	84,413
	正样本	792	666	792

b.FT4-FT8 统计信息

特征		FT4	FT5	FT6	FT7	FT8
均值	负样本	417.04	417.12	834.17	15.16	666,221.91
	正样本	35.84	12.12	47.96	36.90	148,647.80
50%	负样本	0.00	0.00	0.02	2	85,383.78
	正样本	8.07	0.09	11.41	18	50,048.97
75%	负样本	0.30	0.20	2.00	9	740,406.25
	正样本	22.42	3.82	37.16	44.75	160,872.39
最大值	负样本	7.92E+6	7.92E+6	8.27E+6	1376	31,414,808.50
	正样本	6.75E+2	4.03E+2	7.43E+2	657	2,025,817.50

在 FT4（汇入总量）、FT5（汇出总量）和 FT6（交易总量）这三个指标中，正样本的交易总价值均高于负样本，这较为符合钓鱼节点骗取大量钱财的性质。

在 FT7（邻居节点数）、FT8（平均交易频率的倒数）这两个指标中，钓鱼节点平均的 FT7 值高于非钓鱼节点，FT8 值小于非钓鱼节点。网络钓鱼的过程往往是广泛撒网，增加诈骗成功的数量，并且在诈骗后转移赃款，钓鱼节点将会比正常节点有更多的节点，转账的频率更高。

4.1.4 划分时间间隔

论文主要探讨的是离散的动态图嵌入，选取动态图两次图快照的时间间隔为 30 天，即每 30 天获取一次图快照，每个快照包括小于当前划分时间的所有节点和链接的信息。由此可以获得 S1、S2 和 S3 的快照数量分别为 42、43 和 43。

4.2 实验设置

4.2.1 评估指标

为了检测模型嵌入的效果，完成对钓鱼节点的检测任务，实验将对交易网络的节点进行节点表示学习，然后利用嵌入向量进行节点二分类任务。实验的评估指标选取分类的评估指标，包括精确率、召回率、F1 值和 AUC 值等。对于二分类模型，预测情况和实际情况可得出四种组合，混淆矩阵表 4-3 所示：

表 4-3 混淆矩阵定义

	预测正例	预测反例
实际正例	True Positive (TP)	False Negative (FN)
实际反例	False Positive (FP)	True Positive (TN)

由混淆矩阵可计算出：

1. 精确率（Precision，简称 P）表示预测正确的样本数占预测正样本总数的比重，计算公式为：

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

2. 召回率（Recall，简称 R）表示预测正确的样本数占实际正样本总数的比重，计算公式为：

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

3. F1 值（F1-measure，简称 F1）是模型的召回率和精确率的调和平均值，计算公式如下：

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{4.3}$$

4. AUC（Area Under Curve）是 ROC 曲线下的面积。ROC 曲线（Receiver Operating Characteristic Curve）指受试者工作特征曲线，横轴为假阳性概率（False Positive Rate，简称 FPR），纵轴为真阳性概率（True Positive Rate，简称 TPR）, 具体计算公式如下：

$$FPR = \frac{FP}{FP + TN} \tag{4.4}$$

$$TPR = \frac{TP}{TP + FN} \tag{4.5}$$

当 ROC 曲线整体越偏向左上角时，ROC 曲线下面积越大，即 AUC 值越大时，模型预测的准确率越高，分类器的性能越好。

在网络钓鱼诈骗节点检测中，相比将正常节点错判成钓鱼节点，向用户报告可疑节点的假阳性情况（FP），将真正的钓鱼节点判断为正常节点的情况（FN）造成的结果更为严重。因此，在所有的指标中，召回率更值得关注。

4.2.2 对比方法

为了证明模型的有效性，需要将本论文模型 DynGCREM 与现有的模型进行比较。结合相关方法出现的顺序以及对论文具体场景的实用性，选取了三种主要使用神经网络的针对离散动态图嵌入的方法作为基准，与论文提出的模型进行多角度的对比。对比的方法具体介绍如下：

1. DynGEM：基于 SDNE 模型的架构，通过深层自编码器，通过上一个时刻的图快照模型参数初始化下一时刻图快照的模型，加快了收敛的迭代速度，训练时长得以减小。

2. dyngraph2vec：在 DynGEM 的基础上考虑多个时间步长，在训练当前图快照的模型时，结合之前的多个时间步长来学习模型参数。该方法整体的结构还是使用自编码器重构邻接矩阵来获取图嵌入表示。

3. EvolveGCN：使用循环神经网络来动态更新 GCN 的参数，在演化的网络参数中捕获动态，能够处理更灵活的动态网络数据。

4. Feature：除了以上三种动态图嵌入方法，对节点不进行图嵌入表示，仅使用 4.1.3 节计算出的特征完成分类任务。

4.2.3 模型参数设置

本论文模型使用 Python 编写，图卷积网络层选择 Pytorch 框架进行实现，循环神经网络层选择 Tensorflow 框架进行实现，分类任务采用 LightGBM 框架实现梯度提升，提高训练的速度和模型的精度。模型内部涉及的一些权重参数、特征参数均使用随机初始化的方法，选择 Adam 优化器，便于学习率的调参。在对比实验中，动态图嵌入方法使用 CTGCN 框架中的方法，与图卷积网络 GCN 的对比模型中使用 openNE 框架进行对比实验。所有的实验均在不同的随机生成器的种子下重复实验 5 次求平均值。

下面列出超参数和激活函数的选取：

1. 图卷积网络层

- 卷积网络层数：1
- 学习率：0.001
- epoch：6
- 激活函数：使用 *tanh* 函数
- 学习率：从[0.001,0.005,0.01,0.015,0.02,0.025,0.03,0.035,0.04,0.045,0.05]择优选取

2. 循环神经网络层

- 批处理大小：128
- 编码器（解码器）层数：2
- 隐藏层大小：32
- 嵌入层大小：8
- 学习率：从[0.01,0.001,0.0001,0.00001]中择优选取
- 编码器部分选取的激活函数：输入层使用 *tanh* 函数，隐藏层使用 *ReLU* 函数
- 解码器部分选取的激活函数：输入层使用 *tanh* 函数，隐藏层使用 *ReLU* 函数

3. GDBT 分类器

在以太坊交易网络中对异常节点分类是获取节点图嵌入的下游任务之一。在获取节点的图嵌入后，使用 LightGBM 框架进行分类，分类参数保持相同，采用 5 折交叉验证来验证二分类问题的结果。

4.3 模型评估

对本模型和对比方法进行实验，子图 S1、S2、S3 的结果分别如表 4-4、表 4-5 和表 4-6 所示。结果显示，三个动态图嵌入的模型中，DynGEM 在 S2 和 S3 的表现均优于另外两种方法，evolveGCN 仅在 S1 的表现要优于另外两种方法，但值得一提的是 evolveGCN 在 S1 和 S2 的召回率都优于另外两种方法。利用三种动态图嵌入方法表示节点再进行分类，相较于仅使用特征进行分类，各项指标中均有较大的提升。

本论文模型 DynGCREM 与这三个模型相比，在 S1 中各项指标均高于对比方法，但在 S2 中，模型的精确率低于使用 DynGEM 的方法，S3 中 AUC 等指标也低于 DynGEM 方法。一个原因是 DynGEM 模型使用了回溯超参数（look back）控制模型学习动态图时序信息的长度，因此能更好地利用模型的时序信息。尽管部分指标略低于 DynGEM，但本论文模型与 evolveGCN 和 dynamic2graph 相比仍然表现出了较好的性能，在召回率和精确率指标中表现均较为良好，较高地提升了使用特征进行分类的结果。

整体而言，所有方法对交易网络节点的二分类任务的 AUC 值和精确率都并不高，主要的原因仍在于节点本身特征的缺乏，仅利用了节点和链接的信息。但仍然能从结果看出利用图嵌入技术对图节点进行表示学习，能够提升图节点分类的性能，更有效地检测钓鱼节点。结果证明了论文提出的模型 DynGCREM 在动态图嵌入中的有效性。

表 4-4 S1 子图实验结果

评价指标	S1				
	DynGCREM	DynGEM	evolveGCN	dynamic2graph	Feature
AUC	**0.5669**	0.5540	0.5620	0.5600	0.5450
recall	**0.1340**	0.1083	0.1245	0.1203	0.0901
precision	**0.6905**	0.5745	0.4833	0.6072	0.6401
F1	**0.2165**	0.1764	0.1963	0.1907	0.1481

表 4-5 S2 子图实验结果

评价指标	S2				
	DynGCREM	DynGEM	evolveGCN	dynamic2graph	Feature
AUC	**0.5755**	0.5530	0.5557	0.5537	0.5337
recall	**0.1514**	0.1060	0.1118	0.1076	0.0675
precision	0.6325	**0.6928**	0.5331	0.6092	0.4833
F1	**0.2365**	0.1799	0.1743	0.1762	0.1183

表 4-6 S3 子图实验结果

评价指标	S3				
	DynGCREM	DynGEM	evolveGCN	dynamic2graph	Feature
AUC	0.5775	**0.5801**	0.5574	0.5686	0.5548
recall	0.1556	**0.1608**	0.1152	0.1376	0.1101
precision	0.5245	0.5010	0.4782	**0.5376**	0.4700
F1	0.2350	**0.2377**	0.1820	0.2153	0.1750

4.4 模型研究

4.4.1 消融实验

为了探究图卷积网络层和循环神经网络层分别在图嵌入中发挥的作用，需要将包含本论文提出的包含两个模块的模型和仅包含一个模块的模型进行对比，具体地说，需要将以下的模型进行对比：

1. DynGCREM：论文提出的基于图卷积循环的动态图嵌入模型，包括图卷积网络层和循环神经网络层两个模块。

2. M1：只含有图卷积网络层的模型，注意该模型相当于对静态图进行节点嵌入，因此最终的结果是使用图快照序列中最后一个图快照G_T进行实验。M1 通过图卷积网络层获

得输出，并将输出的节点嵌入和原始特征合并，作为分类器的输入，由此获得分类结果。

3. M2：只含有循环神经网络层的模型，与 DynGCREM 模型不同的是，M2 的循环神经网络层输入并不包括节点嵌入特征，使用的是 4.1.3 节提取的特征。将每一个图快照的节点的原始特征通过 LSTM 自编码网络后得到最终的节点嵌入，将该嵌入作为分类器的输入，最终得到结果。

4. Feature：无节点嵌入步骤，仅用 4.1.3 节提取的特征进行计算的模型。

表 4-7、表 4-8 和表 4-9 分别展示了三个子图的消融实验结果。

表 4-7 S1 消融实验结果

评价指标	S1			
	DynGCREM	M1	M2	Feature
AUC	**0.5669**	0.5655	0.5539	0.5450
recall	**0.1340**	0.1314	0.1081	0.0901
precision	**0.6905**	0.5854	0.5843	0.6401
F1	**0.2165**	0.2073	0.1760	0.1481

表 4-8 S2 消融实验结果

评价指标	S2			
	DynGCREM	M1	M2	Feature
AUC	**0.5755**	0.5716	0.5411	0.5337
recall	**0.1514**	0.1438	0.0826	0.0675
precision	**0.6325**	0.4919	0.3405	0.4833
F1	**0.2365**	0.2167	0.1321	0.1183

表 4-9 S3 消融实验结果

评价指标	S3			
	DynGCREM	M1	M2	Feature
AUC	**0.5775**	0.5647	0.5749	0.5548
recall	**0.1556**	0.1301	0.1504	0.1101
precision	**0.5245**	0.4315	0.4452	0.4700
F1	**0.2350**	0.1952	0.2223	0.1750

通过消融实验，能够验证模型中的两个组件的确能够提高图嵌入的性能。

将 M1（仅包含图卷积网络层）的实验结果与 DynGCREM 对比，能发现 M1 的 AUC 和 DynGCREM 模型较为接近，说明图卷积网络层较为显著地提高了分类的 AUC 值，而本论文模型相比单纯使用 GCN 的 M1 的精确率更高，考虑是引入了循环神经网络自编码器带来的效果，但 M2（仅包含循环神经网络层）的指标中精确率并没有明显的提高，可

能是由于 M2 的 LSTM 自编码器重构的仅是原始特征，并不是已经经过图卷积网络嵌入表示的节点向量，因此对结果造成了影响。

M2 的召回率与 Feature（仅使用特征分类）的召回率相比均有较明显的提升，如在 S3 中，M2 的召回率已经和本论文模型 DynGCREM 相近。推测 LSTM 自编码器的引入也提升了模型的召回率。总体而言，实验说明论文提出的模型能够较有效地结合节点的时间信息和空间信息，实现更好的分类效果。

4.4.2 空间信息利用

为了探究图卷积网络对空间拓扑结构信息的利用，本节实验探究图卷积网络与其他基于随机游走的静态图嵌入方法相比，是否更有效地利用了空间拓扑结构信息。因此，实验利用图快照序列的最后一个快照G_T进行静态图的节点嵌入表示学习，并将节点嵌入向量作为节点分类的输入。选取了以下静态图中近期较为经典的基于随机游走的模型作为基准，与论文模型中的图卷积网络层进行对比，具体如下：

