基于多通道注意力图卷积网络的微服务分解

张　攀[1],来风刚[1],周　逸[1],羊麟威[2],钱李烽[2],刘　昕[2],李　静[2]
(1. 国家电网有限公司信息通信分公司,北京 100053;
2. 南京航空航天大学 计算机科学与技术学院/人工智能学院,江苏 南京 211106)

摘　要:为了解决功能、规模和复杂性不断增长的软件系统可能面临的可维护性和可扩展性等一系列软件开发和运维问题,微服务分解成为了目前研究的热点。现有的微服务分解主要是通过微服务的聚类,将单体系统划分为潜在的微服务候选。在微服务的自动化聚类中,基于图卷积网络(Graph Convolutional Network,GCN)的深度学习方法在特征学习方面取得了较好的效果,但是现有模型中缺乏对多通道信息的处理。针对该问题,提出一种基于多通道注意力图卷积网络的微服务分解方法 MAGEMP。该方法使用多通道图注意力网络来学习不同强度的属性图和结构图节点之间的特征嵌入表示,再通过注意力机制获取不同通道嵌入表示的融合信息,最后综合聚类信息的联合学习框架获得高质量的微服务分解。在四个公开数据集上多角度验证该模型的有效性。与同类方法相比,MAGEMP 方法提高了嵌入特征学习能力,在单体程序公开数据集上测试的功能性、模块性等性能方面取得了显著提升。

关键词:微服务架构;微服务分解;图神经网络;多通道;注意力机制
中图分类号:TP311.5　　**文献标识码**:A　　**文章编号**:1673-629X(2023)08-0066-08
doi:10.3969/j.issn.1673-629X.2023.08.010

Multi-channel Attentional Graph Convolutional Networks for Microservice Extraction

ZHANG Pan[1],LAI Feng-gang[1],ZHOU Yi[1],YANG Lin-wei[2],
QIAN Li-feng[2],LIU Xin[2],LI Jing[2]
(1. Information and Communication Branch of State Grid Corporation of China,Beijing 100053,China;
2. School of Computer Science and Technology/School of Artificial Intelligence,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China)

Abstract:In order to solve a series of software development and operation and maintenance problems,such as maintainability and scalability,which may be faced by software systems with increasing functions,scale and complexity,microservice extraction is a hot problem currently. The existing work of microservice extraction is mainly to divide the monolithic program into potential microservice candidates through the clustering of microservices. The graph convolutional network (GCN) for automatic microservice extraction has been obtained potential results,but being lack of taking full advantage of multi-channel information. To solve the above problems,a monolith decomposition method using deep learning clustering based on multi-channel attention map neural network (MAGEMP) is proposed. Multiple independent graph convolutional networks are used to learn different interactions between topological graph and attribute graph nodes of different modal,and then the fusion of different embedding representations is further obtained through the attention mechanism,finally the joint learning framework integrating clustering information obtains high-quality microservice division. The validity of the model is verified from multiple angles on four public data sets. Compared with similar methods,MAGEMP improves the learning ability of embedded features,and significantly improves the performance of testing on the open data set of monomer programs,such as functionality and modularity.

Key words:microservice architecture;microservice decomposition;graph neural network;multi-channel;attention mechanism

收稿日期:2022-08-10　　**修回日期**:2022-12-14
基金项目:国家电网有限公司科技项目(5700-202152169A-0-0-00)
作者简介:张　攀(1989-),男,博士,高级工程师,研究方向为云计算;通信作者:李　静(1976-),女,博士,副教授,CCF 会员(54064M),研究方向为数据挖掘。

0　引　言

伴随云计算技术的快速发展,为了充分利用云基础设施,灵活进行业务扩张和性能伸缩,降低维护成本,Amazon、IBM、Google 等大型公司研究将单体应用软件系统迁移为基于微服务的架构[1-3]。与单体架构将系统各个模块紧耦合不同,微服务架构由许多独立的服务组成,每个服务可独立修改、开发、部署和维护。微服务分解是将现有代码重构为一组较小的独立代码组的过程。目前大多数微服务分解过程是手工完成的,代价昂贵、耗时且易于出错,其质量通常与专家经验和知识密切相关,因此,迫切需要自动化的过程来将单体应用转换成微服务系统,微服务分解已成为软件工程和云计算领域研究的重要问题之一。

近些年,已有许多工作研究单体应用系统的微服务分解方法。Levcovitz 等人[4]于 2016 年提出一种基于人工经验功能分解的微服务分解方法。Chen 等人[5]于 2017 年使用基于数据流进行候选服务分解。Baresi 等人[6]于 2017 年提出了一种基于 OpenApi 规范规定的功能语义相似性的方法。Tyszberowicz 等人[7]于 2018 年提出了一种基于软件需求功能分解微服务的方法。Munezero 等人[8]于 2018 年提出一种使用领域驱动设计(DDD)模式分解微服务的方法。Ding 等人[9]于 2020 年提出了一种场景驱动、自底向上的融合用户反馈的半自动化拆分方法。Zhang 等人[10]于 2020 年提出一种围绕负载均衡作为优化目标的基于运行轨迹的多目标优化的微服务分解方法。Brito 等人[11]于 2021 年提出一种基于主题建模的微服务分解方法。上述的大多数方法需要耗费大量的人力,并且依赖专家知识,微服务分解的效率与质量难以保证。另外,大多数方法存在只适用于一个数据集的情况,在多个数据集上缺乏可扩展性,或者缺少自动化实现的方法。

针对上述问题,受到多通道信息融合模型的启发,该文使用多通道注意力图卷积网络对软件实体进行建模和嵌入特征表示学习,提出基于多通道注意力图卷积网络和 Java 单体程序微服务分解联合学习框架 MAGEMP。

