ICS 35.040

CCS L 71

信息技术 神经网络表示与模型压缩 第3部分：图神经网络

Information technology - Neural network representation and model compression —

Part 3: Graph Neural Network

|  |
| --- |
| （征求意见稿） |
| 在提交反馈意见时，请将您知道的相关专利连同支持性文件一并附上 |

团体标准

T/AI 115.3—XXXX

|  |
| --- |
|  |

xxxx - xx - xx发布

xxxx – xx – xx实施

**中关村视听产业技术创新联盟** 发布

目 次

[前言 II](#_Toc172275511)

[引言 III](#_Toc172275512)

[1 范围 1](#_Toc172275514)

[2 规范性引用文件 1](#_Toc172275515)

[3 术语和定义 1](#_Toc172275516)

[4 缩略语 3](#_Toc172275517)

[5 图神经网络表示与模型压缩概述 4](#_Toc172275518)

[6 图数据表示 5](#_Toc172275519)

[6.1 基本定义 5](#_Toc172275520)

[6.2 图数据类型定义 10](#_Toc172275521)

[6.3 图基本运算 12](#_Toc172275522)

[6.4 图基本任务 28](#_Toc172275523)

[7 图神经网络模型 29](#_Toc172275524)

[7.1 模型结构 29](#_Toc172275525)

[7.2 基础算子 34](#_Toc172275526)

[7.3 点级模型 94](#_Toc172275527)

[7.4 边级模型 121](#_Toc172275528)

[7.5 图级模型 130](#_Toc172275529)

[8 图神经网络压缩 137](#_Toc172275530)

[8.1 图数据压缩 137](#_Toc172275531)

[8.2 模型量化与剪枝 148](#_Toc172275532)

[8.3 模型蒸馏 151](#_Toc172275533)

[8.4 模型加速 157](#_Toc172275534)

[9 图神经网络计算框架 164](#_Toc172275535)

[9.1 基于深度学习平台的图神经网络计算框架 164](#_Toc172275536)

[9.2 图神经网络计算框架与第三方数据源接口 175](#_Toc172275537)

前言

本文件按照GB/T 1.1—2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

本文件是T/AI 115《信息技术 神经网络表示与模型压缩》的第2部分。T/AI 115已经发布了以下部分：

——第1部分：卷积神经网络。

本文件由新一代人工智能产业技术创新战略联盟AI标准工作组提出。

本文件由中关村视听产业技术创新联盟归口。

本文件起草单位：北京大学、北京邮电大学、鹏城实验室、中国电子技术标准化研究院、蚂蚁金服（杭州）网络技术有限公司、华为技术有限公司、阿里巴巴（中国）有限公司、北京百度网讯科技有限公司、西安交通大学、吉林大学、上海中亦图灵数字科技有限公司、中国人民大学、北京海致星图科技有限公司、海信集团有限公司、杭州海康威视数字技术股份有限公司、北京图数创智科技有限公司、中国科学院自动化研究所、上海上湖信息技术有限公司、中科南京人工智能创新研究院、中移动信息技术有限公司、杭州海康威视有限公司、清华大学、深圳市腾讯计算机系统有限公司、浙江大学、北京交通大学、东北大学、南京人工智能芯片创新研究院。

本文件主要起草人：XXXX。

引言

图神经网络已经在推荐系统、知识图谱、智能交通等领域得到广泛应用。基于图神经网络的深度学习算法及其衍生应用系统在许多图任务上取得了突破性的进展，但是其具体的实现仍然依赖于不同的算法平台。由于目前没有统一的图神经网络的表示标准，不同的算法平台采用了不同的图神经网络的表示和存储标准；不同图神经网络框架之间不能互操作与协同工作；图数据和图神经网络模型不能在不同平台上直接迁移使用；不同图神经网络框架对于图神经网络粒度定义不同，妨碍了硬件厂商对于图神经网络的加速和优化；对于新出现的算子，框架都需要进行重定义。本文件旨在提供图数据和图神经网络统一的表示，以及模型算子接口的统一规范参考，提升用户对图神经网络模型的复用效果。对于本标准规定的表示方法不要求平台原生支持，可以通过转换、工具包等形式进行支持。本文件的定义可转化为与特定计算设备、框架匹配的形式和实现。T/AI 115旨在确立适用于不同种类神经网络的表示方法与模型压缩的规范，拟由三个部分组成：

——第1部分：卷积神经网络。目的在于确立适用千卷积神经网络的表示与模型压缩标准。

——第2部分：大规模预训练模型。目的在于确立适应多种推理平台和计算要求的大规模预训练模型的基本表示方法与加速压缩过程。

——第3部分：图神经网络。目的在于确立适应多种计算要求的高效图神经网络模型的基本表示方法与压缩加速过程。

本文件的发布机构提请注意，声明符合本文件时，可能涉及到7.2.1.5与《一种基于移位图卷积神经网络骨骼点行为识别系统及其识别方法》（专利号：202010419839.4）、《一种自适应时序移位神经网络时序行为识别方法》（专利号：202010419814.4）、《异质图神经网络生成方法、装置、电子设备及存储介质》（专利号：201910349275.9）；7.3.2与《一种用于短文本的分类方法及装置》（专利号：201910945503.9）、《基于多通路图卷积神经网络的对象分类方法及装置》（专利号：202010555093.X）、《一种基于图卷积网络模型的分类方法及装置》（专利号：202011604370.8）；7.3.3与《一种信息预测方法、装置、存储介质及计算机设备》（专利号：202110552598.5）、《一种套现用户检测方法、装置及设备》（专利号：201910052269.7）、《一种基于异质图神经网络的节点处理方法、装置及设备》（专利号：202010750181.5）、《基于统一优化目标框架图神经网络的数据分类方法及装置》《专利号：202110023447.0》；7.4.2与《一种基于异质信息网络表示的推荐方法及装置》（专利号：201711239629.1）、《一种基于元路径引导嵌入的查询推荐方法及装置》（专利号：201910342766.0）；7.4.4与《图节点关系表征生成和图节点业务关系预测方法及装置》（专利号：202111003074.7）；9.2.2与《一种图数据采样方法和系统》（专利号：202011038681.2）相关的专利的使用。

本文件的发布机构对于该专利的真实性、有效性和范围无任何立场。

该专利持有人已向本文件的发布机构承诺，他愿意同任何申请人在合理且无歧视的条款和条件下，就专利授权许可进行谈判。该专利持有人的声明已在本文件的发布机构备案，相关信息可以通过以下联系方式获得：

专利持有人：北京大学、北京邮电大学、蚂蚁集团、深圳市腾讯计算机系统有限公司、中国科学院自动化研究所、中科南京人工智能创新研究院

地址：北京市海淀区颐和园路5号、北京市海淀区西土城路10号、浙江省杭州市西湖区西溪路569号蚂蚁A空间、广东省深圳市南山区腾讯滨海大厦、北京市海淀区中关村东路 95 号、江苏省南京市创研路266号麒麟人工智能产业园3号楼3楼

联 系 人：黄铁军

通讯地址：北京大学理科2号楼2641室

邮政编码：100871

电子邮件：tjhuang@pku.edu.cn

电话：+8610-62756172

传真：+8610-62751638

网址：http://www.aitisa.org.cn

请注意除上述专利外，本文件的某些内容仍可能涉及专利。本文件的发布机构不承担识别这些专利的责任。

信息技术 神经网络表示与模型压缩 第3部分：图神经网络

1. 范围

本文件规定了适应多种计算要求的高效图神经网络模型的基本表示方法与压缩加速过程。

本文件适用于各种图神经网络模型的研制、开发、测试评估过程，以及在端云领域的高效应用。

1. 对于本文件规定的表示与模型压缩方法不要求机器学习框架原生支持，可以通过转换、工具包等形式支持。
2. 规范性引用文件

下列文件中的内容通过文中的规范性引用而构成本文件必不可少的条款。其中，注日期的引用文件，仅注日期对应的版本适用于本文件；不注日期的引用文件，其最新版本（包括所有的修改单）适用于本文件。

GB/T 5271.34-2006　信息技术 词汇 第34部分：人工智能 神经网络

GB/T 42382.1-2023　信息技术 神经网络表示与模型压缩 第1部分：卷积神经网络

1. 术语和定义

下列术语和定义适用于本文件。



图神经网络　**graph neural network**

用于处理由图数据结构表示数据的神经网络。

是人工智能应用中代表性的深度学习算法。

模型　**model**

对应完成单个或多个任务的神经网络。

层　**layer**

神经网络中的分级结构。

每个网络层包含多个算子，例如输入层，卷积层，全连接层。

[来源：GB/T 42382.1-2023]

权重　**weight**

不同单元之间的连接强度。

权重是神经网络的固有参数之一。

[来源：GB/T 42382.1-2023]



图　**graph**

一个三元组，包含节点集、边集、关联函数。

节点　**node**

图结构数据中的一个实体或对象，可以包含数据，例如标签、权重或其他属性。

边　**edge**

图结构数据中连接两个节点的线段，代表节点之间的关系或连接。

特征　**feature**

数据集、神经网络计算过程中，数据的属性向量。

图基本运算　**graph fundamental operation**

对图结构信息的基本运算，例如求解节点的度等。

图基本任务　**graph fundamental task**

基于图数据实现某种功能，例如节点分类、链路预测、图分类任务等。

基础算子　**basic operator**

应用在图神经网络模型上的基本运算操作定义。

点级模型　**node-level task model**

应用于节点级任务的图神经网络模型。

边级模型　**edge-level task model**

应用于边级任务的图神经网络模型。

图级模型　**graph-level task model**

应用于图级任务的图神经网络模型。

模型加速　**model acceleration**

降低神经网络模型推理时间，提高图神经网络模型运行和传输效率的方法。

模型压缩　**model compression**

减小神经网络模型规模，提高神经网络模型运行和传输效率的方法。

采样　**sampling**

指用部分图数据或图特征来代替完整图数据或图特征的方法。

量化　**quantization**

将输入值从一个大集合映射到一个较小集合的过程。

剪枝　**pruning**

将深度学习模型去掉一些影响性能较小的单元的方法。

结构化矩阵　**structured matrix**

一类特殊的矩阵，可以通过使用较少的数据以及一定的排列规律，构成完整的矩阵

分块结构化矩阵　**block structured matrix**

指可以分为多个块，且每个分块均按照某种规律排列的矩阵。

[来源：GB/T 42382.1-2023]