1. M1：使用 GCN 结合自动编码器学习节点表示向量。

2. DeepWalk：选择特定节点作为起始节点，使用随机游走来获得节点的有序序列，使用 Word2Vec 学习节点表示向量。

3. Node2Vec：引入两个参数来平衡广度优先搜索和深度优先搜索改善随机游走，对 DeepWalk 方法进行了改进。

4. LINE：引入了节点之间的一阶相似度和二阶接近度，算法的优化目标是保证原空间和嵌入空间的概率分布保持一致，让嵌入表示更好地保留图的一阶相似性。

5. Feature：使用 4.1.3 小节提取的节点的特征信息作为节点分类的输入。

对比的具体结果如表 4-10、表 4-11 和表 4-12 所示。

从表中能看出，在 DeepWalk、Node2Vec 和 LINE 三种方法中，LINE 在子图 S1 和 S3 的效果最好，AUC、Recall 均超过另外两种方法，Precision 更是超过了对比的 GCN 方法，是所有方法中效果最好的，可能的原因在于 LINE 利用了二阶相似性，能够更好地利用边的权重，感知临近的信息。但在 S2 子图中，DeepWalk 的效果要优于另外两种方法，但在三种方法中使得 S2 的 Precision 最高的是 Node2Vec。总的来说，相比单纯使用节点特征进行分类，这几种静态节点嵌入方法均显示出了它们的有效性，在分类任务的性能上都有所提高。

DynGCREM 模型利用图卷积神经网络重构邻接矩阵，无监督地学习节点嵌入表示，意在更好地捕获图的空间拓扑信息，从表中能够看出相比使用原始的特征作为图节点的表示，GCN 比其他三种方法更有效地提高了分类性能。图卷积网络层中利用 GCN，在 AUC、recall 和 F1 值中均优于对比的方法。但是在 Precision 上，GCN 的表现并不稳定。GCN 通过考虑将节点特征和空间结构进行较好的结合，结果也可表明将利用 GCN 和自编

码器的结构相结合无监督式地学习节点嵌入表示的确具有有效性。

表 4-10 S1 的静态嵌入方法对比结果

评价指标	S1				
	M1	DeepWalk	Node2vec	LINE	Feature
AUC	**0.5670**	0.5445	0.5389	0.5525	0.5458
recall	**0.1345**	0.0893	0.0779	0.1051	0.0918
precision	0.5886	0.5263	0.5762	**0.8762**	0.6034
F1	**0.2097**	0.1490	0.1340	0.1784	0.1507

表 4-11 S2 的静态嵌入方法对比结果

评价指标	S2				
	M1	DeepWalk	Node2vec	LINE	Feature
AUC	**0.5753**	0.5720	0.5415	0.5406	0.5505
recall	**0.1511**	0.1445	0.0831	0.0815	0.1014
precision	0.4648	0.5123	**0.5565**	0.4567	0.5050
F1	**0.2247**	0.2190	0.1402	0.1344	0.1613

表 4-12 S3 的静态嵌入方法对比结果

评价指标	S3				
	M1	DeepWalk	Node2vec	LINE	Feature
AUC	**0.5736**	0.5608	0.5628	0.5638	0.5528
recall	**0.1477**	0.1221	0.1258	0.1279	0.1060
precision	0.4560	0.4658	**0.6998**	0.6653	0.4623
F1	**0.2196**	0.1887	0.2064	0.2043	0.1689

4.4.3 图卷积网络层参数设置

本节实验探讨图卷积层网络中的参数对性能的影响。在图卷积神经网络层中，影响结果的主要参数包括卷积层数、嵌入层大小和学习速率。

为了更好地比较图卷积网络层模块的参数，使用消融实验的 M1（即只包含图卷积网络层的模型）进行分类的数据进行对比，这样能够排除循环神经网络层对结果的干扰，更好地探究图卷积层的超参数在空间信息利用上对结果的影响。

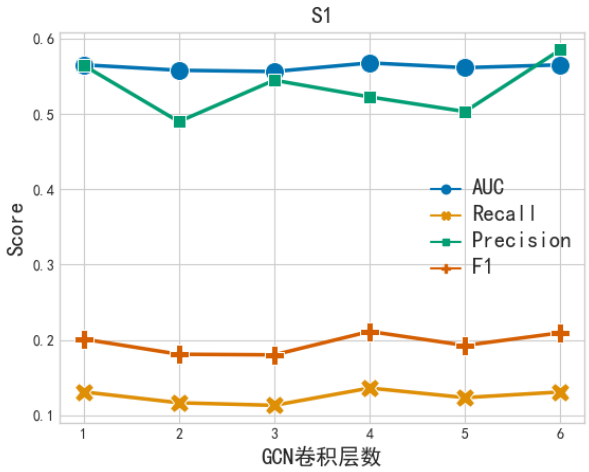
- 卷积层数

设置图卷积网络层的卷积层数分别是 1 至 6 层，模型的结果如图 4-1 所示。

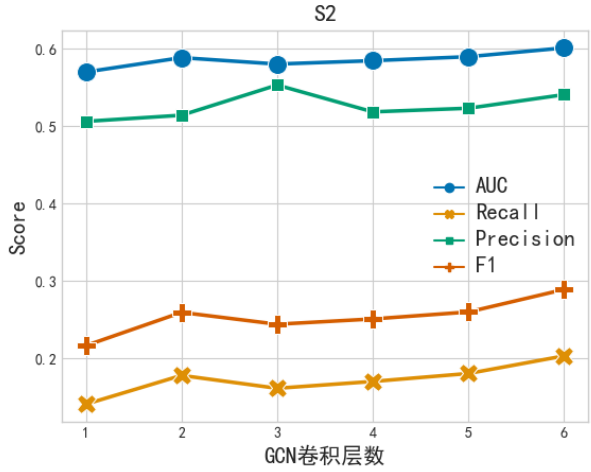
从结果能够看出，当 GCN 的深度增加时，模型的性能会不断提高，但是随着卷积层数量的不断增加，整体性能也会略有下降。如 S1、S2 在图卷积网络层数为 2 时、S3 在图

卷积网络层数为 3 时，都出现了一定的下降。除了 S2 在最后呈现上升趋势，S1 和 S3 在最后均为略有下降的趋势。另外从图中可看出，随着图卷积层的不断增加，AUC 的变化幅度较为平缓，但 Precision 的变化幅度较大。

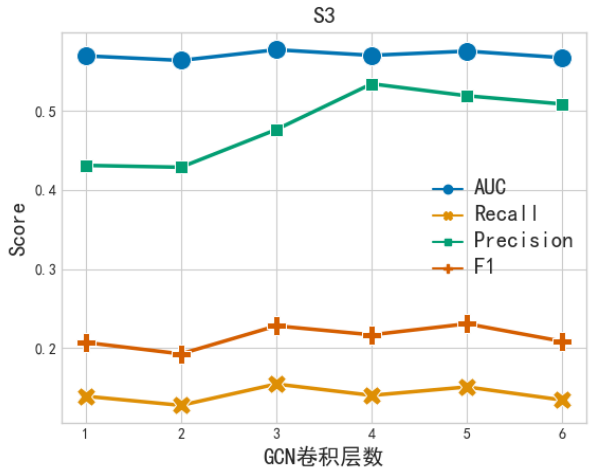
目前的 GCN 模型不断地叠加多层结构，容易出现过拟合的问题。虽然层数越多，更多的高阶邻居信息会更多，但实际上高阶的邻居信息可能会因为小世界原理[35]失去它们本身的特性，从而使得模型的性能变得更差。因此为了使模型的结果更为稳定，避免上述的过拟合等问题，模型在图卷积层数的选择上，尽可能选取低阶且表现好的层数。



a. S1 结果



b. S2 结果



c. S3 结果

图 4-1 不同卷积层数对图卷积层输出的影响

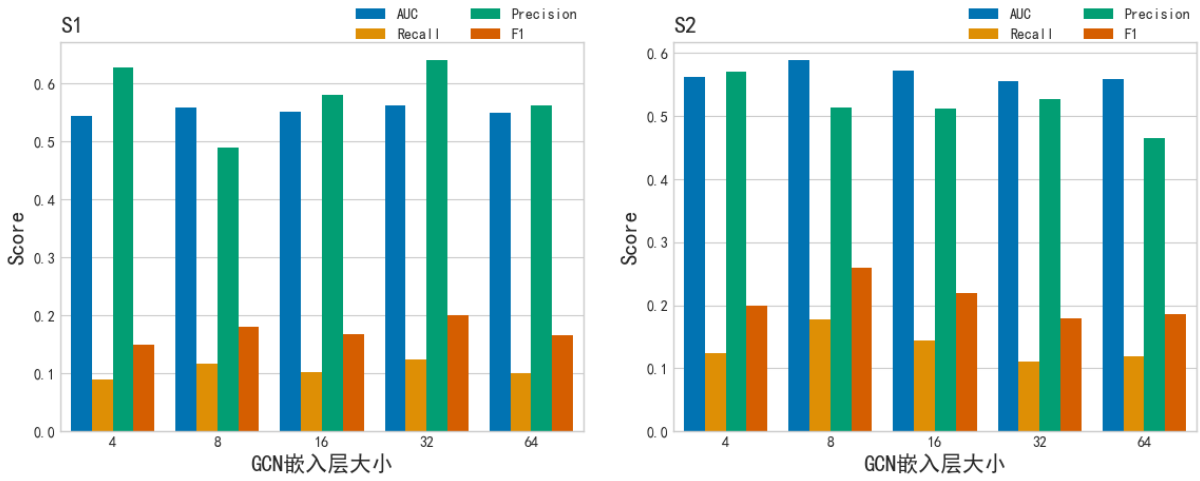
- 嵌入层大小

由于图卷积网络层结合了自编码的结构，因此图卷积网络层输出的嵌入层向量大小也会影响最终的分类效果。在图卷积网络层中，输入是一个 8 维的特征向量，因此当输出的嵌入向量小于 8 维时，嵌入过程是一个在尽可能保留原始信息的降维操作，当输出嵌入

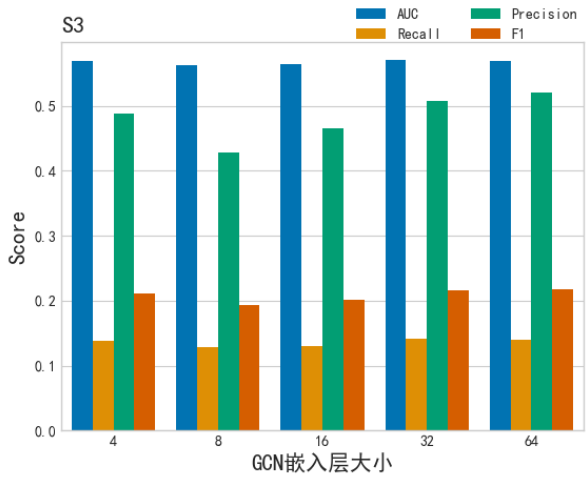
向量大于 8 维时，嵌入过程获取了一个稀疏的嵌入表示。

设置图卷积层的嵌入层大小分别为[4,8,16,32,64]，模型的结果如图 4-2 所示。

当嵌入层小于 8 时，GCN 模型在三个子图的 Precision 较高，但在 S1 和 S2 中 Recall 值较低。当嵌入层为 8 时，节点在 S1 和 S2 的 AUC 值、Recall 值与 F1 值都较高。但当增加嵌入层的维度，如 S1 在嵌入层维度为 32 时，Precision 达到峰值，S2 也在 32 时的 Precision 达到一个较小的峰值，但当嵌入层的维度更高时，性能开始缓慢地下降，且计算时间会不断增加。由于在钓鱼节点的检测中，更关注召回率 Recall 值，并且为了平衡时间与性能，模型选取效果和时间都比较良好的嵌入层大小为 8 的模型。



a. S1 结果　　b. S2 结果



c. S3 结果

图 4-2 图卷积网络层中不同嵌入层数对结果的影响

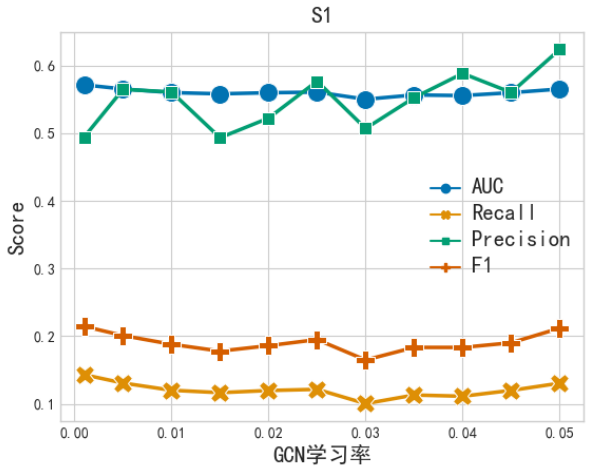
- 学习速率

图卷积网络的学习速率也是一个较为重要的超参数，将图卷积神经网络 GCN 的学习率设置为[0.001,0.005,0.01,0.015,0.02,0.025,0.03,0.035,0.04,0.045,0.05]分别进行训练，S1、