主要贡献表现在:(1)提出一种多通道注意力图卷积网络微服务分解框架,用于建模 Java 源代码中提取的实体类信息的嵌入特征学习任务;(2)构建了基于注意力的多通道图卷积网络特征嵌入融合机制,可以实现多通道信息的有效融合,提升多通道特征表示能力;(3)在 M2M[12]提供的公开通用数据集上进行了广泛的实验,表明多通道深度注意力图神经网络模型在微服务分解方面较同类算法 COGCN[13]有了显著的提升。

1　相关工作

微服务分解的主要标准是每个微服务尽最大可能满足低耦合和高内聚标准[14]。采用聚类的方法进行微服务分解是目前的一个主流趋势,因此,本部分将主要介绍利用聚类算法进行微服务分解的相关工作。

Mitchell 等人于 2006 年提出的 Bunch[15],使用源代码分析工具将源代码转化为一个有向的模块依赖图(MDG),然后将微服务分解建模为 MDG 划分空间的聚类搜索问题。Gysel 等人[16]于 2016 年引入了一个基于 16 个耦合标准的服务分解工具,这些标准来源于行业和文献,通过创建加权图,利用聚类算法分解潜在的微服务。Mazlami 等人于 2017 年提出的 MEM[17]模型使用逻辑耦合、语义耦合及贡献者耦合三种不同的耦合策略,将单体应用程序转换为一个无向加权图并使用基于 KRUSKAL 的最小生成树聚类算法确定可能的微服务。Amiri 等人[18]于 2018 年提出一种使用业务流程模型和符号从业务流程中表示微服务的方法,实现过程中对结构依赖和数据对象依赖信息使用简单的矩阵加法进行了聚合。Jin 等人于 2019 年提出的 FoSCI[19]模型,采用非支配排序遗传算法-II 进行多目标优化,将“功能原子”分组为每个候选服务的类实体,对每个候选服务识别带有操作的接口类,将它们与类实体结合生成一个候选服务。Desai 等人于 2021 年提出的 COGCN[13]方法使用 Soot 工具获得 Java 程序中类和类之间的调用关系图,利用深度图神经网络学习该图的嵌入表示,结合 K-Means 聚类算法实现微服务分解。Kalia 等人于 2021 年提出了 M2M[12]方法。通过运行时追踪数据,构建以类为节点的调用追踪关系。然后通过定义的直接调用关系、间接调用关系、直接调用模式和间接调用模式四种时空特征计算类和类之间的相似性矩阵,最后使用层次聚类方法将应用程序类划分成不相交的微服务。

2　问题描述

MAGEMP 的目的是将 Java 单体应用程序根据代码中类之间的相似性进行划分。在 Java 应用程序中,程序都是由类组成的,类和类之间的调用关系可以表示为类之间的调用图形式,以图的结构来表示。为了充分利用多属性特征以构建更加高效的微服务分解方法,MAGEMP 将该问题建模为一个多通道图神经网络知识表示学习和聚类过程。

2.1　基于多通道图的特征嵌入

通常,图定义为 $G=(V,E)$,$G \in R^{n\times n}$,其中 $V=\{v_1,v_2,\cdots,v_n\}$ 表示节点集合,v_i 表示第 i 个节点,$E=\{e_{ij}\}_{i,j=1}^{n}$ 是边集合,n 表示节点数目。使用多属性特征矩阵:$\boldsymbol{X}_m=\{x_m^1,\cdots,x_m^i,\cdots,x_m^n\}$($\boldsymbol{X}_m \in R^{n\times d_m}$),$m=1,2,$

…,M,对图 G 进行拓展定义。定义多通道属性图 $G^{'}=(V,E,X_m)$,其中 x_m^i 表示与节点 v_i 相关联的特征向量。M 是通道数目(该文取 $M=3$)。d_m 是 m 通道属性维度。图 $G^{'}$ 的结构即邻接矩阵 $\boldsymbol{A}$ 可以通过边 E 得到。如果 $e_{ij}\in E$,则 $A_{ij}=1$,否则,$A_{ij}=0$。给定图 $G^{'}$,学习的任务是得到高质量的特征嵌入表示,能够将每个带有结构和属性的节点 $v_i\in V$ 嵌入到一个 d 维向量 $d\ll|V|$,特征嵌入表示尽可能保持原始图的结构信息 A 和属性信息 X_m。

2.2 多通道图聚类

给定属性图 $G^{'}$,多通道图聚类的目的是在没有标签数据的前提下,使用非监督学习将图中节点分割为 K 个不相交的簇 $\zeta=\{\zeta_1,\zeta_2,\cdots,\zeta_k\}$,分解出 K 个簇,以满足:在同一簇内的节点之间在结构上距离更小且在同一个簇内的节点属性更具相似性。

3 MAGEMP

针对 COGCN 方法没有充分利用多通道信息建模的问题,对 Java 单体程序中获取的类和类之间的多属性关系建模,提出一个集成多通道信息的统一框架 MAGEMP。如图 1 所示,模型框架的主要流程包括:

(1)多通道图构建。将 Java 单体应用中获取的类和类之间的多属性特征矩阵构建成多通道图;

(2)构建基于多通道图卷积神经网络的特征嵌入学习模型,每个通道得到新的特征嵌入表示,每个通道图卷积神经网络由编码器和解码器组成;