随机游走　**random walk**

遍历图中节点的方式，通过随机选择节点的相邻节点来生成节点序列，以捕捉图信息并用于图的表示学习。

点采样器　**node sampler**

根据起始节点，对输入的图数据进行采样的采样器。

层采样器　**layer sampler**

根据起始节点，对输入的图数据进行层级采样的采样器。

子图采样器　**subgraph sampler**

根据输入图数据进行子图采样的采样器。

1. 缩略语

下列缩略语适用于本文件。

MLP：多层感知机（Multilayer Perceptron）

GNN：图神经网络（Graph Neural Network）

GCN：图卷积网络（Graph Convolutional Networks）

GAT：图注意力网络（Graph Attention Networks）

GraphSAGE：图采样与图聚合（Graph Sample and Aggregation）

GIN：图同构网络（Graph Isomorphism Network）

GGNN：门控图神经网络（Gated Graph Neural Network）

LSTM：长短期记忆网络（Long Short Term Memory）

ResGatedgraph：残差门控图神经网络（Residual Gated Graph Neural Network）

GINE：带边特征的图同构网络（Graph Isomorphism Network with Edge features）

GaAN：图注意力网络（Graph Attention Network）

AM-GCN：自适应多通道图卷积网络（Adaptive Multi-channel Graph Convolutional Network）

FAGCN：频率自适应图卷积网络（Frequency Adaptation Graph Convolutional Networks）

GeomGCN：几何图卷积网络（Geometric Graph Convolutional Networks）

HGAT：异质图注意力网络（Heterogeneous Graph Attention Network）

GCNII：简单深度图卷积网络（Simple and Deep Graph Convolutional Networks）

SGC：简化图卷积网络（Simple Graph Convolution）

APPNP：近似个性化神经预测传播（Approximate Personalized Propagation of Neural Predictions）

GPRGNN：广义网页排名图神经网络（Generalized PageRank Graph Neural Network）

ChebNet：切比雪夫谱图卷积网络（Chebyshev Spectral Graph Convolutional Network）

JKnet：跳跃知识网络（Jumping Knowledge Networks）

DCNN：深度卷积神经网络（Deep Convolutional Neural Networks）

Line：大规模信息网络嵌入（Large-scale Information Network Embedding）

HPN：异质图传播网络（Heterogeneous Graph Propagation Network）

GAMLP：图注意力多层感知器（Graph Attention Multi-Layer Perceptron）

GNN-LF/HF：低频/高频图神经网络（Graph Neural Network with Low Frequency/High Frequency）

HeCo：具有协同对比学习的自监督异质图神经网络（Self-Supervised Heterogeneous Graph Neural Network with Co-Contrastive Learning）

HACUD：基于属性异质信息网络和分层注意机制的套现用户检测模型（Cash-Out User Detection based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism）

GTN：图转换器（以下简称“transformer”）网络（Graph Transformer Network）

SAT：简单实例自适应自训练半监督文本分类（Improving Semi-Supervised Text Classification with Simple Instance-Adaptive Self-Training）

GraphGPS：基于图的渐进式传播与搜索（Graph-based Progressive Propagation and Searching）

GloGNN：全局节点信息图神经网络（Graph Neural Network with Global Information）

R-GCN：关系图卷积网络（Relational Graph Convolutional Network）

HIN：异质信息网络（Heterogeneous Information Network）

BERT：双向编码器表征法（Bidirectional Encoder Representations from Transformers）

MCCF：多组件图卷积协同过滤模型（Multi-Component Graph Convolutional Collaborative Filtering）

MEIRec：用于意图推荐的元路径引导嵌入方法（Metapath-guided Embedding method for Intent Recommendation）

HERec：基于异质网络嵌入的推荐方法（Heterogeneous Information Network Embedding for Recommendation）

SEAL：从子图、嵌入和属性学习进行链接预测（Learning from Subgraphs, Embeddings and Attributes for Link Prediction）

GraIL：图归纳学习（Graph Inductive Learning）

NBFNet：神经贝尔曼-福特网络（Neural Bellman-Ford Network）

PAGNN：路径感知图神经网络（Path-aware Graph Neural Network）

Cluster-GCN：聚类图卷积网络（Cluster-Graph Convolution Network）

VQ-GNN：矢量量化图神经网络（Vector Quantization-Graph Neural Network）

LADIES：层相关重要性采样（Layer-Dependent Importance Sampling）

HGT：异质图转换器（Heterogeneous Graph Transformer）

GraphSaint：基于图采样的归纳学习（Graph Sampling Based Inductive Learning）

EXACT：极限激活压缩（Scalable Graph Neural Networks Training via Extreme Activation Compression）

SGC：简单图卷积网络（Simplifying Graph Convolutional Networks）

PPNP：基于个性化PageRank算法的图神经网络预测与传播（Predict then Propagate: Graph Neural Networks meet Personalized PageRank）

LSP：局部敏感剪枝（Locality-Sensitive Pruning）

LTD：元蒸馏（Learning to Distill）

GRACED：基于自定义知识蒸馏的图增强多层感知机（Graph Augmented MLPs via Customized Knowledge Distillation）

ROD：接收感知在线蒸馏（Reception-aware Online Distillation for Sparse Graphs）

GNN-SD：图神经网络自蒸馏（GNN Self-Distillation）

IGSD：迭代图自蒸馏（Iterative Graph Self-Distillation）

Node2Vec：一种学习网络中节点连续特征表示的算法框架（Node to Vector）

DiffPool：可微图池化（Differentiable Graph Pooling）

MPSN：动作感知伪暹罗网络（Motion-aware Pseudo Siamese Network）

MPNN：消息传递图神经网络（Message Passing Neural Networks）

D-MPNN：有向消息传递神经网络（Directed Message Passing Neural Networks）

DmoN：深度模块化网络（Deep Modularity Networks）

GMN：图匹配网络（Graph Matching Networks）

FGNN：因子图神经网络（Factor Graph Neural Network）

molGAN：分子生成对抗网络（Molecular Generative Adversarial Network）

GRAN：图循环注意网络（Graph Recurrent Attention Networks）

BNS：边界节点采样（Boundary Node Sampling）

SSP ：失效同步并行（Stale Synchronous Parallel）

KNN ：K最近邻（K Nearest Neighbour）

1. 图神经网络表示与模型压缩概述

图神经网络表示和模型压缩包括图神经网络的表示格式、图神经网络压缩和加速方法，旨在提高图神经网络模型在各类设备上的开发与运行效率，以及图神经网络计算框架，总体架构如图1所示。