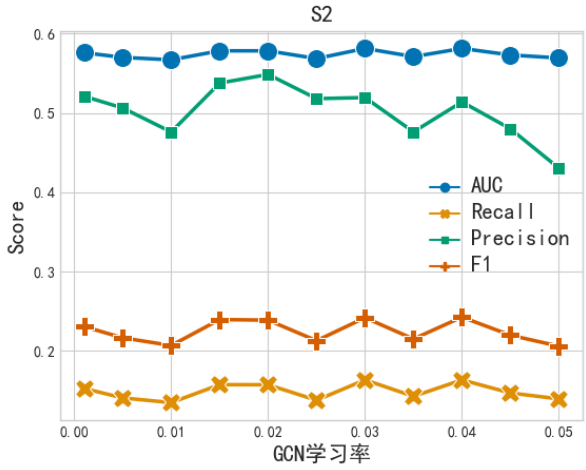
S2 和 S3 结果如图 4-3 所示。

结合结果能够看出，多数情况下，学习率在 0.005 至 0.015 这一区间时，模型表现较好。在 0.02 至 0.025 左右，模型的性能有一定的下降趋势，三个子图的 F1 值和 Recall 值均有所下降。

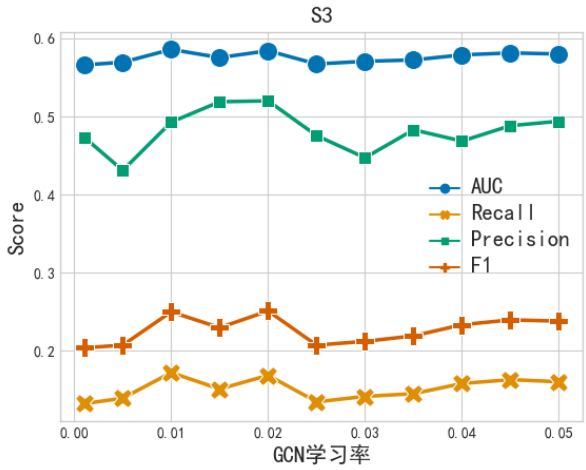
虽然学习率较小时模型能够稳定并收敛，但在训练时使用较小的学习率会导致训练速度缓慢，而且可能陷入局部最优点，无法达到最好的效果。



a. S1 结果



b. S2 结果



c. S3 结果

图 4-3 图卷积网络层中不同学习速率对结果的影响

4.4.4 循环神经网络层参数设置

在 LSTM 自编码器模块中，主要影响模型分类效果的因素包括 LSTM 自编码器的层数、LSTM 输出的嵌入层层数，下面对这些影响因素进行探究。以下对循环神经网络层参数设置的探究均使用 S2 子图进行实验。

- 层数

在设置LSTM自编码器的层数时，编码器的层数神经元个数需要不断减少，解码器的神经元个数需要不断增加。在编码层仅有一层LSTM网络时，该层网络的神经元个数被设置为 8（即嵌入层大小）。在两层 LSTM 网络中，LSTM 层编码器每层的大小被设置为[32,8]，三层和四层的LSTM编码器分别被设置成[32,16,8]和[48,32,16,8]。解码器每层的大小与编码器对称。

表 4-13 LSTM 自编码器不同层数实验结果

评价指标	1	2	3	4
AUC	0.5627	**0.5812**	0.5739	0.5711
recall	0.1257	**0.1628**	0.1482	0.1427
precision	**0.6397**	0.5844	0.5502	0.4959
F1	0.2054	**0.2513**	0.2277	0.2130

实验的结果如表 4-13 所示。在 LSTM 自编码的层数为 2 时，分类效果最好。当 LSTM 自编码器的层数加深时，模型分类效果有所下降，这可能是由于模型过拟合造成的。另外当层数加深时，训练时间的花销也会增加，因此模型设计中选取 2 层的 LSTM 自编码器较为合适。

- 嵌入层大小

设置循环神经网络输出的嵌入层数分别是[4,8,16,32,64]进行实验，模型的结果如图 4-4 所示。当嵌入层大小为 8 时，精确率和 AUC 值都达到了峰值。当嵌入层增大时，模型的性能有所下降。

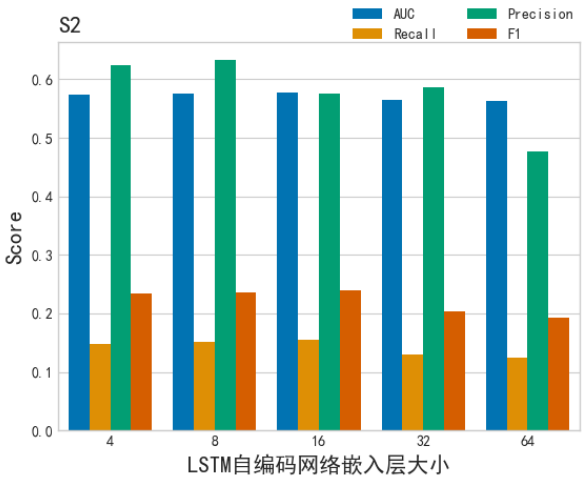


图 4-4 LSTM 自编码器不同嵌入层大小实验结果

4.4.5 图快照划分

在LSTM自编码网络处理时序信息时，不同的动态图快照的时间间隔划分会对训练结

果具有影响，设计不同时间步长下的实验有助于理解动态图中的时间划分对结果的影响。本节设计实验对 S2 进行不同时间段的划分，分别选取 15 天、30 天、45 天、60 天进行划分，其余参数与 4.2.3 节相同。训练的结果如图 4-5 所示。

当时间间隔为 30 天时，模型的效果在各项指标中都最好。当时间间隔为 45 天和 60 天时，模型的 AUC 值略有降低，但精确率、召回率均大幅度降低。模型在时间间隔划分过大的原因是：当时间间隔划分得越大时，动态图序列中的图快照数量越少，时序信息被丢失得越多，更少的信息将会使动态图中节点的嵌入表示难以准确地获取特征中的信息，而且 LSTM 自编码网络可能会错误地学习了网络的时序信息，使得分类性能更差。

当时间间隔为 15 天时，模型的 Precision 值低于时间间隔为 30 天的，但其他指标均和时间间隔为 30 天的模型较为接近。但当时间间隔为 15 天时，动态图快照的数量也比 30 天的增加了 1 倍，这就使模型训练的时间较长。因此，如何平衡时间步长的划分导致的性能和模型训练时长的变化，可以作为一个未来拓展的工作。

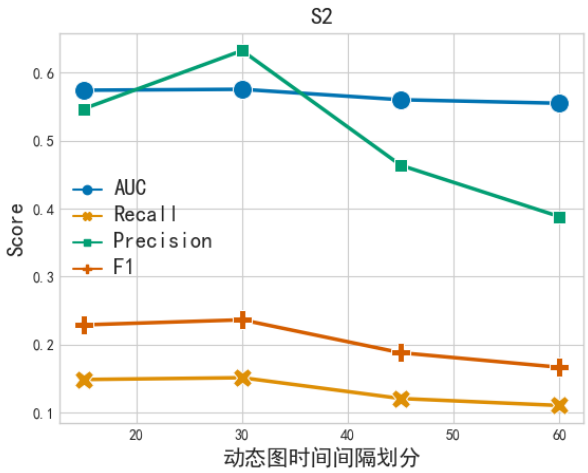


图 4-5 不同时间步长的实验结果

4.6 小结

本章首先介绍了以太坊交易网络数据的具体信息以及处理方法，包括对数据集的采样、特征提取和动态图快照的划分；其次介绍了实验评估指标、具体设置，完成了 DynGCREM 模型和对比方法的对比实验，并对结果进行分析。最后通过多角度的实验对模型进行研究，主要进行了消融实验、图卷积网络和其他静态图嵌入方法的对比实验、不同的图卷积网络层和循环神经网络层的参数设置对比实验、不同时间间隔的图快照划分实验，更深入地探究模型的各项参数对模型性能的影响。

5 总结与展望

本章对论文主要工作、创新点和研究结论进行简单的总结，并指出论文存在的不足之处，在此基础上探讨未来可能的改进方案。

5.1 工作总结

网络钓鱼是区块链数字货币交易平台中的一种常见诈骗行为，它会窃取用户财产，从而威胁数字加密货币交易平台的安全。如何对网络钓鱼诈骗节点进行检测，对维护区块链生态安全有着较大的意义。由于区块链的匿名性，数字货币交易平台难以收集到具有交易行为账户的特征，很难直接通过账户特征对节点进行检测。除此之外，交易网络存在的节点数据量大、正负样本不均衡等问题也导致对钓鱼节点的检测难度较大。

在数字加密货币的交易网络中，每笔交易都包括交易双方、金额和时间信息，账户和交易都随时间动态增加，因此交易网络可以被看作一个动态图结构。为了实现对钓鱼账户的检测，需要将所有的账户分为异常的钓鱼账户和正常账户。对图中的节点进行表示学习是图节点分类前的关键。

现在的图嵌入的相关工作中，目前的静态网络图嵌入模型缺乏对时序信息的利用，而动态图嵌入模型通常在学习节点嵌入表示时仅考虑节点之间的一阶相似性，并未考虑更高阶的相似性，对图的空间拓扑信息利用不足。针对以上的挑战和相关工作的不足，论文重点研究了如何结合图数据中的高阶邻居信息和时序信息来完成动态图的嵌入表征，提出了一种无监督学习的基于图卷积循环的动态图节点嵌入的方法 DynGCREM，能够同时捕捉图的结构信息和时序信息。在图卷积神经网络层中，利用图卷积网络提取潜在的深层结构特征表示；在循环神经网络层，使用长短期记忆网络和自动编码的框架相结合，学习时序信息，最终学习图节的用向量表示，并用于下游任务。这一方法能够同时捕捉图的结构信息和时空信息，一定程度上解决了动态图的节点表示学习问题。

本模型在以太坊的交易网络数据集进行了广泛的实验。首先将模型和动态图嵌入的方法进行对比试验，验证了模型的有效性，证明了图卷积网络和循环神经网络相结合的确可以提升图表示学习的效果。其次对模型的各模块进行消融实验，结果显示图卷积网络层较为显著地影响了动态图嵌入的结果。接着通过实验证明图卷积网络相较于基于随机游走的图嵌入算法能够更好地利用空间信息。最后还分别探究了图卷积网络层和循环神经网络层的参数设置，更深入地探究模型的各项参数对性能的影响。

5.2 未来展望

论文提出的结合图卷积神经网络和循环神经网络的动态图嵌入的模型 DynGCREM 总

体来说达到了预期，但仍然存在着一些不足，主要有以下几点：

1. 模型训练时长较慢，难以适应不断增加节点和链接的动态图。DynGCREM 模型在训练和验证时使用的一段时间进行划分的动态图快照，但在实际场景中，当动态图有更多的节点和边增加时，LSTM 自编码网络需要重新进行训练，将花费较多的时间。

2. 模型无法动态确定图快照的时间间隔。对于离散时间的图快照，并不是每一个时刻的图快照都对结果产生相同的影响：图快照的变化在一段较长的时间内可以被忽略不计，或在某一段较短的时间内，图快照包含了钓鱼节点的特征，对结果的影响较大，却因为时间间隔选取较大而被忽略。本模型仅采用相同等长的时间间隔来划分图快照，并没有考虑如何利用非等长的时间间隔缩小训练的时长和复杂度，更好地捕捉不同时间间隔的图快照的节点特征。

针对上述的不足，未来的改进方向是通过动态地划分图快照的时间间隔，从而减小模型训练的次数，提高模型的效率，让模型更加适用实际场景。

参考文献

[1] Szabo N. Formalizing and Securing Relationships on Public Networks[J]. First Monday, 1997, 2(9) :https://doi.org/10.5210/fm.v2i9.548

[2] Bourgi S.Ethereum’s market cap exceeds that of platinum for the first time[EB/OL].[2021-04-28]. https://cointelegraph.com/news/ethereum-s-market-cap-exceeds-that-of-platinum-for-the-first-time

[3] Chainalysis Team.The Rise of Cybercrime on Ethereum[EB/OL].[2017-08-07]. https://blog.chainalysis.com/the-rise-of-cybercrime-on-ethereum/.

[4] Chainalysis Team. Crypto Crime Series: Decoding Ethereum Scams [EB/OL].[2019-01-23]. https://blog.chainalysis.com/reports/ethereum-scams.

[5] Paul Deniero**.**100s of ETH Stolen After Bee Token ICO Email List Hacked [EB/OL].[2018-02-01]. https://theripplecryptocurrency.com/bee-token-scam/

[6] Li Y, Yang Z, Chen X, et al. A stacking model using URL and HTML features for phishing webpage detection[J]. Future Generation Computer Systems, 2019, 94:27-39.

[7] Sahingoz O K, Buber E, Demir O, et al. Machine learning based phishing detection from URLs[J]. Expert Systems with Applications, 2019, 117:345–357.

[8] Mao J, Bian J, Tian W, et al. Phishing page detection via learning classifiers from page layout feature[J]. EURASIP Journal on Wireless Communications and Networking, 2019, 2019(1):1–14.

[9] Jindal N, Liu B, Lim E-P. Finding unusual review patterns using unexpected rules[C]//Proceedings of the 19th ACM international conference on Information and knowledge management, 2010: 1549–1552.

[10] Fawcett T, Provost F. Activity monitoring: Noticing interesting changes in behavior[C]//Proceedings of the 5th ACM SIGKDD international conference on Knowledge discovery and data mining, 1999: 53–62.

[11] Phua C W C, Alahakoon D, Lee V C-S. Minority Report in Fraud Detection: Classification of Skewed Data[J]. SIGKDD Explorations, 2004, 6(1):50–59..

[12] Wu J, Yuan Q, Lin D, et al. Who are the phishers? phishing scam detection on ethereum via network embedding[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2020,2020(2):1-11.