(3)基于节点重要性的注意力融合机制将融合得到新的嵌入表示。

通过联合学习框架不断优化参数,最后使用谱聚类算法进行聚类,得到微服务的分解方案。

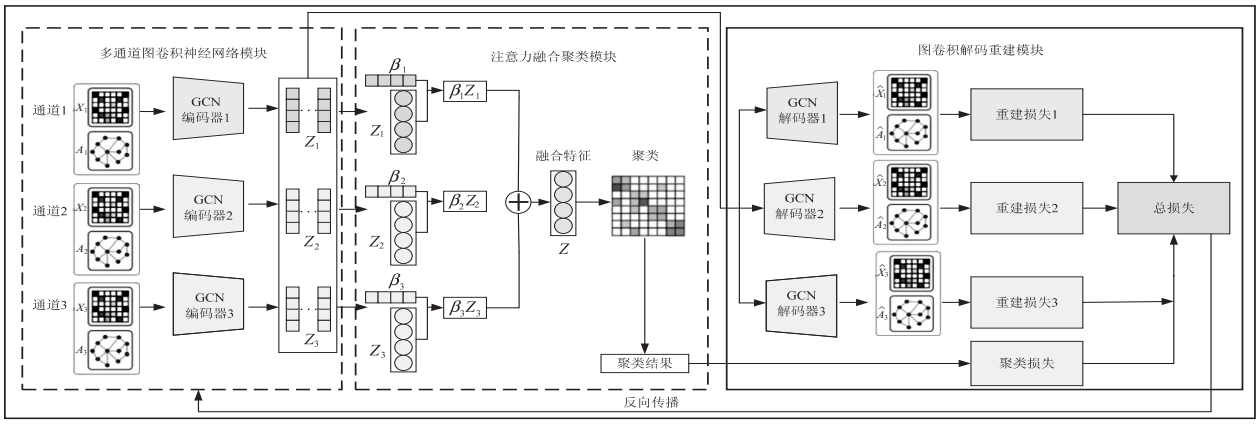

图 1 MAGEMP 框架

3.1 多通道图构建

通过对 Java 应用程序动态分析,可获得运行时追踪数据。一条完整的调用链追踪数据信息 Trace=(序号,业务用例,class_i,调用(或返回到),class_j,调用所在文件和方法)[12]。每个业务用例包含多条调用链,每条调用链是从业务用例测试的入口点(entrypoint)类开始的,并在某个类结束,中间经过多个类。因此根据调用链信息,可获得类与调用链信息,可获得类与调用链的关系矩阵 $\boldsymbol{X}_{\text{class-trace}}^{V\times P}$。

p 是某应用中入口点集合。如果 class_i 出现在入口点 p 所在的执行追踪中,则 $X_{\text{class-trace}}(i,p)=1$,否则 $X_{\text{class-trace}}(i,p)=0$。进一步计算某个入口点开始的调用链同时包含 class_i 和 class_j 的次数,获得属性矩阵 $\boldsymbol{X}_{\text{class-occurence}}$。最后,通过分析类之间的继承关系得到继承关系矩阵 $\boldsymbol{X}_{\text{class-inheri}}$,如果两个类存在继承关系,则 $X_{\text{class-inheri}}(i,j)=X_{\text{class-inheri}}(j,i)=1$,否则,$X_{\text{class-inheri}}(i,j)=X_{\text{class-inheri}}(j,i)=0$。

为了能够捕捉到应用程序中类和类之间的全面深度关系,对三种关系矩阵进行新的组合形成多通道信息数据对,即:

$$\boldsymbol{X}_1=(\boldsymbol{X}_{\text{class-trac}},\boldsymbol{X}_{\text{class-occurence}})$$
$$\boldsymbol{X}_2=(\boldsymbol{X}_{\text{class-trac}},\boldsymbol{X}_{\text{class-inheri}})$$
$$\boldsymbol{X}_3=(\boldsymbol{X}_{\text{class-occurence}},\boldsymbol{X}_{\text{class-inheri}})$$

使用 $\boldsymbol{A}\in R^{|V|\times|V|}$ 表示该图的邻接矩阵,定义为:

$$\boldsymbol{A}_{ij}=\begin{cases}1, & \text{if } (v_i,v_j)\in E\\ 0 & \text{otherwise}\end{cases}$$

因为本问题中类之间的调用图固定,因此三个通道属性图具有相同的邻接矩阵 $\boldsymbol{A}$。下一步,将 $(\boldsymbol{X}_1,\boldsymbol{A})$、$(\boldsymbol{X}_2,\boldsymbol{A})$ 和 $(\boldsymbol{X}_3,\boldsymbol{A})$ 分别输入到多通道注意力图卷积网络编码器模块中。

3.2 多通道图卷积神经网络架构

3.2.1 多通道图卷积神经网络编码器模块

MAGEMP 中,每一个通道都使用了图卷积网络(GCN)[20]。图卷积神经网络从邻居节点聚合信息并学习图中每个节点的嵌入表示:

$$G(X,A)=\sigma(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{X}\boldsymbol{W}) \tag{1}$$

其中,$\tilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$ 且 $\tilde{D}_{ii}=\sum_j\tilde{A}_{ij}$。$\boldsymbol{A}$ 是邻接矩阵,$\boldsymbol{I}$ 是单位矩阵,$\boldsymbol{W}$ 是可训练的权值矩阵,$\sigma(\cdot)$ 是激活函数,