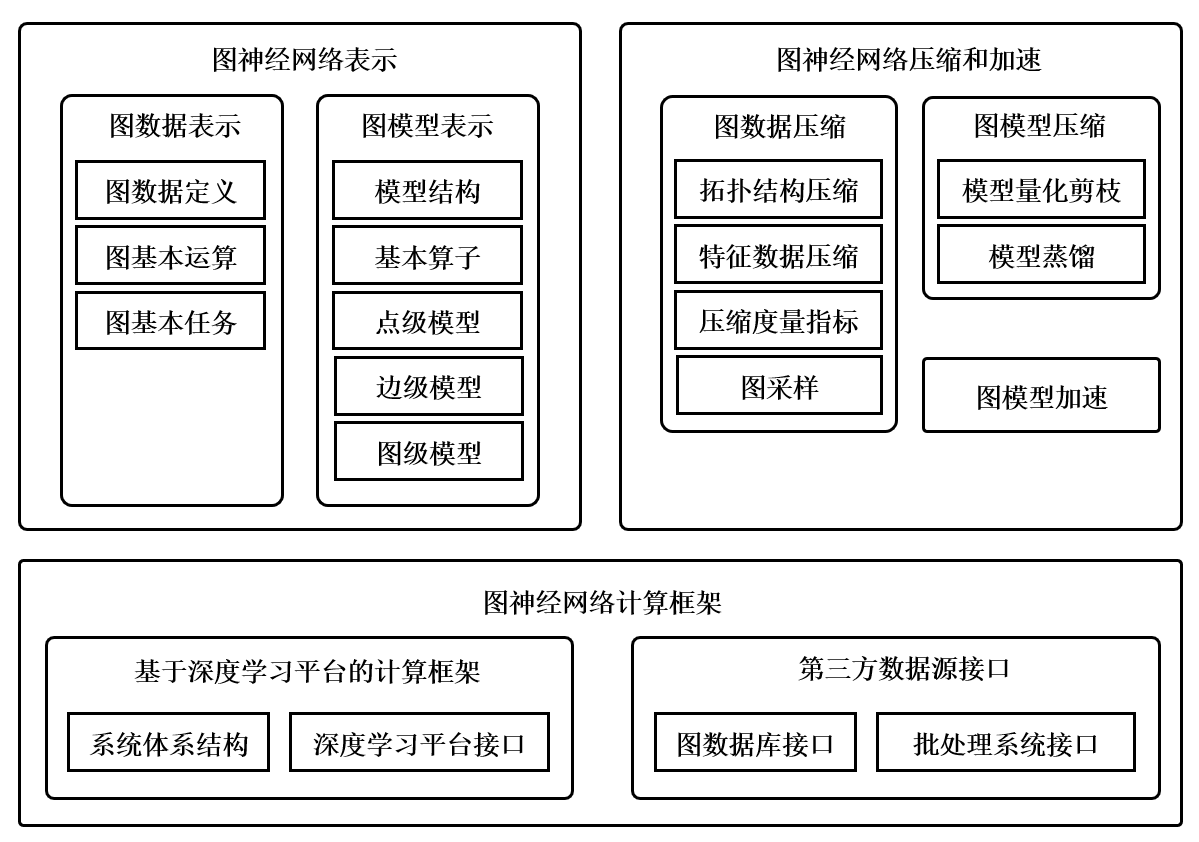


图1　图神经网络表示与模型压缩总体架构

总体架构包括以下三个部分：

1. 图神经网络表示：规范了图神经网络的表示方法，包括图数据的表示方法和图神经网络模型的表示方法。图数据表示中涵盖基本数据类型定义、图数据类型定义、图基本运算定义以及图基本任务定义。图神经网络模型表示中涵盖模型结构和基础算子定义，并从任务划分的角度规范了图神经网络点级模型、边级模型和图级模型的算子接口。具体要求见第6章和第7章；
2. 图神经网络压缩和加速：规范了图神经网络压缩方法和模型加速策略及其算子接口。压缩方法包括图数据的压缩以及图模型的压缩两个部分。图数据的压缩包括图拓扑结构压缩、图特征数据压缩以及图采样。图模型的压缩包括模型量化剪枝和模型蒸馏。对于图数据和模型的压缩方法，均定义了用于度量性能的指标。图模型加速包括并行加速策略、迭代加速策略、图划分策略以及通信加速策略。具体要求见第8章；
3. 图神经网络计算框架：规范了基于深度学习平台的图神经网络计算框架，以及该计算框架与第三方数据源的接口定义。基于深度学习平台的图神经网络计算框架定义了系统逻辑体系结构，以及与深度学习平台的接口规范。与第三方数据源接口定义包括图数据库接口和批处理系统接口。具体要求见第9章节。
4. 图数据表示
   1. 基本定义
      1. 概述

图数据是一种用于表示对象及其关系的结构化数据形式。本章节定义了用于完成图基本运算和图的基本任务的图数据结构，以及用于组成图数据结构的基本数据类型。

* + 1. 数据结构定义

消息为数据结构的基本单元，字段为构成消息的基本元素，一个消息包含一个或多个字段。如图2所示。

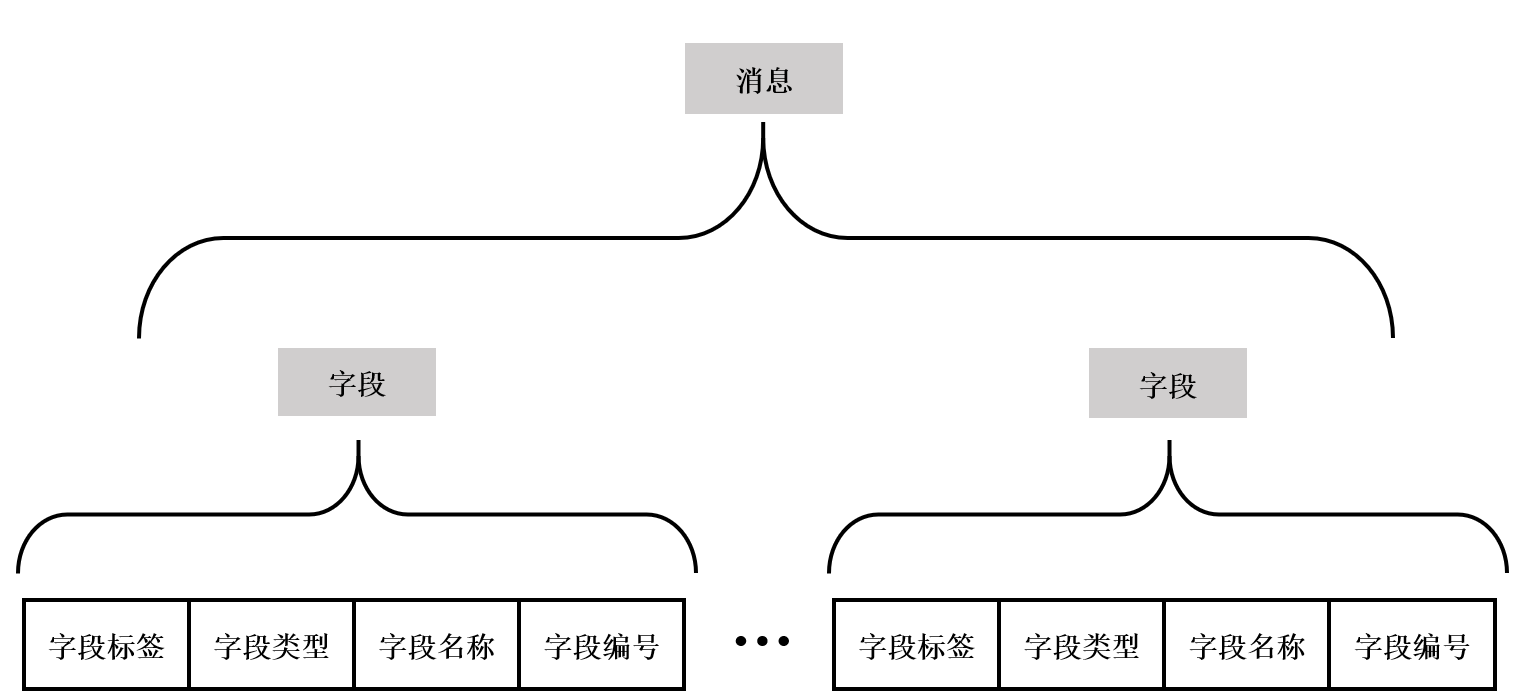


图2　数据结构

每个字段应由标签、类型、名称和编号等子字段组成。

1. 字段标签：字段标签用于定义字段的属性和使用规则，默认情况下，所有字段均被视为可选（"optional"），无需显式声明，以简化编程语言的使用，并确保向后兼容性；当字段需要表示多个值时，应使用"repeated"关键字进行标注，以明确该字段可以包含一个以上的值；
2. 字段类型：字段类型定义了字段可以存储的数据类型，类型包括int32、float、double、bool和string等标准数据类型，以及枚举（enum）、映射（map）、消息（message）等复合类型；
3. 字段名称：字段名称是在消息定义中为字段指定的唯一名称，用于标识和访问消息的具体部分；
4. 字段编号：字段编号是一个正整数，用于在消息的二进制格式中唯一标识字段。
5. 字段编号是消息序列化和反序列化过程中的关键，确保了即使字段名称在不同语言间有所不同，数据的结构仍能被正确理解。
   * 1. 基本数据类型定义
        1. 整数定义

整数定义见表1。

表1　整数定义

|  |  |  |
| --- | --- | --- |
| int | 指定参数 | |
| 字段 | 类型 | 定义 |
| value | oneof {uint32, uint64, int32, int64, sint32, sint64, fixed32, fixed64, sfixed32, sfixed64} | 一组整数类型，确保在不同情况下使用适当的数值范围和编码效率 |

* + - 1. 浮点数定义

浮点数定义见表2。

表2　浮点数定义

|  |  |  |
| --- | --- | --- |
| float | 指定参数 | |
| 字段 | 类型 | 定义 |
| value | oneof {float, double} | 一组浮点数类型，确保在不同情况下使用适当的数值范围和编码效率 |

* + - 1. 元组定义

元组定义见表3。

表3　元组定义

|  |  |  |
| --- | --- | --- |
| Tuple | 指定参数 | |
| 字段 | 类型 | 定义 |
| element-1 | DataType | 元组的第一个元素 |
| element-2 | DataType | 元组的第二个元素 |
| … | … | … |
| element-n | DataType | 元组的第n个元素 |

* + - 1. 列表定义

列表定义见表4。

表4　列表定义

|  |  |  |
| --- | --- | --- |
| List | 指定参数 | |
| 字段 | 类型 | 定义 |
| elements | DataType [repeated] | 列表的各元素 |

* + - 1. 字典定义

字典定义见表5。

表5　字典定义

|  |  |  |
| --- | --- | --- |
| Dict | 指定参数 | |
| 字段 | 类型 | 定义 |
| entries | map<DataType, DataType> | 字典的键值对 <Key, value> 映射, 其中浮点数、消息类型和枚举类型不被允许作为键出现。 |

* + - 1. 张量定义

张量定义见表6。

表6　张量定义

|  |  |  |
| --- | --- | --- |
| tensor | 指定参数 | |
| 字段 | 类型 | 定义 |
| dtype | DataType | 张量的数据类型,通常是数值类型，例如int、float、double等 |
| shape | int64 [repeated] | 张量的形状 |
| data | DataType [repeated] | 张量存储的数据 |
| require\_grad | bool | 张量是否需要计算梯度 |
| device | string | 张量位于的设备 |

* + - 1. 稀疏张量定义

稀疏张量定义见表7。

表7　稀疏张量定义

|  |  |  |
| --- | --- | --- |
| SparseTensor | 指定参数 | |
| 字段 | 类型 | 定义 |
| dtype | DataType | 稀疏张量的数据类型,通常是数值类型，例如int、float、double等 |
| dense\_shape | int64 [repeated] | 稀疏张量稠密表示下的形状 |
| format | enum {"coo", "csc", "csr"} | 稀疏张量的存储格式 |
| data | DataType [repeated] | 稀疏张量非零元素的数据 |
| index\_format | oneof {COOIndex, CSCIndex, CSRIndex} | 稀疏张量索引的格式 |

表7　稀疏张量定义（续）

|  |  |  |
| --- | --- | --- |
| SparseTensor | 指定参数 | |
| 字段 | 类型 | 定义 |
| require\_grad | bool | 稀疏张量是否需要计算梯度 |
| device | string | 稀疏张量位于的设备 |

* + - 1. 坐标格式索引

坐标格式索引定义见表8。

表8　坐标格式索引

|  |  |  |
| --- | --- | --- |
| COOIndex | 指定参数 | |
| 字段 | 类型 | 定义 |
| row\_indices | int64 [repeated] | COO格式的行索引 |
| col\_indices | int64 [repeated] | COO格式的列索引 |