[13] Akoglu L, McGlohon M, Faloutsos C. Oddball: Spotting anomalies in weighted graphs[C]//Springer. Pacific-Asia conference on knowledge discovery and data mining, 2010: 410–421.

[14] Chouiekh A, Haj, EL Hassane Ibn EL. Convnets for fraud detection analysis[J]. Procedia Computer Science, 2018, 127:133–138.

[15] 陈丽, 朱裴松, 钱铁云等. 基于边采样的网络表示学习模型[J]. 软件学报, 2018, 29(3):756–771.

[16] Malhotra P, Vig L, Shroff G, et al. Long short term memory networks for anomaly detection in time series[C] //Presses universitaires de Louvain. Proceedings, 2015: 89–94.

[17] Rossi R A, Gallagher B, Neville J, et al. Modeling dynamic behavior in large evolving graphs[C] //Proceedings of the sixth ACM international conference on Web search and data mining, 2013: 667–676.

[18] Gupta M, Gao J, Sun Y, et al. Integrating community matching and outlier detection for mining evolutionary community outliers[C]//Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, 2012: 859–867.

[19] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations[C]//Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, 2014: 701-710.

[20] Tang J, Qu M, Wang M, et al. Line: Large-scale information network embedding[C]//Proceedings of the 24th International Conference on World Wide Web, 2015: 1067–1077.

[21] Grover A, Leskovec J. node2vec: Scalable feature learning for networks[C]// Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining, 2016: 855-864.

[22] Wang D, Cui P, Zhu W. Structural deep network embedding[C]//Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining, 2016: 1225-1234.

[23] Bruna J, Zaremba W, Szlam A, et al. Spectral networks and locally connected networks on graphs[C]//1st International Conference on Learning Representations,2013.

[24] Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering[C]//Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016: 3844–3852.

[25] Kipf T, Welling M. Semi-Supervised Classification with Graph Convolutional Networks[C]//5th International Conference on Learning Representations, 2017.

[26] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[C]//Advances in Neural Information Processing Systems, 2017: 1024–1034.

[27] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[C]//6th International Conference on Learning Representations, 2018.

[28] Liben-Nowell D, Kleinberg J. The link-prediction problem for social networks[J]. Journal of the American society for information science and technology, 2007, 58(7):1019–1031.

[29] Liu X, Hsieh P-C, Duffield N, et al. Real-time streaming graph embedding through local actions[C]//Companion Proceedings of The 2019 World Wide Web Conference, 2019: 285–293.

[30] Sajjad H P, Docherty A, Tyshetskiy Y . Efficient Representation Learning Using Random Walks for Dynamic Graphs[J]. CoRR, 2019:abs/1901.01346.

[31] Yu W, Cheng W, Aggarwal C C, et al. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018: 2672–2681.

[32] Zhou L, Yang Y, Ren X, et al. Dynamic network embedding by modeling triadic closure process[C]//Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[33] Goyal P, Kamra N, He X, et al. DynGEM: Deep Embedding Method for Dynamic Graphs[J]. CoRR, 2018:abs/1805.11273.

[34] Goyal P, Chhetri S R, Canedo A. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning[J]. Knowledge-Based Systems, 2020, 187:104816.

[35] Pareja A, Domeniconi G, Chen J, et al. Evolvegcn: Evolving graph convolutional networks for dynamic graphs[C]//Proceedings of the AAAI Conference on Artificial Intelligence, 2020: 5363–5370.

[36] Watts D J, Strogatz S H. Collective dynamics of ‘small-world’networks[J]. nature, 1998, 393(6684):440–442.

声　明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

学位论文作者（签名）_____

论文指导教师（签名）_____

2021 年 5 月 28 日

致　　谢

在毕业设计之初，我觉得这个题目对我来说是一个挑战，接近两个学期的学习和实践后，现在终于也写到了最后的致谢部分，即将结束此次毕业设计。在完成毕业设计的过程中，最想感谢的就是指导我的段磊老师和陈亮老师。从选题到完成，他们对我毕业论文提出的建议都让我深受启发，感谢他们细致严谨、认真负责的指导。其次想要感谢实验室的各位师兄师姐，他们耐心地解答我的问题、真诚地分享他们在科研上的经历。在导师和实验室的各位师兄师姐们的帮助中，我逐渐对图机器学习这一领域有了整体的认识，也学习了一些深度学习的方法，开始形成自己的想法和思路。毕业设计对我来说是一次完整的科研训练，感谢这段时间帮助了我的所有人，感谢实验室提供的便利资源。

写到致谢部分，也就意味着大学四年的生活已经接近尾声。回顾四年来，与很多人相遇、相识，受到了他们的指导、帮助、鼓励和关心，在此想对他们道一声感谢。

感谢川大给予了我丰富的大学生活，让我能在图书馆里自由地阅读与学习，能在丰富的课堂上领略知识的魅力，能在绕湖的塑胶跑道上奔跑与运动。感谢川大用它广阔的胸怀改变和塑造了我，感谢它抹去了我的懵懂和迷茫，让我有了继续前行的勇气和自信。

感谢朱敏老师和视觉计算实验室的各位师兄师姐，在我还是一个对科研一无所知的新生时，他们给我上了第一堂关于“科研”的课，培养了我的科研兴趣和素养。感谢每一位任课老师，他们传授给了我知识，指引着我，让我学会学习和思考，不断进步。

感谢吴玉章学院的各位优秀的同学，虽然专业各不相同，但每次和他们交谈与讨论，我都能收获新的思想，认识到自己的局限性。感谢我的朋友们和室友们，他们无论在学习和生活上都给了我很多帮助和鼓励。与这些同窗好友相识，是我大学里最大的幸运。

感谢所有家人们，他们尽管无法理解和解决我在专业学习上遇到的问题，可他们真正地关心着我的身体是否健康，关心我是否快乐。感谢他们的支持，让我能够在求知的道路上专心致志、追求卓越。

寥寥数语，道不尽我的感激之情，愿能将所有的力量汇聚，再次上路，成为更好的自己。

附录 （一）论文翻译

翻译译文

AddGraph：基于注意力时态图卷积网络的动态图异常检测

Li Zheng, Zhenpeng Li , Jian Li , Zhao Li and Jun Gao
The Key Laboratory of High Confidence Software Technologies, Ministry of Education, China
School of EECS, Peking University, China
Alibaba Group, China
{greezheng, gaojun}@pku.edu.cn, {zhen.lzp,zeshan.lj,lizhao.lz}@alibaba-inc.com

摘要

动态图中的异常检测在许多不同的应用场景中非常重要，如推荐系统，但是异常通常是高度灵活的，且标记为异常的数据不足，它也带来了巨大的挑战。最好是通过考虑所有可能的提示来学习异常模式，包括结构、内容和时间特征，而不是对部分特征使用启发式规则。在本文中，我们提出了 AddGraph，一个通用的端到端的异常边检测框架，它使用一个扩展的时间图卷积网络和一个注意力模型，可以捕获动态图中的长期模式和短期模式。为了解决显式标记数据不足的问题，我们在半监督式的 AddGraph 训练中采用了选择性负采样和边丢失的方法。我们在真实数据集上进行了大量的实验，结果表明 AddGraph 在异常检测方面的性能明显优于最新的竞争对手。

1 介绍

近年来，动态图产生了迅速的发展。以电子商务网站为例，大量用户每天在网站上执行不同的操作，如点击物品、购买物品，这会导致数百万条新添加的边进入到图中。修改账户或项目的其他属性也会产生大量的内容信息。这些动态图是电子商务网站中最重要的任务（如查询和商品推荐）的基础。

异常用户可能会执行一些操作以在动态图中生成虚假数据，以实现潜在收益。这些伪数据在本文中称为异常。以推荐中的异常为例。异常用户可以通过与目标项目相关的大量新操作来提高其目标项目的受欢迎程度，例如频繁单击目标项目和受欢迎项目。因此目标项目可能会显示与其他热门项目的相似之处，这增加了推荐中的机会并提升了排名[Hooi 等人，2016]。为了快速实现该目标，异常用户通常控制多个帐户以在短时间内执行这些操作。在将数据输入以下任务之前，非常需要动态图形中的异常检测，尤其是异常边检测[Akoglu 等人，2015; Ranshous 等人，2015]。

由于异常灵活且动态，因此检测异常并非易事。一些异常操作展现出一些显式模式，但尝试将它们隐藏在大图中，其他操作则具有隐式模式。以推荐系统中的显式异常模式为例。由于异常用户通常控制多个帐户来推广目标项目，因此这些帐户和项目之间的边可能会构成一个密集的子图，该子图会在很短的时间内出现。另外，尽管涉及异常的账户有时执行异常操作，但是这些账户通常在大多数时间执行正常，这掩盖了其长期的异

常行为并增加了检测的难度。类似的异常模式出现在针对 IP-IP 网络的网络攻击中[Eswaran 等人.，2018]，在这种攻击中会突然出现大量连接，在网络中形成了非常密集的子图。这种情况表明异常的灵活性质，这要求我们通过组合所有可用的信息（例如结构，时间和内容特征）来学习异常模式。

异常检测的另一个挑战在于标记数据不足。即使初始数据正常，随着时间的流逝，异常数据也将最终与实际应用中的正常数据混合。如果我们每天手动检查异常情况，则会导致沉重的负担，甚至无法实现。即使我们可以标记一些异常操作，它们也可能占据异常的一小部分。它表明显式标记的数据可能不具有代表性，并且如果我们以监督方式学习检测模型，则会导致性能不佳。

在动态大图中检测异常的大多数现有方法都依赖于启发式规则，这些规则严格地考虑了上述特征。例如，[Hooi 等人.，2016]主要依靠结构特征。他们定义密度函数并主要利用结构特征发现目标。其他作品[Zhao 和 Yu，2013；McConville 等人.，2015]考虑了内容特征甚至时间因素。但是，考虑内容，结构和时间因素的方式并不灵活，这使其受到特定模式的限制。另外，由于其在时间维度上的稀疏性，使用长期特征检测异常更加困难。

深度学习的进步对异常检测非常有帮助，因为它能够以合理的方式组合不同的特征并从给定的数据中学习隐式规则。GCN（图卷积网络）是一种将图的内容和结构特征结合起来的代表性模型[Kipf 和 Welling，2017]。与传统的图方法相比，GCN 可以自动传播相邻节点携带的信息，然后可以用来传播节点的异常概率。在异常检测中直接使用 GCN 的主要问题在于，GCN 并未考虑时序因素，这在动态图中是不能忽略的。最新的工作，如 CAD [Sricharan 和 Das，2014]和 Netwalk [Yu 等人.，2018]，已将图嵌入方法应用于动态图。他们的方法经过精心设计，并且在检测动态图中的异常方面取得了良好的效果。但是，它们无法捕获节点的长期和短期模式，这在更通用的图模型框架检测异常时非常需要。

为了克服现有工作的局限性，本文将原始的 GCN 模型扩展，使用基于上下文注意力机制和 GRU（门控递归单元），从而支持时间信息的利用，然后在模型训练中引入选择性的负采样和边际损失逐渐增加异常边。具体来说，我们的工作的主要贡献总结如下。

•我们提出了 AddGraph，一种用于异常边检测的半监督学习框架，该框架使用扩展的时态 GCN 和基于注意力的 GRU，可以结合长期行为模式的隐藏状态和包含短期模式的窗口信息的节点。

•我们在 AddGraph 的训练中引入了选择性的负采样策略和余量损失，以检测异常边，这受知识图谱嵌入技术进步的启发。这些策略试图处理标记不足的异常数据。

•在两个真实世界的数据集上进行的实验获得了最先进的性能，这证明了 AddGraph 在检测各种图形中的异常方面的有效性。

2 相关工作

在本节中，我们回顾了现有的异常检测方法，图形嵌入模型以及在嵌入中检测异常的一些尝试。

2.1 动态图中的异常检测

Goutlier [Aggarwal 等人.，2011]方法是在观察到异常边总是出现在两个不同的节点簇之间的情况下提出的。具体来说，它首先对节点进行分区，然后为分区内的边建立边生成模型。该模型产生的分数可以用作检测异常边的重要度量。[Sun 等人.，2006; Shin 等人，2016; Shin 等人，2017 年]的工作将异常视为密集的子图。Shin [2016; 2017]在动态二分图中定义了密度函数，并采用贪婪搜索策略或序列搜索来找到这些最密集的子图。这些工作主要依赖于预先给出的模式，缺乏灵活性的结构特征。

除了结构特征，时间特征也会在异常检测中被考虑。CM-Sketch [Ranshous 等人.，2016]是一种基于草图的方法，该方法使用边附近的局部结构信息和历史行为来确定边是否异常。Spot-Light [Eswaran 等人.，2018]从整个节点集中随机采样一系列节点集，通过计算这些节点集与当前边集的节点之间的重叠，在每个时间戳上将图编码为向量。该方法通过对这些向量进行聚类来找出异常图， 但它只能捕获瞬时异常。

2.2 图嵌入

图嵌入将节点映射到 K 维向量空间，该空间保留了节点之间的某些属性。Deepwalk [Tang 等人.，2015]，LINE[Tang 等人.，2015]和 Node2vec[Grover 和 Leskovec，2016]是产生节点嵌入的方法，因此两个结构相似的节点具有相似的嵌入。这些工作的不同之处在于结构相似性的含义。Deepwalk 使用随机游走来获得节点的有序序列。LINE 尝试保留一阶相似度和二阶接近度。Node2vec 通过引入两个参数来平衡广度优先搜索和深度优先搜索来改善随机游走。这些方法可用于产生用于检测异常的节点嵌入。但是，它们主要集中在结构相似性的保留上。