例如 ReLU(·) = max(0, ·) 。该文 GCN 网络中使用 2 个图卷积层。

MAGEMP 对多通道属性图构建了三个图卷积编码器,通过三个通道 GCN 编码器将多通道信息数据对映射到新的特征嵌入空间中。具体来说,对于 m (m = 1,2,M)通道,则 GCN 模型的函数为 $f_m(G,X_m;\theta) \rightarrow Z_m$ 。它利用多层图卷积编码器将 X_m 映射成 d 维嵌入特征 Z_m ,其中 θ 是多通道图编码器参数,具体来说是各层编码器的权值矩阵 $\boldsymbol{W}$ 。第 m 个图卷积编码器的第 l 层的输出 Z_m^l 为:

$$Z_m^{(l)} = \sigma(\tilde{\boldsymbol{D}}_m^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}_m^{-\frac{1}{2}}Z_m^{(l-1)}\boldsymbol{W}_m) \tag{2}$$

其中, $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}_N$ 是具有自连接的邻接矩阵, $\boldsymbol{I}_N$ 是单位矩阵, $D_{ii} = \sum_j A_{ij}$, W_m^l 是 m 通道的权值参数, σ 表示非线性激活函数。对于 $Z_m^{(l)}$,当 $l = 0$ 时, $Z_m^{(0)} = X_m$,当 $l = L$ 时, $H_m^{(l)}$ 是第 m 个通道的特征嵌入表示。如选取 2 层 GCN 模型,可表示为:

$$Z_m = f(X_m, A) = \sigma(\hat{A}\mathrm{ReLU}(\hat{\boldsymbol{A}}\boldsymbol{X}_m\boldsymbol{W}_m^0)\boldsymbol{W}_m^1) \tag{3}$$

其中, $\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}$, $\boldsymbol{W}_m^{(0)} \in R^{d\times H}$ 是第一个图卷积编码器的权值矩阵, $\boldsymbol{W}_m^{(1)} \in R^{H\times F}$ 是第二个图卷积编码器的权值矩阵。 $X_m \in R^{N\times d}$, N 是节点个数, d 是属性特征维度。 σ 是激活函数,由此得到不同通道嵌入表示 Z_m 。

3.2.2　多通道图卷积神经网络解码器模块

为了能够提高每个通道 GCN 编码器特征表示的有效性,增加了 GCN 解码器模块[20]来重建多通道信息数据对 $\boldsymbol{X}_m$ 和图的邻接矩阵 $\boldsymbol{A}_m$ 。经过多通道 GCN 编码器模块后,获得特征嵌入表示 Z_m 。在每个解码器端假定 $\hat{\boldsymbol{X}}_m$ 表示 m 通道的解码器重建的信息数据对, $\hat{\boldsymbol{A}}_m$ 代表 m 通道的解码器重建的邻接矩阵。第 m 通道第 l 层的属性特征重建可表示为:

$$\hat{Z}_m^l = \sigma(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\bar{\boldsymbol{A}}\hat{D}^{-\frac{1}{2}}\hat{Z}_m^{l-1}\hat{\boldsymbol{W}}^{l-1}) \tag{4}$$

其中, $\bar{\boldsymbol{A}} = 2\boldsymbol{I}_n - \boldsymbol{A}$, $\hat{\boldsymbol{D}} = 2\boldsymbol{I}_n + \boldsymbol{D}$,经过 L 层解码器,重构后的多通道信息数据对为 $\hat{\boldsymbol{X}}_m = H_m^0$ 。对于第 m 通道邻接矩阵的重建过程如下:

$$\hat{\boldsymbol{A}}_m = \sigma(Z_m Z_m^T) \tag{5}$$

σ 是激活函数。使用重建的信息数据对和邻接矩阵和原始的信息数据对和邻接矩阵的误差优化网络,定义为:

$$L_{\mathrm{sa}} = \min_{\theta_i}\sum_{m=1}^{M} \| \boldsymbol{X}_m - \hat{\boldsymbol{X}}_m \|_F^2 + \lambda\sum_{m=1}^{M} \| \boldsymbol{A} - \hat{\boldsymbol{A}}_m \|_F^2 \tag{6}$$

3.3　注意力融合机制和聚类模块

3.3.1　注意力融合机制

使用注意力机制[21]融合不同通道 GCN 的特征嵌入表示 Z_m 得到新的特征嵌入表示 Z 。以节点 i 为例,其在 m 通道的嵌入表示为 $z_m^i \in R^{1\times h}$, $m = 1, 2, \cdots, M$ 。注意力机制先对每个通道嵌入表示进行非线性变换,再用一个共享注意力向量 $p_m \in R^{h'\times 1}$ 得到 m 通道节点 i 注意力值 w_m^i :

$$w_m^i = \boldsymbol{p}_m^{\mathrm{T}} \cdot \tanh(\boldsymbol{W}_m \cdot (z_m^i)^{\mathrm{T}} + \boldsymbol{b}_m) \tag{7}$$

其中, $\boldsymbol{W}_m \in R^{h'\times h}$ 是权值矩阵, $\boldsymbol{b}_m \in R^{h'\times 1}$ 是偏置向量。对注意力值 w_m^i 进行归一化:

$$\gamma_m^i = \mathrm{softmax}(W_m^i) = \frac{\exp(w_m^i)}{\sum_{m=1}^{M}\exp(w_m^i)} \tag{8}$$