* + - 1. 压缩稀疏行索引

压缩稀疏行索引定义见表9。

表9　压缩稀疏行索引定义

|  |  |  |
| --- | --- | --- |
| CSRIndex | 指定参数 | |
| 字段 | 类型 | 定义 |
| rowptr | int64 [repeated] | CSR格式的行指针 |
| col\_indices | int64 [repeated] | CSR格式的列索引 |

* + - 1. 压缩稀疏列索引

压缩稀疏列索引定义见表10。

表10　压缩稀疏列索列定义

|  |  |  |
| --- | --- | --- |
| CSCIndex | 指定参数 | |
| 字段 | 类型 | 定义 |
| row\_indices | int64 [repeated] | CSC格式的行索引 |
| colptr | int64 [repeated] | CSC格式的列指针 |

* 1. 图数据类型定义
     1. 同质图定义

同质图定义见表11。

表11　同质图定义

|  |  |  |
| --- | --- | --- |
| Graph | 指定参数 | |
| 字段 | 类型 | 定义 |
| X | tensor | 节点特征矩阵 |
| edge\_index | oneof {tensor, SparseTensor} | 边索引 |
| edge\_weight | tensor | 边特征矩阵 |
| Y | tensor | 节点或图标签 |
| pos | tensor | 节点位置 |

* + 1. 异质图定义

异质图定义见表12。

表12　异质图定义

|  |  |  |
| --- | --- | --- |
| HeteroGraph | 指定参数 | |
| 字段 | 类型 | 定义 |
| X\_dict | Dict [string, tensor] | 节点特征矩阵字典 |
| edge\_index\_dict | oneof {Dict [string, tensor], Dict [string, SparseTensor]} | 边索引字典 |
| edge\_weight\_dict | Dict [string, tensor] | 边特征矩阵字典 |
| Y\_dict | Dict [string, tensor] | 节点或图标签字典 |
| pos\_dict | Dict [string, tensor] | 节点位置字典 |

* + 1. 批量图定义

批量图定义见表13。

表13　批量图定义

|  |  |  |
| --- | --- | --- |
| BatchGraph | 指定参数 | |
| 字段 | 类型 | 定义 |
| X | tensor | 节点特征矩阵 |
| edge\_index | oneof {tensor, SparseTensor} | 边索引 |
| edge\_weight | tensor | 边特征矩阵 |
| Y | tensor | 图的标签 |
| pos | tensor | 节点位置矩阵 |
| batch | tensor | 每个节点所属的图 |
| ptr | tensor | 每个图在批量图中的范围 |

* + 1. 动态图定义

动态图定义见表14。

表14　动态图定义

|  |  |  |
| --- | --- | --- |
| DynamicGraph | 指定参数 | |
| 字段 | 类型 | 定义 |
| src | tensor | 源节点列表 |
| dst | tensor | 目标节点列表 |
| t | tensor | 事件时间戳列表 |
| msg | tensor | 消息特征矩阵 |

* + 1. 坐标图定义

坐标图定义见表15。

表15　坐标图定义

|  |  |  |
| --- | --- | --- |
| CoordinateGraph | 指定参数 | |
| 字段 | 类型 | 定义 |
| R | tensor | 节点坐标 |
| X | tensor | 节点特征 |
| X\_coord | tensor | 节点矢量特征 |
| edge\_index | oneof {tensor, SparseTensor} | 边索引 |
| edge\_attr | tensor | 边特征矩阵 |
| Y | tensor | 节点或图标签 |
| Y\_coord | tensor | 节点矢量标签 |
| dis\_coord | tensor | 坐标系中距离类型 |

* 1. 图基本运算

图基本运算的基本运算定义见表16~表48。

degree运算操作定义见表16**。**

表16　degree运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| degree | 计算给定的一维索引张量对应图的（未加权）度 | Input | index | 索引张量 | tensor |
| dtype | 返回张量的数据类型 | string |
| edge\_type | 边的类型 | string |
| num\_nodes | 节点的数量 | int |
| Output | Y | 输出张量 | tensor |

dropEdge运算操作定义见表17。

表17　dropEdge运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dropEdge | 根据伯努利分布，以概率p从图中随机丢弃边 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵 | tensor |
| p | 丢弃概率 | float |
| force\_undirected | 值为True时丢弃无向边的两个边。 | bool |
| num\_nodes | 节点的数量 | Tuple [int, int] |
| training | 值为False时将该操作无效化 | bool |
| Output | Y | 输出张量 | tensor |

sort\_edge\_index运算操作定义见表18。

表18　sort\_edge\_index运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| sort\_edge\_index | 根据节点ID对边索引进行排序 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵 | tensor |
| num\_nodes | 节点的数量 | int |
| sort\_by\_row | 如果设置为True，则按源节点ID对边索引排序，否则将按目标节点ID排序 | bool |
| Output | Y | 输出张量 | tensor |

add\_self\_loops运算操作定义见表19。

表19　add\_self\_loops运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| add\_self\_loops | 为每个节点添加自环，如果是加权图，则将添加自环并且根据fill\_value添加边权重 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵；如关键字包含g，删除该项 | tensor |
| g | 输入图 | Graph |
| fill\_value | 在edge\_weight不为空的情况下，该值会用来填充边权重 | tensor  float  string |
| num\_nodes | 节点数量 | int |
| allow\_duplicate | 如果设置为False，则会先去除所有自环以避免重复的自环 | bool |
| Output | edge\_index | 输出边索引 | tensor  SparseTensor |
| edge\_weight | 输出边权重，如输出g，则删除该项 | tensor |
| g | 输出图 | Graph |

remove\_self\_loops运算操作定义见表20。

表20　remove\_self\_loops运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| remove\_self\_loops | 删除图中的每个自环 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵；如关键字包含g，则删除该项 | tensor |
| g | 输入图 | Graph |
| Output | edge\_index | 输出边索引 | tensor  SparseTensor |

表20　remove\_self\_loops运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| remove\_self\_loops | 删除图中的每个自环 | Output | edge\_weight | 输出边权重或者多维边特征矩阵，如输出g，则删除该项 | tensor |
| g | 输出图 | Graph |

segregate\_self\_loops运算操作定义见表21。

表21　segregate\_self\_loops运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| segregate\_self\_loops | 从图中分离出自环 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵 | tensor |
| Output | edge\_index | 无自环边索引 | tensor  SparseTensor |
| edge\_weight | 无自环边特征矩阵 | tensor |
| loop\_edge\_index | 自环边索引 | tensor  SparseTensor |
| loop\_edge\_weight | 自环边特征矩阵 | tensor |

add\_remain\_self\_loops运算操作定义见表22。

表22　add\_remain\_self\_loops运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| add\_remain\_self\_loops | 将自环添加到没有自环的节点上，如果图被加权，则将添加自环的同时根据fill\_value添加自环的权重 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵 | tensor |
| fill\_value | 在edge\_weight不为空的情况下，该值会用来填充边权重 | float  tensor  string |
| num\_nodes | 节点数量 | int |

表22　add\_remain\_self\_loops运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| add\_remain\_self\_loops | 将自环添加到没有自环的节点上，如果图被加权，则将添加自环的同时根据fill\_value添加自环的权重 | Output | edge\_index | 边索引输出 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵输出 | tensor |

contains\_isolated\_nodes运算操作定义见表23。

表23　contains\_isolated\_nodes运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| contains\_isolated\_nodes | 判断图中是否含有孤立的节点 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| num\_nodes | 节点数量 | int |
| Output | has | 是否含有孤立的节点 | bool |

remove\_isolated\_nodes运算操作定义见表24。

表24　remove\_isolated\_nodes运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| remove\_isolated\_nodes | 删除图中孤立的节点 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重或者边特征矩阵 | tensor |
| num\_nodes | 节点数量 | int |
| Output | edge\_index | 邻接矩阵输出 | tensor  SparseTensor |
| edge\_weight | 边权重或者边特征矩阵输出 | tensor |
| mask | 孤立节点的掩码 | tensor |

subgraph运算操作定义见表25。

表25　subgraph运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| subgraph | 提取节点编号为subset的子图 | Input | subset | 保留的节点编号 | tensor  List [int] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵或者多维边特征矩阵；如关键字包含g，则删除该项 | tensor |
| g | 输入图 | Graph |
| relabel\_nodes | 如果设置为True，则得到的子图节点编号将变为从零开始的连续索引 | bool |
| num\_nodes | 节点数量 | int |
| return\_edge\_mask | 如果设置为True，将返回用于过滤多余的边特征的掩码 | bool |
| Output | edge\_index | 输出边索引 | tensor  SparseTensor |
| edge\_weight | 输出边权重或者多维边特征矩阵，如输出g，则删除该项 | tensor |
| g | 输出图 | Graph |
| edge\_mask | 边特征的掩码 | tensor |

k\_hop\_subgraph运算操作定义见表26。

表26　k\_hop\_subgraph运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| k\_hop\_subgraph | 计算k跳子图 | Input | node\_idx | 中心节点编号 | tensor  List [int]  int |
| num\_hops | 跳数 | int |
| edge\_index | 边索引 | tensor  SparseTensor |
| relabel\_nodes | 如果设置为True，则得到的子图节点编号将变为从零开始的连续索引 | bool |
| g | 输入图 | Graph |
| num\_nodes | 节点数量 | int |
| flow | k跳聚合的流向 | string |
| directed | 若设置为False，则会包括被采样到节点间所有的有向边 | bool |
| Output | subset | 子图中涉及的节点编号，如输出g，则删除该项 | tensor |
| edge\_index | 输出边索引 | tensor  SparseTensor |
| mapping | 从节点索引到其新位置的映射 | tensor |
| g | 输出图 | Graph |
| edge\_mask | 边掩码，指示边的保留情况，如输出g，则删除该项 | tensor |