GCN 和后续扩展的工作可以处理结构特征和内容特征。GCN 将卷积模型的概念从规则图（即图像）扩展到了一般图。[Defferrard 等人，2016; Kipf 和 Welling，2017]的工作从不同的角度改进了基础 GCN 的性能，例如时间/空间复杂度的优化。由于具有处理结构和内容特征的能力，GCN 可以作为我们异常检测方法的基础。但是，我们不能在工作中直接使用 GCN，因为它没有考虑动态图形中的时间特征。注意我们在本文中选择了基本的 GCN，也可以用进行较小的修改的扩展。

知识图嵌入将三元(h,r,t)投影到低维向量空间，以保留多关系数据之间潜在的相似性和差异。数据不足也是知识图嵌入的关键挑战，因为原始数据集中只有黄金三元组，这可能导致数据在不同关系中的区别微弱。为了解决这一挑战，使用负采样通过随机替换头部或尾部实体来产生边[Bordes 等人，2013]。在[Wang 等人，2014]中，伯努利分布用于负采样，它具有替换关系中首尾关系的不同概率。为了使负采样产生的三元组与黄金采样组不同，可以使用余量损失来扩大正三元组和负样本之间的差异。在本文中，我们采用类似的负采样和知识图谱嵌入的余量的思想来解决异常检测中数据不足的问题。

2.3 图嵌入异常检测

一些工作开始将图嵌入异常检测中。在[Sricharan 和 Das，2014]中，时间通勤距离用于检测动态图的异常变化，而该方法主要关注结构特征，无法捕获长期异常。[Yu等人.，2018]提出了 NetWalk，这是一种基于随机游动的动态图嵌入模型。通过节点表征的动态聚类模型实现异常检测。我们的工作大致遵循类似的想法。但是，我们将 GCN 扩展到时序 GCN，以便可以更合理地捕获时间特征，并且我们建立了端到端的半监督学习模型来检测异常边，而不是两阶段聚类， 取得更好结果的潜力。

3 提出的方法

在本节中，我们首先提出问题，提出用于异常检测的 AddGraph 框架，然后讨论其训练策略。

3.1 问题定义

设T为最大时间戳。图序列 G 的形式为$\{G^t\}_{t=1}^{T}$，其中每个$G^t = (V^t, \epsilon^t)$代表时间戳 t 的整个快照，而V^t和ϵ^t分别是节点和边的集合。边 $e = (i, j, w) \in \epsilon^t$意味着第$i$个节点和第$j$个节点在动态图上的时间戳 t 处具有权重 w。对于非加权图，w 始终为 1； 对于加权图，$w \in \mathrm{R}^+$。邻接矩阵$A^t \in \mathrm{R}^{m \times n}$表示$\epsilon^t$中的边，其中$\forall(i, j, w) \in \epsilon^t, A^t[i][j] = w$。为方便起见，令 G =(V，E)为 G 的并集，即$V = \bigcup_{t=1}^{T} V^t$且$E = \bigcup_{t=1}^{T} \epsilon^t$，我们令$n = |V|$和$m = |E|$。

本文的目的是检测ϵ^t中的异常边。具体来说，对于每个$e \in \epsilon^t$，本文产生$f(e)$，即e的异常概率。我们在训练阶段不需要异常的标记数据，而是假设初始时间戳集合中的所有边都是正常的，即t小于训练阶段的时间戳T_{train}。在测试阶段，我们使用标记的异常数据来测量以不同方法产生的$f(e)$。

3.2 AddGraph 框架

我们的 AddGraph 框架的概述如图 1 所示。AddGraph 的核心思想是通过使用训练阶段快照中的所有可能特征（包括结构，内容和时间特征）来构建一个描述正常边的框架。然后可以进一步完善框架并将其用于测量以下快照中的异常边。大致而言，AddGraph 通过考虑节点的结构和内容特征，使用 GCN 来处理当前快照中具有边的先前节点状态。然后使用基于上下文注意力的模型将短窗口中的节点状态汇总为短期信息。我们将 GCN 的输出和短期信息放入 GRU 中，以在新的时间戳上获取节点的隐藏状态。我们将在每个时间戳下使用节点的隐藏状态来计算现有边和负采样边的异常概率，然后将其馈入余量损失。

GCN 的内容和结构特征

在时间戳t处，我们收到快照$G^t = (V^t, \epsilon^t)$，具有其邻接矩阵A^t和在时间戳$t-1$处框架的输出隐藏状态矩阵$H^{t-1} \in \mathrm{R}^{n \times d}$。首先，我们利用 GCN 传播隐藏状态矩阵,有：

$$Current^t = GCN_L(H^{t-1}), \tag{1}$$

其中$Current^t$代表结合了当前输入和长期隐藏状态的节点的当前状态，而GCN_L则表

示在[Kipf 和 Welling]中提出的 L 层 GCN，2017]。GCN_L的具体细节如下：

$$Z(0) = H^{t-1}, \tag{2}$$

$$Z^{(l)} = ReLU\left(\hat{A}^t Z^{(l-1)} Z^{(l-1)}\right), \tag{3}$$

其中$l \in [1, L-1]$. $\hat{A}^t = \widetilde{D}^{-\frac{1}{2}}\tilde{A}^t\widetilde{D}^{-\frac{1}{2}}$ 是具有自循环的正则化邻接矩阵，其中$\tilde{A}^t = A^t + I_n$ 表示有自循环的邻接矩阵 , $\widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}^t$ 表示节点i的度数。

注意力 GRU 结合短期和长期状态

为了捕捉节点的短期模式，我们应用了基于上下文关注的模型，该模型受[Liu 等人，2017]的启发，并由[Cui 等人，2017]提出。在我们的框架中，我们构造了本地窗口的短状态：

$$C_{h,i}^t = [h_i^{t-\omega}; \ldots; h_i^{t-1}] C_{h,i}^t \in R^{\omega \times d} \tag{5}$$

$$e_{h,i}^t = r^T tanh\left(Q_h\left(C_{h,i}^t\right)^T\right) e_{h,i}^t \in R^{\omega} \tag{6}$$

$$a_{h,i}^t = softmax\left(e_{h,i}^t\right) a_{h,i}^t \in R^{\omega} \tag{7}$$

$$Short_i^t = \left(a_{h,i} C_{h,i}^t\right)^T short_i^t \in R^d \tag{8}$$

其中h_i^t表示第i个节点的隐藏状态，ω是捕捉短期模式的窗口大小。Q_h和r是优化基于上下文注意的模型的参数。我们将（5）-（8）式化简成一个函数：

$$Short_i^t = CAB(h_i^{t-\omega}; \ldots; h_i^{t-1}) \tag{9}$$

对于V的所有节点，该函数可以被写为：

$$Short^t = CAB(H_i^{t-\omega}; \ldots; H_i^{t-1}) \tag{10}$$

现在我们获得了$Current^t$和$Short^t$。$Current^t$表示结合当前输入和长期隐藏状态的节点的当前状态, $Short^t$表示捕捉节点短期兴趣的窗口信息。为了使 AddGraph 对时态特征进行编码，我们使用 GRU 进行处理$Current^t$和$Short^t$：

$$H^t = GRU(Current^t, Short^t) \tag{11}$$

GRU 是 LSTM 网络的一个变种。它比 LSTM 网络更简单、更有效[Chung 等，2014]。GRU 可以记录长期信息，避免梯度消失和爆炸问题。在我们的框架中，GRU 的前向传播方程是：

$$P^t = \sigma(U_P Current^t + W_P Short^t + b_P) \tag{12}$$

$$R^t = \sigma(U_R Current^t + W_R Short^t + b_R) \tag{13}$$

$$\widetilde{H}^t tanh\left(U_c Current^t + W_c(R^t \odot Short^t)\right) \tag{14}$$

$$H^t = (1 - P^t) \odot Short^t + P^t \odot \widetilde{H}^t \tag{15}$$

其中P^t是控制输出的更新门，R^t是平衡输入和内存的复位门。现在我们得到了包含结构、内容和时间特征的H^t。

边异常分数计算

现在，我们得到节点H^t在时间戳t处的隐藏状态。对于每条边$(i, j, w) \in \epsilon^t$，我们确定

了H^t中第i个节点和第j个节点的嵌入，我们可以在嵌入上计算异常分数：

$$f(i,j,w) = w \cdot \sigma\left(\beta \cdot \left(\left|\left|a \odot h_i + b \odot h_j\right|\right|_2^2 - \mu\right)\right) \tag{16}$$

其中，h_i和h_j分别是第i个和第j个节点的隐藏状态，$\sigma(x) = \frac{1}{1+e^x}$是 sigmoid 函数。$a$ 和 b 是输出层中要优化的参数。β 以及 μ 是分数函数中的超参数。注意，本文中使用的单层网络可以被其他复杂的网络所取代。

3.3 选择性负采样和丢失

为了处理异常数据的不足，我们尝试构建一个模型来描述正常数据。我们假设在训练阶段所有边都是正常的。对于图中的每条正常边，我们生成一个负样本作为异常边。受[Wang 等人，2014]中提出的方法的启发，我们定义了参数为$\frac{d_i}{d_i+d_j}$的伯努利分布，其参数i用概率$\frac{d_i}{d_i+d_j}$替换i，而j用概率$\frac{d_j}{d_i+d_j}$替换j，其中d_i和d_j表第i个结点的度数和第j个结点的度数。

由于生成的采样边可能仍然正常，因此我们无法使用严格的损失函数（例如交叉熵）来区分现有的边和生成的边。然后，我们在[Bordes 等，2013]中采用相同的思想，并在 AddGraph 的训练中使用基于边际的成对损失：

$$L^t = min \sum_{(i,j,w)\in\epsilon^t} \sum_{(i',j',w)\notin\epsilon^t} max\{0, \gamma + f(i,j,w) - f(i',j',w)\} \tag{17}$$

其中，$f(\cdot,\cdot,\cdot)$是边的异常得分函数，而$\gamma \in (0,1)$是法线边和异常边的可能性之间的余量。损失函数L_t的最小化促使$f(i,j,w)$变小而 $f(i',j',w)$变大，这与我们的预期一致。我们不能假设训练阶段后快照中的所有边仍然完全正常，而我们需要为每个快照计算隐藏状态。在混合了正常边和异常边的图形上进行采样的过程中，我们实际上选择了对于训练更可靠的部分边。具体来说，对于每个边(i, j, w)，我们产生一个负采样边(i',j',w)。如果$f(i,j,w) > f(i',j',w)$，则丢弃采样的边对。选择性否定采样策略可确保 AddGraph 框架长期稳定。

最终损失函数被归纳如下：

$$L = L^t + \lambda L_{reg} \tag{18}$$

其中 λ 是一个超参数，L_{reg}是为了防止过拟合的 L2 正则化损失,定义如下：

$$\begin{aligned} L_{reg} = X(&||W_1||_2^2 + ||W_2||_2^2 + ||Q_h||_2^2 + ||r||_2^2 + ||U_z||_2^2 + ||W_z||_2^2 + ||b_z||_2^2 + ||U_r||_2^2 \\ &+ ||W_r||_2^2 + ||b_r||_2^2 + ||U_c||_2^2 + ||U_c||_2^2 + ||a||_2^2 + ||b||_2^2) \end{aligned} \tag{19}$$

我们总结上述算法如算法 1 所示。

4 实验

在本节中，我们首先描述实验设置，然后将 AddGraph 与其他方法进行比较。

4.1 实验设置数据集

我们在两个数据集上评估了我们的框架，这两个数据集的详细信息如表 2 所示。UCI 消息是一个有向的网络，其中包含加利福尼亚大学欧文分校的一个在线社区中的消息。每个节点代表一个用户，每条有向边表示两个用户之间的消息。Digg 是社交新闻网站 Digg 的响应网络。网络中的每个节点都是该站点的用户，每条边都表示一个用户答复另一个用户。两个数据集中的边均带有时间戳。我们为每个节点随机生成一个初始向量作为其内容特征。我们需要手动建立所需的数据集，因为很难获得测试阶段的真实性[Akoglu 等人，2015]，并且我们遵循[Yu 等人，2018]中使用的方法将异常边注入两个数据集。

基线方法

我们将 AddGraph 与三种异常检测方法进行了比较。

•GOutlier [Aggarwal 等人，2011]。它为节点簇中的边建立了一个生成模型，该模型还可以用于生成给定边的异常分数。

•CM-Sketch [Ranshous 等人，2016]。它使用边附近的局部结构特征和历史行为来衡量边是否异常。

•NetWalk [Yu 等人，2018]。该方法首先基于随机游走构建节点嵌入，然后使用节点嵌入上的聚类检测异常。

实验设计

我们将在没有时间戳的情况下测试图上的异常检测方法，以查看框架是否可以有效地利用内容和结构特征，然后扩展到具有所有特征的动态图。我们将研究不同参数对 AddGraph 的影响。用于比较不同方法性能的度量标准是 AUC（ROC 曲线下的面积）。