较大 γ_m^i 值表示 m 通道节点 i 的嵌入特征越重要。类似地,对于所有的节点,可获得学习后的权值 $\gamma_m = [\gamma_m^i] \in R^{n\times 1}$,进一步地, $\beta_m = \mathrm{diag}[\gamma_m]$, β_m 描述了同一节点在不同通道下的重要性。将三个通道嵌入表示 Z_m 进行融合可获得最终的特征嵌入表示 Z:

$$Z = \sum_{m=1}^{M}\beta_m \cdot Z_m \tag{9}$$

3.3.2　聚类模块

经过多通道图卷积神经网络计算和注意力机制融合后,得到了新的特征嵌入表示 Z 。谱聚类是基于图论的一种典型聚类算法,减少了对样本空间形状的要求,可以处理非凸数据集,同时能有效克服集中经典聚类算法收敛于局部最优的缺点,因此利用谱聚类算法[22]对嵌入表示 Z 进行聚类[23-24]。为了进行谱聚类,使用高斯核定义仿射矩阵 $\boldsymbol{S}$,如果 $i \neq j$,则 $S_{ij} = \exp(\frac{\| z_i - z_j \|_2^2}{\sigma})$, $i, j = 1, 2, \cdots, n$;否则 $S_{ij} = 0$ 。其中 $\| \cdot \|$ 是向量 z_i 和 z_j 之间的欧几里得距离, σ 是尺度参数, S_{ij} 对称且非负。然后,利用谱聚类算法将 N 节点聚类到 K 个不同划分中。

为了能够获得更具有判别性的嵌入表示和最优聚类结果,完成谱聚类后,构建簇的 k-d 树,借助 k-d 树实现最近邻的快速查找,找到与节点 z_i 同一个簇中曼哈顿距离最近的邻居节点 z_j 。定义损失函数[22]:

$$L_{\mathrm{clus}} = \sum_i \min_j d(z_i, z_j) \quad i, j = 1, 2, \cdots, n \tag{10}$$

其中, $d(z_i, z_j)$ 表示节点 i 和节点 j 之间的曼哈顿距离 $| z_i - z_j |$ 。

3.4　损失函数

根据 MAGEMP 网络模型结构,其损失函数由

多通道图卷积网络的重建损失和聚类损失组成。定义为：

$$L = \sum_{i=1}^{M} L_{\text{sa}}^{i} + \gamma L_{\text{clus}} \tag{11}$$

其中，$\gamma > 0$，是超参数。通过最小化该损失函数训练整个多通道注意力图卷积神经网络的参数，得到较高质量嵌入质量表示和聚类效果。

3.5 训练过程

实验过程由三个步骤组成。第一步是对多通道图卷积神经网络模块网络参数进行预训练；第二步是对整个网络进行正式训练，得到最终的参数；第三步是使用学习到的特征嵌入表示 Z 进行聚类。

(1)预训练优化网络参数。构建图邻接矩阵 $\boldsymbol{A}$，多通道信息数据对 $\boldsymbol{X}_m$ 输入到 m 个 GCN 编码器中获得嵌入表示 Z_m。多通道 GCN 解码器获得多通道信息数据对重建 $\hat{\boldsymbol{X}}_m$ 和邻接矩阵重建 $\hat{\boldsymbol{A}}_m$，使用重建误差优化每个通道 GCN 网络参数。将训练得到的 GCN 编码器的参数作为网络的初始参数，计算注意力融合特征嵌入表示 Z，使用谱聚类方法初始化聚类 $(u_1, u_2, \cdots, u_k)$，完成初始化。

(2)正式开始训练过程。对 GCN 编码器、解码器和注意力层参数进行迭代优化，得到最终的参数。首先将多通道信息数据对和邻接矩阵输入到预训练好的 GCN 编码器网络。经过模型训练，获得隐含层嵌入表示和注意力层嵌入表示，使用注意力层嵌入表示数据谱聚类，得到划分结果，计算总体损失函数。通过最小化总体损失函数反向传播训练整体网络。

(3)通过学习到的嵌入表示 $\boldsymbol{Z}_m$ 计算仿射矩阵 $\boldsymbol{S}$，进行谱聚类获得最终聚类结果。MAGEMP 具体如算法 1 所示。

算法 1：MAGEMP 训练算法。

输入：多属性特征图 $G' = (V, E, X_m)$，簇的个数 K，迭代次数 N。

输出：聚类结果 R。

/* 预训练阶段 */

(1) $\tilde{\boldsymbol{A}} \leftarrow \boldsymbol{A} + \boldsymbol{I}_N$；$\tilde{D}_{ii} \leftarrow \sum_j \tilde{A}_{ij}$

(2)利用公式(3)～公式(6)对多通道 GCN 编码器进行预训练，得到初始参数值；

(3)利用公式(3)计算每个通道图卷积编码器嵌入特征表示，利用公式(7)～公式(9)使用注意力机制得到融合的嵌入表示；

(4)对步骤 3 中经过融合得到的嵌入表示使用 k-means 算法初始化聚类；

/* 在线训练阶段

(5)For iter $\in 0, 1, \cdots, N$

(6)使用 m 通道 GCN 编码器计算 Z_m；

(7)使用注意力机制融合嵌入表示；

(8)使用谱聚类计算嵌入表示 Z 聚类结果；

(9)利用公式(11)计算总损失函数；

(10)利用 Adam 算法更新神经网络参数；

(11)End for

(12)根据嵌入表示计算仿射矩阵作为谱聚类的权值进行聚类；

(13)计算聚类结果 R；

(14)返回 R。

4 实 验

4.1 数据集

为了评估该方法，使用 Mono2Mirco 提供的 4 个公开数据集[15]进行测试。acmeair 是航空公司订票应用系统，daytrader 是在线股票交易系统，jpetstore 是宠物商店系统，plants 是植物商店系统。

4.2 评价指标

为了较好地量化评估 MAGEMP 模型微服务分解的性能，采用了微服务分解中较通用的满足上述要求的评价指标。评价指标主要有 3 个：

(1)IFN(independence of functionality)[19]评价微服务是否具有良好定义和独立性，可表示为：

$$\text{IFN} = \frac{1}{N}\sum_{i=1}^{N} \text{ifn}_i,\ \text{ifn}_i = |\ I_i\ | \tag{12}$$