get\_laplacian运算操作定义见表27。

表27　get\_laplacian运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| get\_laplacian | 计算出图的拉普拉斯矩阵 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| normalization | 拉普拉斯归一化方法 | string |
| num\_nodes | 节点数量 | int |
| Output | edge\_index | 拉普拉斯矩阵对应的边索引 | tensor  SparseTensor |
| edge\_weight | 拉普拉斯矩阵边权重矩阵输出 | tensor |

to\_dense\_batch运算操作定义见表28。

表28　to\_dense\_batch运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| to\_dense\_batch | 将批量图上的节点特征转换为各个子图上稠密的节点特征 | Input | X | 节点特征矩阵 | tensor |
| batch | 每一个节点所属批次 | tensor |
| fill\_value | 填充输出节点缺失的特征 | float |
| max\_num\_nodes | 各个批次最大节点数目 | int |
| batch\_size | 批量大小 | int |
| to\_dense\_batch | 将批量图上的节点特征转换为各个子图上稠密的节点特征 | Output | Y | 稠密节点特征矩阵 | tensor |
| mask | 节点掩码，指示哪些节点存在 | tensor |

to\_dense\_adj运算操作定义见表29。

表29　to\_dense\_adj运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| to\_dense\_adj | 将稀疏批量边索引张量转换为稠密张量 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| batch | 批向量，将每个节点分配一个批量 | tensor |
| edge\_weight | 边权重或者边特征矩阵 | tensor |
| max\_num\_nodes | 各个批次最大节点数目 | int |
| batch\_size | 批量大小 | int |
| Output | adj | 稠密的批量邻接矩阵输出 | tensor |

to\_sparse运算操作定义见表30。

表30　to\_sparse运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| to\_sparse | 将稠密的批量邻接矩阵转换为稀疏表示 | Input | adj | 稠密邻接矩阵 | tensor |
| mask | 描述节点的保留情况 | tensor |
| Output | edge\_index | 稀疏边索引输出 | tensor  SparseTensor |
| edge\_weight | 边权重或者边特征矩阵输出 | tensor |

normalized\_cut运算操作定义见表31。

表31　normalized\_cut运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| normalized\_cut | 计算加权图的归一化分割 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重或者边特征矩阵 | tensor |
| num\_nodes | 节点的数量 | int |
| Output | cut | 图的归一化分割 | tensor  SparseTensor |

grid运算操作定义见表32。

表32　grid运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| grid | 返回给定宽高的网格图的边索引以及节点位置矩阵 | Input | height | 网格图的高 | int |
| width | 网格图的宽 | int |
| Output | edge\_index | 网格图边索引 | tensor  SparseTensor |
| pos | 网格图节点位置矩阵 | tensor |

softmax\_nodes运算操作定义见表33。

表33　softmax\_nodes运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| softmax\_nodes | 对节点特征执行softmax计算 | Input | g | 输入图 | Graph  HeteroGraph |
| X | 输入的节点特征 | tensor |
| node\_type | 节点类型 | string |
| Output | Y | 归一化后的节点特征 | tensor |

softmax\_edges运算操作定义见表34。

表34　softmax\_edges运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| softmax\_edges | 对边特征执行softmax计算 | Input | g | 输入图 | Graph  HeteroGraph |
| edge\_weight | 输入的边特征 | tensor |
| edge\_type | 边类型 | string |
| Output | edge\_weight | 归一化后的边特征 | tensor |

broadcast\_nodes运算操作定义见表35。

表35　broadcast\_nodes运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| broadcast\_nodes | 将图级特征广播到节点级表示 | Input | g | 输入图 | Graph  HeteroGraph |
| feature | 输入的图级表示 | tensor |
| node\_type | 节点类型 | string |
| Output | Y | 节点特征 | tensor |

broadcast\_edges运算操作定义见表36。

表36　broadcast\_edges运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| broadcast\_edges | 将图级特征广播到边级表示 | Input | g | 输入图 | Graph  HeteroGraph |
| feature | 输入的图级表示 | tensor |
| edge\_type | 边类型 | string |
| Output | edge\_weight | 边特征 | tensor |

graph\_partition运算操作定义见表37。

表37　graph\_partition运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| graph\_partition | 将图划分成若干分片 | Input | g | 输入图数据 | Graph  HeteroGraph |
| num\_part | 分片个数 | int |
| part\_method | 分片策略 | string |
| Output | Y | 输出图数据的列表 | Dict [Graph]  Dict [HeteroGraph] |

fps运算操作定义见表38。

表38　fps运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| fps | 一种用于点云数据采样的运算符 | Input | X | 节点特征矩阵 | tensor |
| batch | 样本点所属批次 | tensor |
| ratio | 采样率 | float |
| random\_start | 是否采用第一个节点作为起始点 | bool |
| batch\_size | 批量大小 | int |
| Output | Y | 输出张量 | tensor |

graclus运算操作定义见表39。

表39　graclus运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| graclus | 一种贪婪聚类运算符，该算法选择未标记的节点并将其与未标记的节点之一匹配 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| relabel\_idx | 是否将节点编号重新映射成连续的编号 | bool |

表39　graclus运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| graclus | 一种贪婪聚类运算符，该算法选择未标记的节点并将其与未标记的节点之一匹配 | Input | edge\_weight | 边权重矩阵，如关键字包含g，则删除该项 | tensor |
| num\_nodes | 节点数目，如输出g，则删除该项 | int |
| Output | Y | 输出节点编号 | tensor |

knn运算操作定义见表40。

表40　knn运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| knn | 用于为y中的每个元素查找x中的k个最近邻的运算符 | Input | X | 测试样本点坐标的集合 | tensor |
| Y | 训练样本点坐标的集合 | tensor |
| k | 选取训练样本点的数目 | int |
| batch\_x | X样本点的所属批次 | tensor |
| batch\_y | Y样本点的所属批次 | tensor |
| cosine | 是否采用余弦距离 | bool |
| num\_workers | 并发数目 | int |
| batch\_size | 批量大小 | int |
| Output | Z | 输出张量，表示选出的最近邻 | tensor |

knn\_graph运算操作定义见表41。

表41　knn\_graph运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| knn\_graph | 用于构建样本点满足k近邻的图 | Input | X | 点坐标集合 | tensor |
| k | 近邻大小 | int |
| batch | 批量向量，用于将每个节点分配给一个特定样本 | tensor |

表41　knn\_graph运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| knn\_graph | 用于构建样本点满足k近邻的图 | Input | dist | 采用的距离函数 | string |
| batch\_size | 批量大小 | int |
| algorithm | 采用的算法 | string |
| Output | g | 满足要求的k近邻图 | Graph |

nearest运算操作定义见表42。

表42　nearest运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| nearest | 用于为y中每个元素，在x中查找距离最近的元素的运算符 | Input | X | 测试样本点坐标的集合 | tensor |
| Y | 训练样本点坐标的集合 | tensor |
| batch\_x | X样本点的所属批次 | tensor |
| batch\_y | Y样本点的所属批次 | tensor |
| Output | Z | 输出张量，表示聚类结果 | tensor |

radius运算操作定义见表43。

表43　radius运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| radius | 用于为y中的每个元素找到x中距离不超过r的所有点的搜索运算符 | Input | X | 测试样本点坐标的集合 | tensor |
| Y | 训练样本点坐标的集合 | tensor |
| r | 距离半径 | float |
| batch\_x | X样本点的所属批次 | tensor |
| batch\_y | Y样本点的所属批次 | tensor |
| max\_num\_neighbors | y中每个元素返回的最大邻居数 | int |
| num\_workers | 并发数目 | int |

表43　radius运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| radius | 用于为y中的每个元素找到x中距离不超过r的所有点的搜索运算符 | Input | batch\_size | 批量大小 | int |
| Output | Z | 输出张量，表示半径内搜索结果 | tensor |

radius\_graph运算操作定义见表44。

表44　radius\_graph运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| radius\_graph | 构建距离在一定半径内的图 | Input | X | 点坐标集合 | tensor |
| r | 距离大小 | float |
| p | 距离计算范数 | float |
| self\_loop | 构建的图是否包含自环 | bool |
| compute\_mode | 计算模式 | string |
| get\_distances | 是否返回构建的图对应边的距离 | bool |
| Output | g | 满足距离半径在r之内的图 | Graph |

voxel\_grid运算操作定义见表45。

表45　voxel\_grid运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| voxel\_grid | 利用在点云上覆盖规则网格并进行聚类的体素网格池化运算符 | Input | pos | 输入张量，节点位置矩阵 | tensor |
| batch | 每一个样本点所属批次 | tensor |
| size | 每个体素（voxel）的尺寸 | float,  tensor  List [float] |
| start | 每个维度中网格的起始坐标 | float,  tensor  List [float] |

表45　voxel\_grid运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| voxel\_grid | 利用在点云上覆盖规则网格并进行聚类的体素网格池化运算符 | Input | end | 每个维度中网格的结束坐标 | float,  tensor  List [float] |
| Output | Y | 输出张量 | tensor |

dropNode运算操作定义见表46。

表46　dropNode运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dropNode | 使用来自伯努利分布的样本，以概率p从图节点中随机丢弃某些节点出发的所有边 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| X | 节点特征 | tensor |
| p | 丢弃概率 | float |
| drop\_num\_nodes | 丢弃的节点个数 | int |
| Output | edge\_index | 边索引输出 | tensor  SparseTensor |
| Y | 节点特征输出 | tensor |

dropFeature运算操作定义见表47。

表47　dropFeature运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dropFeature | 使用来自伯努利分布的样本，以概率p从图节点中随机丢弃所有节点某些维度的特征元素 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| X | 节点特征 | tensor |
| p | 丢弃概率 | float |
| drop\_num\_elements | 丢弃特征列数 | int |
| Output | edge\_index | 边索引输出 | tensor  SparseTensor |
| Y | 节点特征输出 | tensor |

dropMessage运算操作定义见表48。

表48　dropMessage运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dropMessage | 使用来自伯努利分布的样本，以概率p从图节点中随机丢弃节点发出消息中的元素 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| X | 节点特征 | tensor |
| edge\_attr | 边权重矩阵或者多维边特征矩阵 | tensor |
| p | 丢弃概率 | float |
| drop\_num | 丢弃节点发出的消息元素的个数 | int |
| Output | edge\_index | 边索引输出 | tensor  SparseTensor |
| Y | 节点特征输出 | tensor |

dropGraph运算操作定义见表49

表49　dropGraph运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dropGraph | 使用来自伯努利分布的样本，以概率p从图节点中随机丢弃某些节点及其Q阶邻居节点，并引入注意力机制，让被网络关注的区域具有更大的被丢弃的概率。 | Input | Edge\_index | 边索引 | tensor  SparseTensor |
| X | 节点特征 | tensor |
| Keep\_prob | 保持节点不被丢弃的概率 | Float |
| Q | 邻居节点范围 | Int |
| Output | Y | 节点特征输出 | tensor |