4.2 实验结果

无时间戳图的结果

在没有时间戳的图上的测试主要集中在结构和内容特征的探索上。对于 AddGraph，本实验执行的是没有窗口信息的简化版本。我们将数据集分为两部分，前 50％作为训练数据，后 50％作为测试数据。GCN 层数为 2。正则化的权重衰减 λ 为 5e-7。学习率lr为 0.002。辍学率为 0.2。对于 UCI 消息数据集，隐藏状态的维度大小为 500。余量 γ 被设置为 0.5。参数 β 和 μ 分别设置为 1.0 和 0.3。对于 Digg 数据集，隐藏状态的尺寸为 200。裕度 γ 被设定为 0.7。参数 β 和 μ 分别设置为 3.0 和 0.5。我们将异常数据的 1％，5％和 10％注入到不同数据集的测试数据中。

结果示于表 1，其中基线数据由[Yu 等人，2018]提供。我们可以看到，AddGraph 击败了两个数据集上具有不同异常比例的所有基线方法，并且具有更好的捕获结构和内容特征的能力。这主要归因于基础的图卷积网络，它使框架能够更好地在节点及其邻居之间传播信息。特别是在 Digg 数据集上，我们的方法已获得 10％以上的改进。这种出色的效果证明我们的框架可以有效地利用内容和结构特征。与基准结果的趋势不同，异常情况为 1％的框架所产生的 AUC 略低于异常情况为 5％的 AUC，但仍高于所有基准。这可

能是由于以下事实：当测试集的异常比例较低时，在原始数据和负样本上计算出的分数无法很好地区分它们，从而导致预测准确性下降。

动态图结果

在动态图测试中，我们将前 50%用作训练数据，将后 50%用作测试数据。在以 5%的比例异常注入后，我们将训练数据和测试数据分成快照。根据数据集的大小，UCI Message 和 Digg 的快照大小分别设置为 1,000 和 6,000。在训练阶段，我们使用训练数据的快照来构建初始模型。在测试阶段，我们将在时间戳 t 的每个快照到来时逐步维护模型。GCN 层数设置为 3。学习率 lr 为 0.001。对于 UCI 消息数据集，隐藏状态的维度大小为 100。对于 Digg 数据集，隐藏状态的尺寸为 50。余量 γ 被设置为 0.6。其余参数具有与上述相同的值。

图 2 显示了动态图的结果，其中基线数据由[Yu 等人，2018]提供。结果表明，除了 Digg 中的最后一个快照外，AddGraph 几乎都超过了所有快照的基线。预测结果表明我们的框架能够捕获时间特征。注意力机制使我们的框架能够在窗口期内注意到节点的变化。具有 GRU 的扩展 GCN 使得记录长期依赖关系成为可能。上次快照减少的原因可能是由于在测试阶段出现了太多新节点，这些新节点之前并未得到处理。

参数敏感性

现在，我们尝试找出超参数对 AddGraph 的影响，包括 L 代表 GCN 中的总层数，d 代表隐藏状态维的大小，以及整个数据集的训练率。

首先，我们评估 L 和 d 的影响。L 的范围是{1、2、3、4、5}，d 的范围是{10、25、50、100、200}。其他参数设置为最佳。为了显示不同参数值的影响，我们在这项研究中选择了一个相对更具挑战性的任务。我们将 Digg 数据集与 10%的异常数据用于测试数据，并根据 AUC 指标在最后一个时间戳上检测异常。为了便于观察，我们使用 log（d）而不是 d 作为 x 轴。如图 3 所示，当 L 从 1 增加到 2 时，AUC 显着增加，当 L 为 3 时达到其峰值。随着 d 的增加，AddGraph 的性能逐渐提高，并且当 d 为 50 时达到最佳值。在 d 和 L 达到最佳配置后，AUC 会随着它们的增加而减小。当层数过多时，GCN 可能会捕获远程邻居的无用信息，从而降低了框架的准确性。较大的 d 将增加框架的复杂性，并使其更难以收敛到最佳点。

其次，我们评估了整个数据集的训练比率的影响。训练比率的范围为{10%，20%，30%，40%，50%，60%}，其他参数设置为最佳。我们在测试数据中使用异常率 10%的 Digg 数据集，并在测试阶段记录每个时间戳记的 AUC 分数。如图 4 所示，随着训练比率的降低，AddGraph 的平均和最大 AUC 值呈上升趋势，而最小值表示下降趋势。由于训练集中没有异常数据，因此较高比例的测试数据包含更多异常边，这使 AddGraph 可以增加正负样本得分之间的距离。这些结果表明，我们的框架具有强大的能力，可以在训练数据不足的情况下检测动态图中的异常边。最小值的减少归因于训练数据的减少，这降低了 Addgraph 的稳定性。

5 结论

我们在动态图上提出了异常检测框架 AddGraph，它可以灵活地检测异常模式，而无需显式标记异常数据。AddGraph 尝试通过使用时态 GCN 和基于上下文注意力的模型来考虑所有可能的提示，包括结构，内容和时态特征，从而学习异常模式。它还在训练中采用了选择性的负采样策略和边距损失，以处理标记不足的异常数据。在多个现实世界数据集上的实验表明，AddGraph 明显优于其他现有的异常检测方法。

致谢

本研究得到了国家重点研究开发项目 2016YFB1000700、国家自然科学基金项目 61572040 和 61832001 以及阿里巴巴-北大联合项目的部分资助。

论文原文

地址：https://www.ijcai.org/Proceedings/2019/0614.pdf

引用：Zheng L, Li Z, Li J, et al. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN[C]//IJCAI, 2019: 4419–4425.

AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN

Li Zheng, Zhenpeng Li , Jian Li , Zhao Li and Jun Gao
The Key Laboratory of High Confidence Software Technologies, Ministry of Education, China
School of EECS, Peking University, China
Alibaba Group, China
{greezheng, gaojun}@pku.edu.cn, {zhen.lzp,zeshan.lj,lizhao.lz}@alibaba-inc.com

Abstract

Anomaly detection in dynamic graphs becomes very critical in many different application scenarios, e.g., recommender systems, while it also raises huge challenges due to the high flexible nature of anomaly and lack of sufficient labelled data. It is better to learn the anomaly patterns by considering all possible hints including the structural, content and temporal features, rather than utilizing heuristic rules over the partial features. In this paper, we propose AddGraph, a general end-to-end anomalous edge detection framework using an extended temporal GCN (Graph Convolutional Network) with an attention model, which can capture both long-term patterns and the short-term patterns in dynamic graphs. In order to cope with insufficient explicit labelled data, we employ a selective negative sampling and margin loss in training of AddGraph in a semi-supervised fashion. We conduct extensive experiments on real-world datasets, and illustrate that AddGraph can outperform the state-of-the-art competitors in anomaly detection significantly.

1 Introduction

The recent years witness the rapid development of dynamic graphs. Taking the e-commerce sites as an example. Massive users perform different operations, such as item clicking, item buying, in the sites every day, which contribute to millions of newly-added edges into the graph. The modification of other attributes for accounts/items also produces a large amount of content information. These dynamic graphs serve as the basis for the most important tasks in the e-commerce sites like the query and item recommendation.

Anomalous users may perform some operations to generate fake data in the dynamic graphs to achieve the potential gain. These fake data are called anomaly in this paper. Taking the anomaly in the recommendation as an example. Anomalous users can improve the popularity of their target items through a large number of new operations related to target items, like clicking both target items and popular ones frequently. Then, the target items may show some similarities to other popular ones, which increases the chances and upgrade rankings in the recommendation [Hooi et al., 2016]. In order to achieve the goal quickly, anomalous users usually control multiple accounts to perform these operations in a short time period. The anomaly detection in dynamic graph, especially anomalous edges detection, is then highly needed before the data are fed into the following tasks [Akoglu et al., 2015; Ranshous et al., 2015].

It is not trivial to detect the anomaly due to its flexible and dynamic nature. Some anomalous operations show some explicit patterns but try to hide them in a large graph, while others are with implicit patterns. Take an explicit anomaly pattern in the recommender system as an example. As anomalous users usually control multiple accounts to promote the target items, the edges between these accounts and items may compose a dense subgraph, which emerge in a short time period. In addition, although the accounts which involve the anomaly perform anomalous operations sometimes, these accounts perform normally most of the time, which hides their long-term anomalous behavior and increases the difficulty of detection. The similar anomaly pattern appears in the network attack against IP-IP network [Eswaran et al., 2018], where there are sudden large number of connections, forming a very dense subgraph in the network. Such cases indicate the flexible nature of anomaly, which requires us to learn the anomaly patterns by combining all available hints like structural, temporal and content features.

Another challenge in the anomaly detection lies in the insufficient labelled data. Even if the initial data are normal, anomaly data will be finally mixed with the normal ones in the real-world applications as time goes by. It results in high burden or is even infeasible if we check the anomaly every day by hand. Even if we can label some anomaly operations, they may occupy a small part of anomalies. It indicates that the explicit labelled data may be not representative, and results in the poor performance if we learn a detection model in a supervised way.

Most of existing approaches to detecting the anomalies in dynamic large graphs rely on the

heuristic rules which consider the above features in a rigid way. For example, [Hooi et al., 2016] mainly relies on the structural features. They define a density function and discover the target mainly using structural features. Other works [Zhao and Yu, 2013; McConville et al., 2015] consider content feature or even temporal factor. However, the way taking the content, structural and temporal factors into account is not flexible, which makes it restricted in a specific pattern. In addition, it is more difficult to detect anomalies using the long-term features due to their sparsity in the time dimension.

The advance of deep learning is very helpful in anomaly detection, with its ability to combine different features in a reasonable way and to learn implicit rules from the given data. GCN(Graph Convolutional Network) is a representative model to combine the content and structural features in a graph [Kipf and Welling, 2017]. Compared with traditional graph methods, GCN can automatically propagate the information carried by neighboring nodes, which can then be used to spread the anomalous probabilities of nodes. The major issue of direct usage of GCN in the anomaly detection lies in the fact that GCN does not consider the timing factors, which cannot be ignored in the dynamic graphs. The more recent works, like CAD [Sricharan and Das, 2014] and Netwalk [Yu et al., 2018] have applied the graph embedding method to dynamic graph. Their methods are well designed and have achieved good results on detecting anomalies in dynamic graphs. However, they cannot capture the long-term and short-term patterns of nodes, which are highly needed in a more general graph model framework to detect anomalies.

In order to overcome the limitations of the existing works, this paper extends the original GCN model to support temporal information using GRU(Gated Recurrent Unit) with a contextual attention-based model, and then introduces a selective negative sampling and margin loss in model training for anomalous edges incrementally. Specifically, the main contributions of our work are summarized as follows.

• We propose AddGraph, a semi-supervised learning framework for anomalous edge detection, using an extended temporal GCN with an attention-based GRU, which can combines the hidden states for long-term behavior patterns and the window information containing the short-term patterns of the nodes.

• We introduce a selective negative sampling strategy and margin loss in the training of AddGraph for detecting anomalous edges, inspired by the advances in embedding of knowledge graph. Those strategies attempt to handle the insufficient labelled anomaly data.

• Experiments on two real-world datasets achieve stateof-the-art performance, which proves the effectiveness of AddGraph on detecting anomalies in different kinds of graphs.

2 Related Work

In this section, we review the existing anomaly detection approaches, the graph embedding

model, and some attempts to detect anomaly on embeddings.

2.1 Anomaly Detection in Dynamic Graph

Goutlier [Aggarwal et al., 2011] is proposed with an observation that anomalous edges always appear between two different node clusters. Specifically, it first partitions nodes, and then builds an edge generative model for the edges inside partition. The scores produced by the model can be used as an important measure in detecting anomalous edges. The works [Sun et al., 2006; Shin et al., 2016; Shin et al., 2017] view the anomaly as a dense sub-graph. Shin [2016; 2017] define a density function in the dynamic bipartite graph, and employ a greedy search strategy or sequence search to find these most dense sub-graphs. These works mainly rely on structural features lacking in flexibility as patterns are given in advance.

Besides the structural features, the temporal ones are considered in the anomaly detection. CM-Sketch [Ranshous et al., 2016] is a sketch-based method, which uses the local structural information and historical behavior near an edge to decide whether the edge is anomalous or not. Spot-Light [Eswaran et al., 2018] randomly samples a series of node sets from the entire node set, and encodes the the graph at each timestamp to a vector by computing the overlap between these sets and the nodes of current edge set. The method finds out the anomalous graph by clustering these vectors. However, it can only catch instantaneous anomalies.

2.2 Graph Embedding

Graph embedding maps the nodes into a K-dimensional vector space, which preserves certain properties among nodes. Deepwalk [Tang et al., 2015], LINE [Tang et al., 2015] and Node2vec [Grover and Leskovec, 2016] are the methods to yield node embeddings so that two structural similar nodes have the similar embeddings. The difference of these works lies in the meaning of structural similarities. DeepWalk uses random walk to get an ordered sequence of nodes. LINE attempts to preserve the first-order similarity and the second order proximity. Node2vec improves random walk by introducing two parameters to balance the breadth-first search and depth-first search. Those methods can be used to yield node embeddings for detecting anomaly. However, they mainly focus on the preservation of structural similarity.

The work of GCN and following extensions can process structural features and content features. GCN extends the idea of convolution model over regular graphs (i.e., image) to general graphs. The works [Defferrard et al., 2016; Kipf and Welling, 2017] improve the performance of basic GCN from different viewpoints, like the optimization in time/space complexity. Due to its ability to handle both structural and content features, GCN can be leveraged as the basis in our anomaly detection approach. However, we cannot directly use GCN in our work, as it does not consider temporal features in dynamic graphs. Note that we choose the basic GCN in this paper, and its extensions can also be used with minor modification.