通常 IFN 值越小，表示微服务承担责任越单一。

(2)SM(structural modularity quality)[19]用于量化微服务内的类间结构一致性，表示为：

$$\text{SM} = \frac{1}{M}\sum_{i=1}^{M} \text{scoh}_i - \frac{1}{(M(M-1))/2}\sum_{i\neq j}^{M} \text{scop}_{i,j} \tag{13}$$

其中，$m_i \in M$，scoh 和 scop 量化服务之间的耦合。$\text{scoh}_i = \frac{u_i}{m_i^2}$，$\text{scop}_{i,j} = \frac{\sigma_{i,j}}{2 * (m_i * m_j)}$。$u_i$ 表示当 M 个实体类之间发生结构调用依赖时一个服务内部边的数目，$\sigma_{i,j}$ 指的是服务之间的依赖发生时边的数目。

(3)ICP(inter-partition call percentage)[25]用于评估微服务之间的交互百分比。可表示为：

$$\text{icp}_{i,j} = \frac{c_{i,j}}{\sum_{i=1,j=1,i\neq j}^{M} c_{i,j}} \tag{14}$$

其中，$c_{i,j}$ 表示不同微服务之间的调用数目。ICP 值越小，说明服务之间调用纯度越好。

4.3 基线设置

为了评价所提方法的性能，将其与目前最先进的方法进行比较。第 1 个基准方法是 Bunch[15]，它是较早提出的将服务划分问题描述为一个搜索问题的方法，利用爬山搜索技术找到一个能够最大化两个特定

度量的划分(微服务),该划分可以最大化结构模块化质量。MEM[17]是基于逻辑、语义和变更历史的耦合构造一个图,将微服务分解建模为图割问题,然后使用最小生成树从构造的图中寻找最小割划分。每个 min-cut 中的类代表分解出的微服务。FoSCI[19]使用运行时追踪数据作为数据源,利用跟踪规约算法删除冗余的跟踪,然后通过层次聚类来创建“功能原子”,再使用多目标优化算法对这些功能原子进行合并。M2M[12]通过不同的业务用例收集运行时追踪数据[26],使用层次聚类方法将单体应用划分为不同微服务,并公开了相应的数据集。COGCN 是第一个使用图神经网络进行微服务分解的深度学习方法,并利用多个损失函数构成了一个端到端的学习框架,再使用 K-Means 聚类方法分解微服务,也是主要对比的同类方法。

4.4　性能比较

为了验证模型对于微服务分解的效果,分别在四个公开数据集上进行了测试,结果如表 1 ~ 表 4 所示。性能测试的结果是在参数设置的范围内运行 100 次,然后取每个性能指标的中位数[12-13]。其中 ↑ 表示该指标值越大越好,↓ 表示该指标值越小越好。表中粗体表示性能最好。在所有数据集上,MAGEMP 取得了显著的效果。

表 1　daytrader 数据集上的结果

指标	M2M	FoSCI	MEM	Bunch	COGCN	MAGEMP
ICP ↓	0.346	0.748	0.355	0.572	0.455	**0.284**
SM ↑	0.078	0.092	0.089	**0.269**	0.086	0.122
IFN ↓	**1.922**	3.489	4.200	-	2.880	4.113

表 2　acmeair 数据集上的结果

指标	M2M	FoSCI	MEM	Bunch	COGCN	MAGEMP
ICP ↓	0.527	0.706	0.589	0.55	0.444	**0.301**
SM ↑	0.072	0.095	0.097	**0.177**	0.038	0.062
IFN ↓	3.375	4.375	4.333	3.875	2.846	**2.805**

表 3　jpetstore 数据集上的结果

指标	M2M	FoSCI	MEM	Bunch	COGCN	MAGEMP
ICP ↓	0.333	0.478	0.434	0.477	0.582	**0.263**
SM ↑	0.054	0.044	**0.124**	-	0.091	0.116
IFN ↓	**1.857**	3.750	3.429	7.948	2.533	2.148

表 4　plants 数据集上的结果

指标	M2M	FoSCI	MEM	Bunch	COGCN	MAGEMP
ICP ↓	0.381	0.682	**0.320**	0.501	0.571	0.608
SM ↑	0.078	0.135	**0.210**	0.155	0.133	0.138
IFN ↓	6.000	4.875	4.750	6.357	4.875	**4.576**

与同类方法 COGCN 相比,在 daytrader 数据集上,ICP 提升了 37.6%,SM 提升了 41.86%。在 acmeair 数据集上,ICP 提升了 32.21%,SM 提升了 63.16%,IFN 提升了 1.44%。在 jpetstore 数据集上,ICP 提升了 54.81%,SM 提升了 27.47%,IFN 提升了 15.2%。在 plant 数据集上,ICP 提升了 6.48%。MAGEMP 的多通道信息的充分利用和有效融合显著提升了微服务分解在功能性和模块性上的性能。

与另外四种非深度学习模型比较,MAGEMP 在 daytrader 数据集的 ICP 指标,在 acmeair 数据集上的 ICP 指标、IFN 指标,在 jpetstore 数据集上的 ICP 指标,在 plants 数据集上的 IFN 指标都取得了最好的实验效果,而在一些其他的指标上,例如 acmeair 数据集上的 IFN 和 plants 上的 ICP 效果并不理想。这很大一部分原因是 MAGEMP 中类调用图的节点属性过于简单,从而无法挖掘更深层次的信息。