* 1. 图基本任务

图神经网络的下游任务按任务形式与监督形式进行划分：

1. 按任务形式划分，图神经网络的下游任务划分为节点级别任务、边级别任务和图级别任务：
   1. 节点级别任务分为：节点分类、节点回归、节点聚类等。节点分类是指对于给定的图中的节点，将其划分为某类别或标签；节点回归是指对给定图中的节点进行数值预测，用于预测节点的属性值；节点聚类是指将图中的节点划分为不同的群组或簇；
   2. 边级别任务分为：边分类、边回归、链接预测等。边分类是指对于给定的图中的边，将其划分为某类别或标签；边回归是指对给定图中的边进行数值预测，以预测边上的属性值或边的权重等信息；链接预测是指预测图中尚未出现但潜在存在的边；
   3. 图级别分为：图分类、图回归、图匹配、图生成等。图分类是指将给定的图划分为某类别或标签；图回归是指预测图的某些连续性目标变量；图匹配是指对给定的一对图进行相似度评估或匹配；图生成是指模型生成图对象实例，使其满足符合要求的某些特定性质。
2. 按监督形式划分，图神经网络的下游任务分为：监督训练、半监督训练、无监督训练三种形式：
   1. 监督训练是指每一个参与训练的样本对象都有对应的标记标签；
   2. 半监督训练是指仅有部分参与训练的样本对象有对应的标记标签。在测试阶段，直推式学习配置方式让图神经网络预测已经在训练阶段给出的未标记样本的标签值；归纳式学习配置方式让图神经网络预测未在训练阶段见过的全新样本；
   3. 无监督训练是指参与训练的样本没有来自外部的标记标签，模型寻找样本对象的内在模式，将其转化为监督信号来指导模型训练。预训练策略是使用图神经网络来学习样本对象的向量化表示，向量化表示应表征样本对象在图上的内在性质。
3. 图神经网络模型
   1. 模型结构
      1. 计算图定义

计算图定义见表50。

表50　计算图定义

|  |  |  |
| --- | --- | --- |
| Computational Graph | 指定参数 | |
| 字段 | 类型 | 定义 |
| node | NodeDef [repeated] | 计算图中所有操作节点的列表 |
| versions | Version | 计算图的兼容性版本信息 |
| library | FunctionDefLibrary | 提供用户定义函数的库 |
| debug\_info | GraphDebugInfo | 包含计算图节点的调试信息 |

* + - 1. 操作节点定义

操作节点定义见表51。

表51　操作节点定义

|  |  |  |
| --- | --- | --- |
| NodeDef | 指定参数 | |
| 字段 | 类型 | 定义 |
| name | string | 操作节点的唯一标识名 |
| op | string | 操作节点的操作类型 |

表51　操作节点定义（续）

|  |  |  |
| --- | --- | --- |
| NodeDef | 指定参数 | |
| 字段 | 类型 | 定义 |
| input | string [repeated] | 操作节点的输入列表 |
| device | string | 操作节点所在的设备 |
| attr | map<string, DataType> | 操作节点属性的键值对 |

* + - 1. 版本定义

版本定义见表52。

表52　版本定义

|  |  |  |
| --- | --- | --- |
| Version | 指定参数 | |
| 字段 | 类型 | 定义 |
| producer | int32 | 生成这个数据的代码的版本 |
| min\_consumer | int32 | 任何低于此版本的消费者不允许使用这个数据 |
| bad\_consumers | int32 [repeated] | 不允许使用这个数据的特定消费者版本（通常由于bug） |

* + - 1. 函数库定义

函数定义库见表53。

表53　函数库定义

|  |  |  |
| --- | --- | --- |
| FunctionDefLibrary | 指定参数 | |
| 字段 | 类型 | 定义 |
| function | FunctionDef [repeated] | 函数定义的集合 |
| gradient | GradientDef [repeated] | 每个函数的梯度函数定义 |
| registered\_gradients | RegisteredGradient [repeated] | 注册在函数库中使用的梯度函数 |

函数定义见表54。

表54　函数定义

|  |  |  |
| --- | --- | --- |
| FunctionDef | 指定参数 | |
| 字段 | 类型 | 定义 |
| signature | OpDef | 函数的签名，包括函数名称、参数、返回值、属性等信息 |
| attr | map<string, DataType> | 特定于此函数定义的属性 |
| arg\_attr | map<uint32, map<string, DataType>> | 函数参数的属性 |
| resource\_arg\_unique\_id | map<uint32, uint32> | 每个资源参数的唯一标识符 |
| node\_def | NodeDef [repeated] | 函数体中的节点定义 |
| ret | map<string, string> | 函数的返回值与函数体中节点输出的映射 |
| control\_ret | map<string, string> | 函数的控制输出与函数体中节点名称的映射 |

函数签操作定义见表55。

表55　函数签名操作定义

|  |  |  |
| --- | --- | --- |
| OpDef | 指定参数 | |
| 字段 | 类型 | 定义 |
| name | string | 操作的名称，以大写字母开头，符合特定的正则表达式规则 |
| input\_arg | string [repeated] | 输入参数的描述列表 |
| output\_arg | string [repeated] | 输出参数的描述列表 |
| control\_output | string [repeated] | 命名控制输出的列表，用于复合操作 |
| attr | string [repeated] | 图构建时配置的描述，即NodeDef中将指定的属性字段 |
| summary | string | 操作的简略可读描述 |
| description | string | 操作的详细可读描述 |
| is\_commutative | bool | 指示操作是否是可交换的 |
| is\_aggregate | bool | 指示操作是否是聚合操作 |
| is\_stateful | bool | 指示操作是否是有状态的 |

表55　函数签名操作定义（续）

|  |  |  |
| --- | --- | --- |
| OpDef | 指定参数 | |
| 字段 | 类型 | 定义 |
| allows\_uninitialized\_input | bool | 指示操作是否允许未初始化输入 |
| is\_distributed\_communication | bool | 指示操作实现是否使用分布式通信 |

梯度定义见表56。

表56　梯度定义

|  |  |  |
| --- | --- | --- |
| GradientDef | 指定参数 | |
| 字段 | 类型 | 定义 |
| function\_name | string | 定义梯度函数的函数名称 |
| gradient\_func | string | 对应的梯度函数名称 |

注册梯度定义见表57。

表57　注册梯度定义

|  |  |  |
| --- | --- | --- |
| RegisteredGradient | 指定参数 | |
| 字段 | 类型 | 定义 |
| gradient\_func | string | 梯度函数的名称 |
| registered\_op\_type | string | 梯度函数注册的操作类型 |

* + - 1. 计算图调试信息

计算图调试信息定义见表58。

表58　计算图调试信息定义

|  |  |  |
| --- | --- | --- |
| GraphDebugInfo | 指定参数 | |
| 字段 | 类型 | 定义 |
| files | string [repeated] | 所有源代码文件的名称列表，可通过file\_index索引 |
| frames\_by\_id | map<fixed64, FileLineCol> | 按唯一标识符索引的帧到文件行列信息的映射 |

表58　计算图调试信息定义（续）

|  |  |  |
| --- | --- | --- |
| GraphDebugInfo | 指定参数 | |
| 字段 | 类型 | 定义 |
| traces\_by\_id | map<fixed64, StackTrace> | 按唯一标识符索引的栈追踪信息的映射 |
| name\_to\_trace\_id | map<string, fixed64> | 将节点名称映射到traces\_by\_id中包含的跟踪ID |

文件行列定义见表59。

表59　文件行列定义

|  |  |  |
| --- | --- | --- |
| FileLineCol | 指定参数 | |
| 字段 | 类型 | 定义 |
| file\_index | int32 | 文件名的索引，用于从files字符串列表中检索具体文件名 |
| line | int32 | 在文件中的行号 |
| col | int32 | 在文件中的列号 |
| func | string | 包含该文件行的函数名称 |
| code | string | 该文件行包含的源代码 |

栈追踪定义见表60。

表60　栈追踪定义

|  |  |  |
| --- | --- | --- |
| StackTrace | 指定参数 | |
| 字段 | 类型 | 定义 |
| frame\_id | fixed64 [repeated] | 栈帧的唯一标识符，用于标识栈追踪中的每个帧的ID |

* + 1. 图神经网络模型结构定义

图神经网络模型结构定义见表61。

表61　图神经网络模型结构定义

|  |  |  |
| --- | --- | --- |
| Model | 指定参数 | |
| 字段 | 类型 | 定义 |
| version | Version | 模型定义的版本号 |
| contributors | string [repeated] | 贡献者列表 |
| framework\_name | string | 框架名称，如TensorFlow、PyTorch |
| framework\_version | string | 使用的框架版本 |
| model\_name | string | 模型名称 |
| model\_version | string | 模型版本号 |
| graph | Computational Graph [repeated] | 描述模型计算图的结构 |
| attribute | map<string, DataType> | 模型属性的键值对，描述模型的特性 |
| signature | OpDef | 模型输入输出等参数信息的签名定义 |
| description | string | 模型的详细描述 |