Knowledge graph embedding projects a triple (h, r, t) to low-dimensional vector spaces to preserve potential similarities and differences between multi-relational data. Insufficient data is also a key challenge for knowledge graph embedding, because there are only golden triples in the original dataset, which may result in weak distinction of data in different relationships. In order to solve this challenge, negative sampling is used to produce edges by randomly replacing the head or tail entities [Bordes et al., 2013]. In [Wang et al., 2014], a Bernoulli distribution is used for negative sampling, which holds different probabilities to replace head and tail in relations. To make the triple produced by negative sampling different from the golden one, margin loss is used to enlarge the difference between positive triples and negative samples. In this paper, we use a similar idea of negative sampling and margin loss from knowledge graph embedding to solve the problem of insufficient data in anomaly detecting.

2.3 Anomaly Detection over Graph Embedding

Some works began to combine the graph embedding into the anomaly detection. In [Sricharan and Das, 2014], a time-commute distance is used to detect anomalous changes in dynamic graph, while the method mainly focuses on structural features and cannot catch long-term anomalies. [Yu et al., 2018] proposed NetWalk, a dynamic graph embedding model based on random walks. The anomaly detection is realized by the dynamic clustering model of node representation. Our work roughly follows a similar idea. However, we extend GCN to the temporal GCN so that we can capture the temporal features in a more reasonable way, and we build an end-to-end semi-supervised learning model to detect anomalous edges rather than two-phases clustering, which has the potential to achieve better results.

3 Proposed Method

In this section, we first formulate the problem, propose an AddGraph framework for anomaly detection, and then discuss its training strategies.

3.1 Problem Definition

Let T be the maximum timestamp. A graph steam G takes the form of $\{G^t\}_{t=1}^{T}$ where each $G^t = (V^t, \epsilon^t)$represents the entire snapshot at timestamp t, and V^t and ϵ^t are the set of nodes and edges respectively. An edge $e = (i, j, w) \in \epsilon^t$ means that the i-th node and the j-th node have a connection in the dynamic graph at the timestamp t with its weight w. For unweighted graphs, w is always 1; for weighted graphs, $w \in \mathrm{R}^+$. An adjacency matrix $A^t \in \mathrm{R}^{m \times n}$ is to represent the edges in ϵ^t, where $\forall(i, j, w) \in \epsilon^t, A^t[i][j] = w$. For convenience, let G =(V， E) be the union of G, i.e., $V = \bigcup_{t=1}^{T} V^t$and $E = \bigcup_{t=1}^{T} \epsilon^t$. We let $n = |V|$ and $m = |E|$.

The goal of this paper is to detect anomalous edges in ϵ^t. Specifically, for each $e \in \epsilon^t$, this paper produces $f(e)$, the anomalous probability of e. We do not need the labelled data for anomaly in the training phase, but assume that all edges in sets at the initial timestamps are normal, i.e., t

is smaller than the timestamp T_{train} in training phase. In the test phase, we use the labelled anomaly data to measure $f(e)$ produced in different methods.

3.2 AddGraph Framework

The overview of our AddGraph framework is illustrated in Figure 1. The core idea behind AddGraph is to build a framework to describe the normal edges by using all possible features in the snapshots in the training phase, including structural, content and temporal features. The framework then can be further refined and used to measure the anomalous edges in the following snapshots. Roughly, AddGraph employs GCN to process the previous node state with edges in the current snapshot by considering the structural and content features of nodes. The node states in a short window are then summarized as the short-term information with a contextual attention-based model. We put the output of GCN and short term information into GRU to get the hidden state of nodes at a new timestamp. We will use the hidden state of the nodes at each timestamp to calculate the anomalous probabilities of an existing edge and a negative sampled edge, and then feed them to a margin loss.

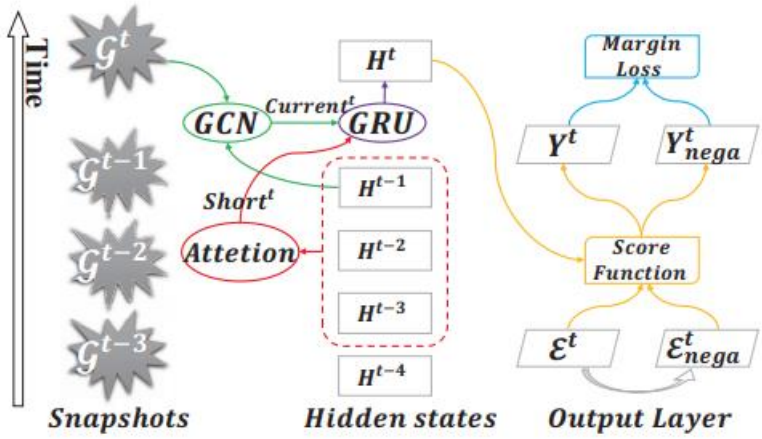


Figure 1: AddGraph framework

GCN for content and structural features.

At timestamp t, we receive the snapshot $G^t = (V^t, \epsilon^t)$ with its adjacency matrix A^t and the output hidden state matrix $H^{t-1} \in \mathrm{R}^{n \times d}$ of the framework at timestamp $t-1$. First, we propagate the hidden state matrix with GCN,

$$Current^t = GCN_L(H^{t-1}), \tag{1}$$

where $Current^t$ represents current state of nodes combining the current input with the long-term hidden state, and GCN_L denotes an L-layered GCN which is proposed in [Kipf and Welling, 2017]. The details of GCN_L are shown below:

$$Z(0) = H^{t-1}, \tag{2}$$

$$Z^{(l)} = ReLU\left(\hat{A}^t Z^{(l-1)} Z^{(l-1)}\right), \tag{3}$$

$$Current^t = ReLU\left(\hat{A}^t Z^{(L-1)} Z^{(L-1)}\right), \tag{4}$$

where $l \in [1, L-1]$. $\hat{A}^t = \widetilde{D}^{-\frac{1}{2}} \tilde{A}^t \widetilde{D}^{-\frac{1}{2}}$ is the regularized adjacency matrix with self loops, where$\tilde{A}^t = A^t + I_n$ denotes the adjacency matrix with self loops, and $\widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}^t$ denotes the degree of node i.

GRU with attention to combine short-term and long-term states.

To catch the short-term pattern of nodes, we apply the contextual attention-based model which is inspired by [Liu et al., 2017] and proposed by [Cui et al., 2017]. In our framework, we construct short state of local window as follow:

$$C_{h,i}^t = [h_i^{t-\omega}; \dots; h_i^{t-1}] C_{h,i}^t \in R^{\omega \times d} \tag{5}$$

$$e_{h,i}^t = r^T tanh\left(Q_h \left(C_{h,i}^t\right)^T\right) e_{h,i}^t \in R^{\omega} \tag{6}$$

$$a_{h,i}^t = softmax\left(e_{h,i}^t\right) a_{h,i}^t \in R^{\omega} \tag{7}$$

$$Short_i^t = \left(a_{h,i} C_{h,i}^t\right)^T short_i^t \in R^d \tag{8}$$

whereh_i^t denotes the hidden state of the i-th node, and ω is the size of the window to catch short-term pattern.$Q_h$and r are parameters to optimize the contextual attention-based model. We brief (5) − (8) to a single function:

$$Short_i^t = CAB(h_i^{t-\omega}; \dots; h_i^{t-1}) \tag{9}$$

For all nodes in V, the function is written as:

$$Short^t = CAB(H_i^{t-\omega}; \dots; H_i^{t-1}) \tag{10}$$

Now we get $Current^t$ and $Short^t$. $Current^t$ represents current states of nodes which combine the current input with the long-term hidden state, and $Short^t$ represents the window information which catches the short-term interest of nodes. To make AddGraph encode temporal features, we use GRU to process $Current^t$ and $Short^t$:

$$H^t = GRU(Current^t, Short^t) \tag{11}$$

GRU is a variant of LSTM network. It is simpler and more effective than LSTM network [Chung et al., 2014]. GRU can record long-term information, and avoid gradient vanishing and exploding problems. The forward propagation equations of GRU in our framework are:

$$P^t = \sigma(U_P Current^t + W_P Short^t + b_P) \tag{12}$$

$$R^t = \sigma(U_R Current^t + W_R Short^t + b_R) \tag{13}$$

$$\widetilde{H}^t tanh\left(U_c Current^t + W_c(R^t \odot Short^t)\right) \tag{14}$$

$$H^t = (1 - P^t) \odot Short^t + P^t \odot \widetilde{H}^t \tag{15}$$

where P^t is the update gate to control output and R^t is the reset gate to balance input and memory. Now we get H^t containing the structural, content and temporal features.

Anomalous score computation for edges.

Now, we get the hidden state of nodes H^t at timestamp t. For each edge $(i,j,w) \in \epsilon^t$, we locate the embeddings for the i-th node and the j-th node in H^t , on which we can compute the anomalous scores:

$$f(i,j,w) = w \cdot \sigma\left(\beta \cdot \left(\left|\left|a \odot h_i + b \odot h_j\right|\right|_2^2 - \mu\right)\right) \tag{16}$$

where h_i and h_j are the hidden state of the i-th and j-th node respectively, and $\sigma(x) = \frac{1}{1+e^x}$ is the sigmoid function. a and b are parameters to optimize in the output layer. β and μ are the hyper-parameters in the score function. Note that the single layer network used in this paper can be replaced by other sophisticated networks.

3.3 Selective Negative Sampling and Loss

In order to handle the insufficiency of anomaly data, we try to build a model to describe the normal data instead. Recall that we assume that all edges are normal in the training phase. For each normal edge in the graph, we generate a negative sample as an anomalous edge. Inspired by the method proposed in [Wang et al., 2014], we define a Bernoulli distribution with parameter$\frac{d_i}{d_i+d_j}$ for sampling: given a normal edge (i,j), we replace i with probability $\frac{d_i}{d_i+d_j}$ and replace j with probability $\frac{d_j}{d_i+d_j}$, where d_i and d_j denote the degree of the i-th node and the j-th node respectively.

As the generated sampled edges may be still normal, we cannot use a strict loss function such as cross entropy to distinguish the existing edges and the generated ones. We then take the same idea in [Bordes et al., 2013] and use margin-based pairwise loss in training of AddGraph:

$$L^t = min \sum_{(i,j,w)\in\epsilon^t} \sum_{(i',j',w)\notin\epsilon^t} max\{0, \gamma + f(i,j,w) - f(i',j',w)\} \tag{17}$$

where $f(\cdot,\cdot,\cdot)$ is the anomalous score function for edges, and $\gamma \in (0,1)$ is the margin between the possibilities of normal edge and anomalous one. The minimization of the loss function $L_t$encourages that $f(i,j,w)$ becomes smaller while $f(i',j',w)$ becomes larger, which is in the same line with our expectation.

We cannot assume that all edges are still completely normal in the snapshots after training phase, while we need to compute the hidden states for each snapshot. In the process of sampling on the graph mixed with normal and anomalous edges, we actually select partial edges which are more credible for the training. Specifically, for each edge (i, j, w), we produce a negative sampled edge (i',j',w). The sampled edge pair is discarded if $f(i,j,w) > f(i',j',w)$. The selective negative sampling strategy ensures the stability of AddGraph framework for a long time.

The overall loss function is summarized as follows:

$$L = L^t + \lambda L_{reg} \tag{18}$$

where λ is a hyper-parameter, and L_{reg} is an L2- regularization loss to avoid overfitting, which is shown as follows:

$$L_{reg} = X(||W_1||_2^2 + ||W_2||_2^2 + ||Q_h||_2^2 + ||r||_2^2 + ||U_z||_2^2 + ||W_z||_2^2 + ||b_z||_2^2 + ||U_r||_2^2 + ||W_r||_2^2 + ||b_r||_2^2 + ||U_c||_2^2 + ||U_c||_2^2 + ||a||_2^2 + ||b||_2^2) \tag{19}$$

We summarize our algorithm as Algorithm 1.

Algorithm 1 AddGraph algorithm

Input: Edge stream $\{\mathcal{E}^t\}_{t=1}^T$
Parameter: $\beta, \mu, \lambda, \gamma, L, \omega, d$
Output: $\{\mathbf{H}^t\}_{t=1}^T$

Initialize $\mathbf{H}^0$
repeat
  for $t = 1$ to T **do**
    Let $\mathcal{L}^t = 0$
    $\mathbf{Current}^t = \mathrm{GCN}(\mathbf{H}^{t-1})$
    $\mathbf{Short}^t = \mathrm{CAB}(\mathbf{H}^{t-w}; ...; \mathbf{H}^{t-1})$
    $\mathbf{H}^t = \mathrm{GRU}(\mathbf{Current}^t, \mathbf{Short}^t)$
    for all $(i, j, w) \in \mathcal{E}^t$ **do**
      Sample(i', j', w) for $f(i, j, w)$
      $\mathcal{L}^t = \mathcal{L}^t + \max(0, \gamma + f(i, j, w) + f(i', j', w))$
    end for
    $\mathcal{L}^t = \mathcal{L}^t + \mathcal{L}_{reg}$
    Minimize $\mathcal{L}^t$
  end for
until Convergence
return $\{\mathbf{H}^t\}_{t=1}^T$

4 Experiment

In this section, we first describe the experimental setup, and then compare AddGraph with other competitors.