从整体实验效果上来看,MAGEMP 具有一定的优势,但也有可提升的空间,并且进一步验证了图神经网络在微服务分解问题上的可行性。

4.5　消融分析

为了进一步评估 MAGEMP 模型中不同模块的贡献,对 MAGEMP 进行了消融实验,考虑了 MAGEMP 的几种变体:

(1)MAGEMP(mean):无注意力融合机制的 MAGEMP。

(2)MAGEMP(m1):只使用通道 1 的 MAGEMP。

(3)MAGEMP(m2):只使用通道 2 的 MAGEMP。

(4)MAGEMP(m3):只是用通道 3 的 MAGEMP。

表 5　daytrader 数据集上消融分析

模型	ICP ↓	SM ↑	IFN ↓
MAGEMP	0.284	0.124	4.113
MAGEMP(mean)	0.313	0.114	4.164
MAGEMP(m1)	0.384	0.123	3.445
MAGEMP(m2)	0.443	0.186	6.56
MAGEMP(m3)	0.202	0.102	3.723

MAGEMP 在不同数据集上消融实验结果如表 5 ~ 表 8 所示。从实验结果中可以看到,在所有数据集和指标上,带有注意力融合机制的 MAGEMP 模型性能更好,表明 MAGEMP 中的注意力机制可以利用多通道的嵌入特征表示的不同贡献,进一步提高聚类(微

服务分解)的性能。从表中还可以看出,MAGEMP 的性能优于单通道图卷积模型,充分验证了多通道图卷积网络模型可以利用多种信息更好地达到微服务分解的目的。

表 6 acmeair 数据集上消融分析

模型	ICP ↓	SM ↑	IFN ↓
MAGEMP	0.301	0.062	2.805
MAGEMP(mean)	0.322	0.057	2.934
MAGEMP(m1)	0.358	0.055	3.600
MAGEMP(m2)	0.483	0.083	4.800
MAGEMP(m3)	0.335	0.045	3.002

表 7 jpetstore 数据集上消融分析

模型	ICP ↓	SM ↑	IFN ↓
MAGEMP	0.263	0.116	2.148
MAGEMP(mean)	0.336	0.108	2.786
MAGEMP(m1)	0.299	0.108	2.286
MAGEMP(m2)	0.425	0.114	3.714
MAGEMP(m3)	0.356	0.109	2.857

表 8 plants 数据集上消融分析

模型	ICP ↓	SM ↑	IFN ↓
MAGEMP	0.608	0.138	4.576
MAGEMP(mean)	0.623	0.130	4.644
MAGEMP(m1)	0.646	0.136	4.857
MAGEMP(m2)	0.704	0.109	5.214
MAGEMP(m3)	0.615	0.147	4.714

4.6 注意力系数分析

注意力融合机制是 MAGEMP 模型的一个关键组成部分,它可以衡量和量化不同通道的重要性和不同通道中不同节点的重要性。图 2 可视化了在 acmeair 数据集上三个通道中每个图节点的注意力系数。从图 2 可以看出,三个通道具有不同的重要性,在不同数据集上,通道 1 和通道 3 的权值比通道 2 的权值高。实验结果表明 MAGEMP 可以很好地利用不同通道的不同特征进行聚类。

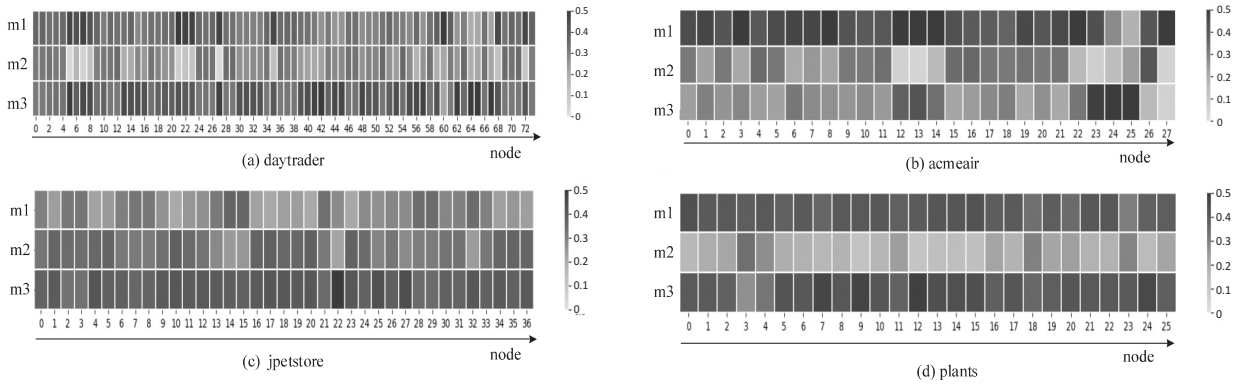

图 2 acmeair 数据集上三种模态的图节点注意力系数热图

5 结束语

受多通道信息处理模型启发,针对 Java 单体应用程序微服务分解问题,对面向对象程序设计中的类实体关系特征表示进行建模,提出了一种基于多通道注意力深度图卷积网络聚类的微服务分解方法 MAGEMP。针对 Java 单体应用程序的特点,类和类之间的关系除了调用关系外,还可以充分利用源代码中的主题建模等信息,因此将会继续研究更加有效的多通道信息,进一步提升微服务分解模型的泛化能力。

深度学习方法具有很好的扩展性,但是在运行时间和效率上,与其他类方法相比,仍然有很大的提升空间,如何改进深度图卷积网络的性能,获得特征表示和运行时的高效,也是一个很好的研究方向。另外,如何在微服务分解领域拓展深度卷积网络的研究方法,提出新型的图卷积神经网络模型,也是值得深入研究的方向。

参考文献:

[1] DI FRANCESCO P, LAGO P, MALAVOLTA I. Migrating towards microservice architectures: an industrial survey [C]//Proc of 2018 IEEE international conference on software architecture. Seattle: IEEE, 2018: 29-2909.