* 1. 基础算子
     1. 概述

基础算子是图神经网络中传递和聚合消息的基本组件，分为消息传递算子、池化算子和归一化算子。

* + 1. 消息传递算子
       1. 概述

消息传递算子是图神经网络中的核心操作，模拟消息在图结构中的传播和交互过程。其计算过程包括三个子算子：消息算子、聚合算子、更新算子。基于消息传递的图神经网络卷积算子，可通过消息算子、聚合算子和更新算子的组合方式实现。

* + - 1. 消息算子

消息算子是定义在每条边上，通过将边的特征与其附带节点的特征相结合来生成信息的操作。

send\_message运算操作定义见表62。

表62　send\_message运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| send\_message | 自定义消息发送函数 | Input | src\_feat | 源节点特征 | tensor  Dict [string, tensor] |
| dst\_feat | 目标节点特征 | tensor  Dict [string, tensor] |
| edge\_feat | 边特征 | tensor  Dict [string, tensor] |
| edge\_index | 边索引 | tensor  SparseTenosr  Dict [string, tensor]  Dict [string, SparseTensor] |
| Output | Y | 输出张量字典 | tensor  Dict [string, tensor] |
| Attributes | message\_func | 消息产生模块 | Model |

* + - 1. 聚合算子

聚合算子是消息传递过程中的关键组件，用于从节点的邻居聚合消息。

具体定义见表63~表67。

recv\_message运算操作定义见表63。

表63　recv\_message运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| recv\_message | 自定义消息接收函数 | Input | X | 输入消息 | tensor  Dict [string, tensor] |

表63　recv\_message运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| recv\_message | 自定义消息接收函数 | Input | edge\_index | 边索引 | tensor  SparseTenosr  Dict [string, tensor]  Dict [string, SparseTensor] |
| Output | Y | 输出结果 | tensor  Dict [string, tensor] |
| Attributes | reduce\_func | 消息接收模块 | Model |

sum聚合操作定义见表64。

表64　sum聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| sum | 聚合来自邻居节点的特征，通过聚合函数“和”生成节点级表示 | Input | X | 接收的消息 | tensor  Dict [string, tensor] |
| edge\_index | 边索引 | tensor  SparseTenosr  Dict [string, tensor]  Dict [string, SparseTensor] |
| num\_nodes | 节点个数 | int |
| Output | Y | 输出聚合结果 | tensor  Dict [string, tensor] |

max聚合操作定义见表65。

表65　max聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| max | 聚合来自邻居节点的特征，通过聚合函数“最大”生成节点级表示 | Input | X | 接收的消息 | tensor  Dict [string, tensor] |
| edge\_index | 边索引 | tensor  SparseTenosr  Dict [string, tensor]  Dict [string, SparseTensor] |
| num\_nodes | 节点个数 | int |
| Output | Y | 输出聚合结果 | tensor  Dict [string, tensor] |

min聚合操作定义见表66。

表66　min聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| min | 聚合来自邻居节点的特征，通过聚合函数“最小”生成节点级表示 | Input | X | 接收的消息 | tensor  Dict [string, tensor] |
| edge\_index | 边索引 | tensor  SparseTenosr  Dict [string, tensor]  Dict [string, SparseTensor] |
| num\_nodes | 节点个数 | int |
| Output | Y | 输出聚合结果 | tensor  Dict [string, tensor] |

mean聚合操作定义见表67。

表67　mean聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| mean | 聚合来自邻居节点的特征，通过聚合函数“平均”生成节点级表示 | Input | X | 生成的消息 | tensor  Dict [string, tensor] |
| edge\_index | 边索引 | tensor  SparseTenosr  Dict [string, tensor]  Dict [string, SparseTensor] |
| num\_nodes | 节点个数 | int |
| Output | Y | 输出表示张量 | tensor  Dict [string, tensor] |

* + - 1. 更新算子

更新算子是消息传递过程中的关键组件，定义在每个节点上，把聚合的消息或自身的特征进行处理以更新节点特征。

update更新操作定义见表68。

表68　update更新操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| update | 在每个节点上定义，用于处理聚合的消息或自身的特征来更新节点特征 | Input | H | 聚合后的消息 | tensor  Dict [string, tensor] |
| X | 自身特征 | tensor  Dict [string, tensor] |
| num\_nodes | 节点个数 | int |
| Output | Y | 更新后的节点表征 | tensor  Dict [string, tensor] |

表68　update更新操作（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| update | 在每个节点上定义，用于处理聚合的消息或自身的特征来更新节点特征 | Attributes | update\_rule | 更新规则，如使用ReLU，线性层等 | Module |

* + - 1. 卷积算子

图卷积算子的具体定义见表69~表119。

GCNConv运算操作定义见表69。

表69　GCNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GCNConv | 构建谱域图卷积，使用特征张量和邻接矩阵，输出表示张量 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| add\_self\_loops | 是否给输入图添加自环 | bool |
| improved | 是否在添加自环操作时，添加两个自环 | bool |
| normalize | 是否进行归一化 | bool |
| bias | 是否添加偏置项 | bool |
| activation | 激活函数名称 | string |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |

ChebConv运算操作定义见表70。

表70　ChebConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ChebConv | Chebyshev谱图卷积运算符，使用特征张量和邻接矩阵，输出表示张量 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| edge\_weight | 边权重矩阵 | tensor |
| batch | 每个节点的批次所属 | tensor |
| lambda\_max | 拉普拉斯正则化矩阵的最大特征值 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| k | 切比雪夫滤波器大小 | int |
| normalize | 归一化方法 | string |
| bias | 是否添加偏置项 | bool |
| activation | 激活函数名称 | string |

SAGEConv运算操作定义见表71。

表71　SAGEConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SAGEConv | 通过对图中的节点进行采样和聚合操作来学习节点的表示，从而能够支持对图数据的有效的学习和推理 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |

表71　SAGEConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SAGEConv | 通过对图中的节点进行采样和聚合操作来学习节点的表示，从而能够支持对图数据的有效的学习和推理 | Input | size | 邻接矩阵大小 | Tuple [int, int] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |
| normalize | 是否进行归一化 | bool |
| aggr | 聚合方法 | string  List [string] |
| feat\_drop | 特征的丢弃率 | float |
| activation | 激活函数名称 | string |
| bias | 是否添加偏置项 | bool |

GraphConv运算操作定义见表72。

表72　GraphConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphConv | 构建高阶图神经网络，使用特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | int  Tuple [int, int] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |

表72　GraphConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphConv | 构建高阶图神经网络，使用特征张量和邻接矩阵，输出特征张量 | Attributes | aggr | 要使用的聚合方法 | string |
| bias | 是否添加偏置项 | bool |

GravNetConv运算操作定义见表73。

表73　GravNetConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GravNetConv | 使用最近邻居动态构建图。邻居通过特征空间的可学习的低维投影来构造。然后，使用距离权重将输入特征空间的第二个投影从相邻点传播到每个顶点，该距离权重是通过对距离应用高斯函数得出的 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| batch | 每一个节点的批次所属 | tensor  Tuple [tensor, tensor] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| space\_dimesions | 用于构造邻居的空间的维数 | int |
| k | 最近邻居的数量 | int |

GatedgraphConv运算操作定义见表74。

表74　GatedgraphConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GatedgraphConv | 构建门控图卷积运算符，使用特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_type | 边的类型 | tensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |

表74　GatedgraphConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GatedgraphConv | 构建门控图卷积运算符，使用特征张量和邻接矩阵，输出特征张量 | Attributes | out\_channels | 输出特征的维度 | int |
| num\_steps | 序列长度 | int |
| aggr | 要使用的聚合方法 | string |
| n\_etypes | 边类型数目 | int |
| bias | 是否添加偏置项 | bool |

ResGatedgraphConv运算操作定义见表75。

表75　ResGatedgraphConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ResGatedgraphConv | 构建残差门控图卷积运算符，使用特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边的权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| edge\_dim | 边的特征维度 | int |
| act | 激活函数名称 | string |
| bias | 是否添加偏置项 | bool |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |

GATConv运算操作定义见表76。

表76　GATConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GATConv | 构建图注意力运算符，使用特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| get\_attention | 是否返回注意力权重 | bool |
| Output | Y | 输出特征表示 | tensor  Tuple [tensor, tensor] |
| attention\_weight | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| heads | 多头注意力的数量 | int |
| concat | 如果是False，对多头注意力进行average操作，如果是True，则进行concat操作 | bool |
| negative\_slope | 负斜率的LeakyReLU角度 | float |
| feat\_drop | 特征的丢弃率 | float |
| attn\_drop | 在训练过程中，将每个节点暴露于随机采样的邻域的归一化注意力系数的丢弃概率 | float |
| bias | 是否添加偏置项 | bool |
| add\_self\_loops | 是否给输入图添加自环 | bool |

表76　GATConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GATConv | 构建图注意力运算符，使用特征张量和邻接矩阵，输出特征张量 | Attributes | edge\_dim | 边特征维度 | int |
| fill\_value | 用于生成自环的边特征 | float  tensor  string |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |
| residual | 是否使用残差连接 | bool |
| activation | 激活函数名称 | string |

TransformerConv运算操作定义见表77。

表77　TransformerConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TransformerConv | 构建图transformer运算符，使用特征张量、邻接矩阵和边属性张量，输出特征张量 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边特征矩阵 | tensor |
| get\_attention | 是否返回注意力权重 | bool |
| Output | Y | 输出张量 | tensor |
| attention\_weight | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| heads | 多头注意力的数量 | int |
| concat | 如果是False，对多头注意力进行average操作，如果是True，则进行concat操作 | bool |

表77　TransformerConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TransformerConv | 构建图transformer运算符，使用特征张量、邻接矩阵和边属性张量，输出特征张量 | Attributes | beta | 通过beta加权聚合信息和跳过信息 | bool |
| dropout | 在训练过程中，将每个节点暴露于随机采样的邻域的归一化注意力系数的丢弃概率 | float |
| edge\_dim | 边特征的维度 | int |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |
| bias | 是否添加偏置项 | bool |