4.1 Experimental Setup

Dataset. We evaluate our framework on two datasets and the details of these two datasets are shown in Table 2. UCI Message is a directed network containing messages among an online community at University of California, Irvine. Each node represents a user and each directed edge is for a message between two users. Digg is a response network of Digg, a social news site. Each node in the network is a user of the site, and each edge indicates that one user replies to another. Edges in both datasets are annotated with timestamps. We randomly generate an initial vector for each node as its content feature. We need to manually build the required datasets because the

ground-truth for the test phase is difficult to obtain [Akoglu et al., 2015], and we follow the approach used in [Yu et al., 2018] to inject anomalous edges into two datasets.

Baselines.

We compare AddGraph with three anomaly detection methods.

• GOutlier [Aggarwal et al., 2011]. It builds a generative model for edges in a node cluster, and the model can also be used to produce anomalous score for a given edge.

• CM-Sketch [Ranshous et al., 2016]. It uses the local structural feature and historical behavior near an edge to measure whether the edge is anomalous or not.

• NetWalk [Yu et al., 2018]. The method first builds node embeddings based on random walks, and then detects anomaly using the clustering on the node embedddings.

Experimental Design.

We will test the anomaly detection methods over graphs without timestamps to see whether the framework can exploit the content and structural features effectively, and then extend to the dynamic graphs with all features. We will study the impacts of different parameters on AddGraph. The metric used to compare the performance of different methods is AUC (the area under the ROC curve).

4.2 Experimental Results

Dataset	Anomaly proportion	GOutlier	CM-Sketch	Netwalk	AddGraph
UCI Message	1%	0.7181	0.7270	0.7758	**0.8083**
	5%	0.7053	0.7086	0.7647	**0.8090**
	10%	0.6707	0.6861	0.7226	**0.7688**
Digg	1%	0.6963	0.6871	0.7563	**0.8341**
	5%	0.6763	0.6581	0.7176	**0.8470**
	10%	0.6353	0.6179	0.6837	**0.8369**

Table 1: AUC results for anomalous scores on graphs without timestamps

Dataset	#Node	#Edge	Max. Degree	Avg. Degree
UCI Message	1,899	13,838	255	14.57
Digg	30,360	85,155	283	5.61

Table 2: Statistics of Datasets

Results on Graphs without Timestamps

The tests over graphs without timestamps mainly focus on the exploration of structural and content features. As for AddGraph, a simplified version without window information is conducted for this experiment. We divide the dataset into two parts, the first 50% as the training data and the latter 50% as the test data. The number of GCN layers is 2. The weight decay λ for regularization

is 5e-7. The learning rate lr is 0.002. The dropout rate is 0.2. For UCI Message dataset, the size of dimension is 500 for hidden state. The margin γ is set to 0.5. The parameters β and μ is set to 1.0 and 0.3 respectively. For Digg dataset, the size of dimension is 200 for hidden state. The margin γ is set to 0.7. The parameters β and μ is set to 3.0 and 0.5 respectively. We injected 1%, 5%, and 10% of the anomalous data into the test data of different datasets.

The results are shown in Table 1, in which the data of baselines are reported by [Yu et al., 2018]. We can see that AddGraph beats all baselines on the two datasets with varying anomaly proportions and has better ability to catch structural and content features. This is mainly due to the underlying GCN, which enables the framework to spread information between nodes and its neighbors better. In particular, on Digg dataset, our approach has gained more than 10% improvement. This outstanding effect proves that our framework can exploit the content and structural features effectively. Unlike the trend of baselines' results, AUC produced by our framework with 1% anomaly is a little lower than that with 5% anomaly, while it's still higher than all baselines. This may be due to the fact that when the anomaly proportion of the test set is low, the scores computed on the original data and negative samples cannot distinct them well, resulting in a decline of prediction accuracy.

Results on Dynamic Graphs

In the tests over dynamic graphs, we use the first 50% as the training data and the latter 50% as the test data. After anomaly injection with proportion of 5%, we split the training data and test data into snapshots. According to the size of dataset, the snapshot size is set to 1,000 and 6,000 for UCI Message and Digg respectively. In training phase, we use snapshots of training data to build an initial model. In test phase, we maintain the model incrementally as each snapshot at timestamp t arrives. The number of GCN layers is set to 3. The learning rate lr is 0.001. For UCI Message dataset, the size of dimension is 100 for hidden states. For Digg dataset, the size of dimension is 50 for hidden states. The margin γ is set to 0.6. The rest parameters are with the same values as above.

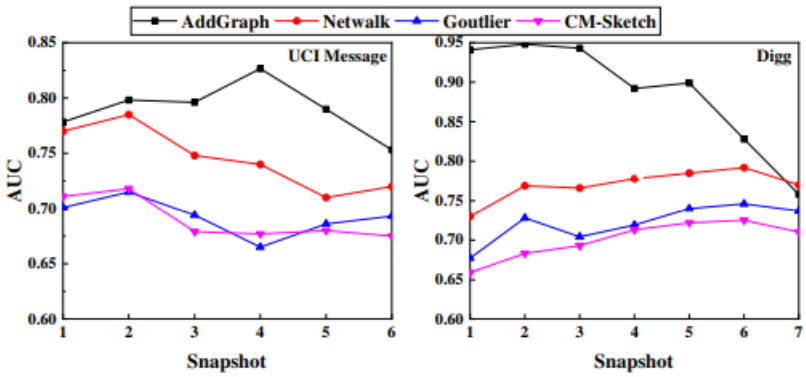


Figure 2: AUC results for anomalous scores on dynamic graphs

Figure 2 illustrates the results on dynamic graph, in which the data of baselines are reported by [Yu et al., 2018]. The results indicate that AddGraph beats baselines on almost all snapshots except the last one in Digg. The prediction results show our framework is able to catch temporal features. The attention mechanism enables our framework to notice the changes of nodes during the window period. The extended GCN with GRU makes it possible to record long-term dependency. The decline on the last snapshot may be due to the fact that too many new nodes emerge in the test phase, which are not processed before.

Parameter Sensitivity

Now, we attempt to find out the influence of hyper-parameters on AddGraph, including L for the total number of layers in GCN, d for the size of dimension of hidden state, and the training ratio of the entire dataset.

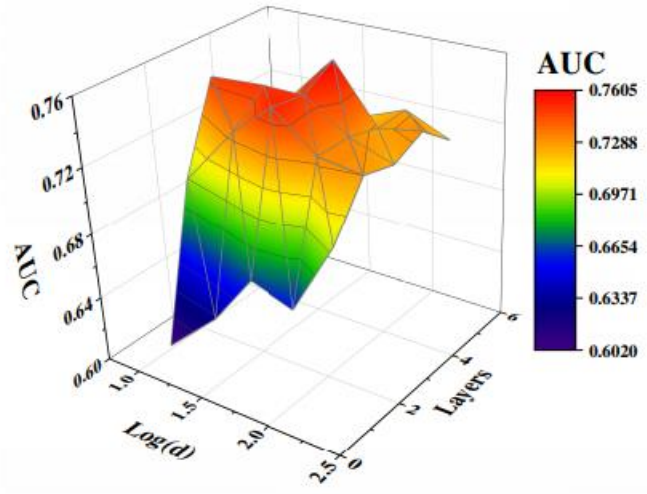


Figure 3: AUC results on Digg with different parameters

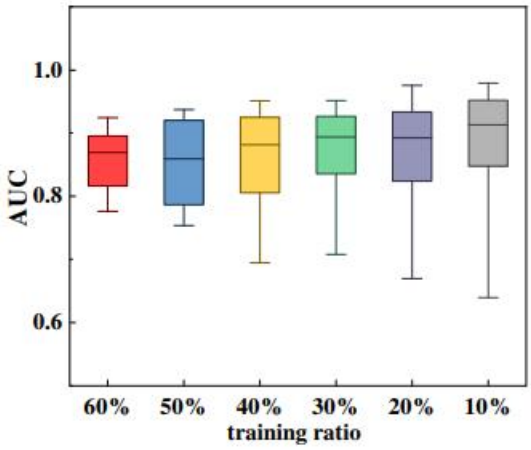


Figure 4: AUC results on Digg with different training ratios

First, we evaluate the influence of L and d. The range of L is {1, 2, 3, 4, 5} and the range of

d is {10, 25, 50, 100, 200}. Other parameters are set to optimum. In order to show the influence of different values of the parameters, we choose a relatively more challenging task in this study. We use Digg dataset with 10% anomalies in test data and detect anomalies at the last timestamp in terms of the AUC metric. For ease of observation, we use log(d) instead of d as the x-axis. As shown in Figure 3, AUC increases significantly when L increases from 1 to 2, and reaches its peak when L is 3. With the increase of d, the performance of AddGraph is gradually improved, and reaches the optimal value when d is 50. After d and L reach the optimal configuration, AUC decreases as they continue to increase. When there are too many layers, GCN may capture useless information of remote neighbors, which reduces the accuracy of the framework. Larger d will increase the complexity of the framework and make it more difficult in converging to the optimal point.

Second, we evaluate the influence of the training ratio of the entire dataset. The range of training ratio is {10%, 20%, 30%, 40%, 50%, 60%} and other parameters are set to optimum. We use Digg dataset with 10% anomalies in test data and record the AUC score of each timestamp at the test phase. As shown in Figure 4, with the decrease of training ratio, the average and maximum AUC values of AddGraph show an upward trend, while the minimum values indicate a downward trend. Since there is no anomalous data in the training set, a higher proportion of test data contains more anomalous edges, which enables AddGraph to increase the distance between the scores of positive and negative samples. These results show the strong ability of our framework to detect anomalous edges in dynamic graphs with insufficient training data. The decrease of the minimum value is due to the reduction of the training data, which reduces the stability of Addgraph.

5 Conclusion

We propose an anomaly detection framework, AddGraph, on dynamic graphs which can detect the patterns of anomalies flexibly without explicit labelled anomaly data. AddGraph attempts to learn the anomaly patterns by considering all possible hints including the structural, content and temporal features using the temporal GCN with a contextual attention based model. It also employs a selective negative sampling strategy and margin loss in training to handle the insufficient labelled anomaly data. Experiments on several real world datasets show that AddGraph outperforms other existing anomaly detection methods significantly.

Acknowledgments

This work was partially supported by National Key Research and Development Program No.2016YFB1000700, NSFC under Grant No.61572040 and 61832001, and Alibaba-PKU joint Program.

References

[Aggarwal et al., 2011] Charu C Aggarwal, Yuchen Zhao, and S Yu Philip. Outlier detection

in graph streams. In 2011 IEEE 27th International Conference on Data Engineering, pages 399–409. IEEE, 2011.

[Akoglu et al., 2015] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. Data mining and knowledge discovery, 29(3):626–688, 2015.

[Bordes et al., 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In Advances in neural information processing systems, pages 2787–2795, 2013.

[Chung et al., 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[Cui et al., 2017] Qiang Cui, Shu Wu, Yan Huang, and Liang Wang. A hierarchical contextual attention-based GRU network for sequential recommendation. arXiv preprint arXiv:1711.05114, 2017.

[Defferrard et al., 2016] Michael Defferrard, Xavier Bres- ¨ son, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in neural information processing systems, pages 3844–3852, 2016.

[Eswaran et al., 2018] Dhivya Eswaran, Christos Faloutsos, Sudipto Guha, and Nina Mishra. Spotlight: Detecting anomalies in streaming graphs. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1378–1386. ACM, 2018.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pages 855–864. ACM, 2016.

[Hooi et al., 2016] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 895–904. ACM, 2016.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

[Liu et al., 2017] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. Global context-aware attention lstm networks for 3d action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1647–1656, 2017.

[McConville et al., 2015] Ryan McConville, Weiru Liu, and Paul Miller. Vertex clustering of augmented graph streams. In Proceedings of the 2015 SIAM International Conference on Data

Mining, pages 109–117. SIAM, 2015.

[Ranshous et al., 2015] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F Samatova. Anomaly detection in dynamic networks: a survey. Wiley Interdisciplinary Reviews: Computational Statistics, 7(3):223–247, 2015.

[Ranshous et al., 2016] Stephen Ranshous, Steve Harenberg, Kshitij Sharma, and Nagiza F Samatova. A scalable approach for outlier detection in edge streams using sketch-based approximations. In Proceedings of the 2016 SIAM International Conference on Data Mining, pages 189–197. SIAM, 2016.

[Shin et al., 2016] Kijung Shin, Bryan Hooi, and Christos Faloutsos. M-zoom: Fast dense-block detection in tensors with quality guarantees. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 264–280. Springer, 2016.

[Shin et al., 2017] Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. D-cube: Dense-block detection in terabyte-scale tensors. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pages 681–689. ACM, 2017.

[Sricharan and Das, 2014] Kumar Sricharan and Kamalika Das. Localizing anomalous changes in time-evolving graphs. In Proceedings of the 2014 ACM SIGMOD international conference on Management of data, pages 1347– 1358. ACM, 2014.

[Sun et al., 2006] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 374–383. ACM, 2006.

[Tang et al., 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[Wang et al., 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In Twenty-Eighth AAAI conference on artificial intelligence, 2014.

[Yu et al., 2018] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 2672–2681. ACM, 2018.

[Zhao and Yu, 2013] Yuchen Zhao and Philip S Yu. On graph stream clustering with side information. In Proceedings of the 2013 SIAM International Conference on Data Mining, pages 139–150. SIAM, 2013.