[2] MATHAI A, BANDYOPADHYAY S, DESAI U, et al. Monolith to microservices: representing application software through heterogeneous GNN[J]. arXiv: 2112.01317, 2021.

[3] SELLAMI K, OUNI A, SAIED M A, et al. Improving microservices extraction using evolutionary search[J]. Information and Software Technology, 2022, 151: 106996.

[4] LEVCOVITZ A, TERRA R, VALENTE M T. Towards a technique for extracting microservices from monolithic enterprise systems[J]. arXiv: 1605.03175, 2016.

[5] CHEN R, LI S, LI Z. From monolith to microservices: a dataflow-driven approach[C]//Proc of the Asia-Pacific software engineering conf. (APSEC). Nanjing: IEEE, 2017: 466-475.

[6] BARESI L, GARRIGA M, RENZIS A D. Microservices identification through interface analysis[C]//Proc of European conference on service-oriented and cloud computing. London: Springer, 2017: 19-33.

[7] TYSZBEROWICZ S, HEINRICH R, LIU B, et al. Identifying microservices using functional decomposition[C]//Proc of international symposium on dependable software engineering: theories, tools, and applications. London: Springer, 2018: 50-65.

[8] MUNEZERO I J, MUKASA D T, KANAGWA B, et al. Partitioning microservices: a domain engineering approach [C]//Proc of 2018 IEEE/ACM symposium on software engineering in africa (SEiA). Gothenburg: IEEE, 2018: 43-49.

[9] DING D, PENG X, GUO X F, et al. Scenario-driven and bottom-up microservice decomposition for monolithic systems[J]. Ruan Jian Xue Bao/Journal of Software, 2020, 31 (11): 3461-3480.

[10] ZHANG Y, LIU B, DAI L, et al. Automated microservice identification in legacy systems with functional and non-functional metrics[C]//Proc of the 2020 IEEE Int'l conf. on software architecture (ICSA 2020). Salvador: IEEE, 2020: 135-145.

[11] BRITO M, CUNHA J, SARAIVA J. Identification of microservices from monolithic applications through topic modelling[C]//Proc of the 36th annual ACM symposium on applied computing. New York: ACM, 2021: 1409-1418.

[12] KALIA A K, XIAO J, KRISHNA R, et al. Mono2Micro: a practical and effective tool for decomposing monolithic Java applications to microservices[C]//Pro of the 29th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering. New York: ACM, 2021: 1214-1224.

[13] DESAI U, BANDYOPADHYAY S, TAMILSELVAM S. Graph neural network to dilute outliers for refactoring monolith application[C]//Proceedings of the AAAI conference on artificial intelligence. Menlo Park: AAAI, 2021: 72-80.

[14] 晋武侠,钟定洪,张宇云,等. 基于多源特征空间的微服务可维护性评估[J]. 软件学报, 2021, 32(5): 1322-1340.

[15] MITCHELL B, MANCORIDIS S. On the automatic modularization of software systems using the bunch tool[J]. IEEE Transactions on Software Engineering, 2006, 32(3): 193-208.

[16] GYSEL M, KÖLBENER L, GIERSCHE W, et al. Service cutter: a systematic approach to service decomposition[C]// Proc of the European conf. on service-oriented and cloud computing. Berlin: Springer-Verlag, 2016: 185-200.

[17] MAZLAMI G, CITO J, LEITNER P. Extraction of microservices from monolithic software architectures[C]//2017 IEEE international conference on web services (ICWS). Honolulu: IEEE, 2017: 524-531.

[18] AMIRI M J. Object-aware identification of microservices [C]//Proc of 2018 IEEE international conference on services computing (SCC). San Francisco: IEEE, 2018: 253-256.

[19] JIN W, LIU T, CAI Y, et al. Service candidate identification from monolithic systems based on execution traces[J]. IEEE Trans. on Software Engineering, 2019, 47(5): 987-1007.

[20] PARK J, LEE M, CHANG H J, et al. Symmetric graph convolutional autoencoder for unsupervised graph representation learning[C]//Proc of the IEEE/CVF international conference on computer vision. Seoul: IEEE, 2019: 6519-6528.

[21] XIE Y, ZHANG Y, GONG M, et al. Mgat: multi-view graph attention networks[J]. Neural Networks, 2020, 132(12): 180-189.

[22] YEDIDA R, KRISHNA R, KALIA A, et al. Partitioning cloud-based microservices (via deep learning)[J]. arXiv: 2109.14569, 2021.

[23] YAN D, HUANG L, JORDAN M I. Fast approximate spectral clustering[C]//Proc of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM, 2009: 907-916.

[24] ZHANG C, CHEN H, WU C, et al. Multiscale fast spectral clustering based on kd tree[C]//2019 IEEE 3rd information technology, networking, electronic and automation control conference (ITNEC). Piscataway: IEEE, 2019: 1607-1611.

[25] FRITZSCH J, BOGNER J, ZIMMERMANN A, et al. From monolith to microservices: a classification of refactoring approaches[C]//Proc of international workshop on software engineering aspects of continuous development and new paradigms of software production and deployment. New York: Springer, 2018: 128-141.

[26] KALIA A K, XIAO J, LIN C, et al. Mono2micro: an ai-based toolchain for evolving monolithic enterprise applications to a microservice architecture[C]//Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering. New York: ACM, 2020: 1606-1610.