AGNNConv运算操作定义见表78。

表78　AGNNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| AGNNConv | 通过引入注意力机制来加权邻居节点的贡献，使模型能够更加关注于重要的邻居节点，从而提高图神经网络处理图数据的能力 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出张量 | tensor |
| Attributes | requires\_grad | 参数beta是否可学习 | bool |
| add\_self\_loops | 是否给输入图添加自环 | bool |
| beta | 参数beta的初始值 | float |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |

TAGConv运算操作定义见表79。

表79　TAGConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TAGConv | 基于节点特征自适应的图卷积方法，它通过考虑节点自身特征和邻居节点特征之间的关系来更新节点的表示 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| k | 跳数k | int |
| activation | 激活函数名称 | string |
| normalize | 是否进行归一化 | bool |
| bias | 是否添加偏置项 | bool |

GINConv运算操作定义见表80。

表80　GINConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GINConv | 基于图同构的卷积操作。不考虑节点之间的顺序关系，而是专注于节点之间的结构关系 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_weight | 边的权重 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | nn | 将输入节点特征映射到输出维度的神经网络方法 | Model |
| eps | 初始自环缩放参数 | float |

表80　GINConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GINConv | 基于图同构的卷积操作。不考虑节点之间的顺序关系，而是专注于节点之间的结构关系 | Attributes | train\_eps | 初始自环缩放参数是否可学习 | bool |
| activation | 激活函数名称 | string |
| aggregation\_type | 聚合类型 | string |

GINEConv运算操作定义见表81。

表81　GINEConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GINEConv | 改进的图同构运算符，使用一种新的策略和自监督方法来预训练图神经网络 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| edge\_weight | 边特征矩阵 | tensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | nn | 将输入节点特征映射到输出维度的神经网络方法 | Model |
| edge\_dim | 边特征维度 | int |
| eps | 初始自环缩放参数 | float |
| train\_eps | 初始自环缩放参数是否可学习 | bool |

ARMAConv运算操作定义见表82。

表82　ARMAConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ARMAConv | 构建ARMA图卷积运算符，通过结合自回归和移动平均模型的思想，对图上的节点进行特征更新 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| num\_stacks | 并行栈数 | int |
| num\_layers | 层数 | int |
| act | 激活函数名称 | string |
| shared\_weights | 每个栈是否共享相同的参数 | bool |
| dropout | 丢弃率 | float |
| bias | 是否添加偏置项 | bool |

SGConv运算操作定义见表83。

表83　SGConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SGConv | 简化传统图卷积网络中的计算复杂度，并提高模型的效率的图卷积算子 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| k | 跳数 | int |
| cached | 是否缓存第一次执行特征聚合的结果 | bool |

表83　SGConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SGConv | 简化传统图卷积网络中的计算复杂度，并提高模型的效率的图卷积算子 | Attributes | add\_self\_loops | 是否添加自环 | bool |
| normalize | 归一化函数名称 | string |
| activation | 激活函数名称 | string |
| bias | 是否添加偏置项 | bool |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |

APPNPConv运算操作定义见表84。

表84　APPNPConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| APPNPConv | 构建神经预测层的近似个性化预测传播运算符，使用特征张量、邻接矩阵和边权重矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | k | 跳数 | int |
| alpha | 传送概率参数 | float |
| edge\_drop | 节点接收的消息丢弃率 | float |
| cached | 是否缓存第一次执行的对称归一化邻接矩阵的结果 | bool |
| add\_self\_loops | 是否添加自环 | bool |
| normalize | 是否进行归一化 | bool |

MFConv运算操作定义见表85。

表85　MFConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MFConv | 通过对分子图进行卷积操作，捕捉分子的结构特征，生成分子指纹，用于分子属性预测、药物筛选等应用的卷积算子 | Input | X | 节点特征矩阵 | tenso  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | tensor  Tuple [tensor, tensor] |
| out\_channels | 输出特征的维度 | int |
| max\_dgree | 节点更新最大节点度 | int |
| bias | 是否添加偏置项 | bool |

SignedConv运算操作定义见表86。

表86　SignedConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SignedConv | SignedConv利用了正负边的信息来调整节点表示，使得节点能够更好地捕捉图中正负边的影响 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| pos\_edge\_index | 正向邻接矩阵 | tensor  SparseTensor |
| neg\_edge\_index | 负向邻接矩阵 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |

表86　SignedConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SignedConv | SignedConv利用了正负边的信息来调整节点表示，使得节点能够更好地捕捉图中正负边的影响 | Attributes | out\_channels | 输出特征的维度 | int |
| first\_aggr | 是否进行额外一次聚合 | bool |
| bias | 是否添加偏置项 | bool |

DNAConv运算操作定义见表87。

表87　DNAConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DNAConv | 在节点更新时，根据节点自身特征和邻居节点的特征动态地确定邻域大小和邻域节点的重要性，从而实现对节点表示的动态聚合 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | channels | 输入或输出特征维度 | int  Tuple [int, int] |
| groups | 进行线性映射的组数 | int |
| heads | 多头注意力的数量 | int |
| dropout | 注意系数的丢弃概率 | float |
| cached | 是否缓存第一次执行的对称归一化邻接矩阵的结果 | bool |
| normalize | 是否进行归一化 | bool |
| add\_self\_loops | 是否添加自环 | bool |
| bias | 是否添加偏置项 | bool |
| 此处x输入的节点特征的形状为[num\_nodes, num\_layers, channels]。 | | | | | |

PointConv运算操作定义见表88。

表88　PointConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PointConv | 专用于点云数据，在计算节点表示时，不再依赖于节点之间的连接关系，而是仅基于点的坐标和属性信息来进行卷积操作 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| pos | 点坐标矩阵 | tensor  Tuple [tensor, tensor] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | local\_nn | 映射节点特征和相对空间坐标的神经网络 | Model |
| global\_nn | 映射聚合后的节点特征的神经网络 | Model |
| add\_self\_loops | 是否添加自环 | bool |

GMMConv运算操作定义见表89。

表89　GMMConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GMMConv | 利用高斯混合模型学习生成每个节点的自适应卷积核。这些卷积核允许网络根据节点的局部邻域结构和特征动态地调整，从而更好地捕捉图中的信息 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| pseudo | 伪坐标张量大小 | tensor |
| g | 边索引 | Graph |
| edge\_index | 输入图 | tensor  SparseTensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| edge\_weight | 边特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |

表89　GMMConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GMMConv | 利用高斯混合模型学习生成每个节点的自适应卷积核。这些卷积核允许网络根据节点的局部邻域结构和特征动态地调整，从而更好地捕捉图中的信息 | Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| dim | 伪坐标维度 | int |
| k | 核的数量 | int |
| separate\_gaussians | 为每对输入和输出通道学习单独的GMM | bool |
| aggr | 要使用的聚合方法 | string |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |
| coord\_dim | 伪坐标的维数 | int |
| residual | 是否使用残差连接 | bool |
| bias | 是否添加偏置项 | bool |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |

SplineConv运算操作定义见表90。

表90　SplineConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SplineConv | 使用B-spline基函数来建模图上的卷积操作。从而捕捉图中节点之间的局部结构和关系，允许网络学习节点特征之间更复杂的关联 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| edge\_weight | 边特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |

表90　SplineConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SplineConv | 使用B-spline基函数来建模图上的卷积操作。从而捕捉图中节点之间的局部结构和关系，允许网络学习节点特征之间更复杂的关联 | Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| dim | 伪坐标维度 | int |
| kernel\_size | 卷积核的大小 | int  List [int] |
| is\_open\_spline | 是否在此维度上使用封闭的B样条曲线 | bool |
| degree | B样条基度 | int |
| aggr | 要使用的聚合方法 | string |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |
| bias | 是否添加偏置项 | bool |

NNConv运算操作定义见表91。

表91　NNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| NNConv | 基于自适应卷积核的卷积操作，通过神经网络动态地学习每个节点的卷积核。在节点更新过程中，其利用节点自身的特征以及与其相邻的节点的特征，通过神经网络确定每个节点的自适应卷积核，进而实现节点特征的更新和聚合 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| edge\_weight | 边特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |

表91　NNConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| NNConv | 基于自适应卷积核的卷积操作，通过神经网络动态地学习每个节点的卷积核。在节点更新过程中，其利用节点自身的特征以及与其相邻的节点的特征，通过神经网络确定每个节点的自适应卷积核，进而实现节点特征的更新和聚合 | Attributes | out\_channels | 输出特征的维度 | int |
| nn | 映射边特征的神经网络 | Model |
| aggr | 要使用的聚合方法 | string |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |
| residual | 是否使用残差连接 | bool |
| bias | 是否添加偏置项 | bool |

CGConv运算操作定义见表92。

表92　CGConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| CGConv | 构建晶体图卷积运算符，根据晶体图中原子的邻接关系以及原子的特征向量，利用图卷积操作来聚合每个原子的特征 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| edge\_dim | 边特征维度 | int |
| aggr | 要使用的聚合方法 | string |
| batch\_norm | 是否使用批量归一化 | bool |
| bias | 是否添加偏置项 | bool |

EdgeConv运算操作定义见表93。

表93　EdgeConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| EdgeConv | 一种处理点云数据的动态图卷积算子，通过构建动态图来描述点之间的连接关系，利用这些边来聚合每个点的邻域信息，并更新点的特征表示 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor  Tuple [tensor, tensor] |
| Attributes | nn | 映射成对串联节点特征的神经网络 | Model |
| in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| batch\_norm | 是否使用批归一化函数 | bool |
| bias | 是否添加偏置项 | bool |
| aggr | 要使用的聚合方法 | string |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |

DynamicEdgeConv运算操作定义见表94。

表94　DynamicEdgeConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DynamicEdgeConv | 一种处理点云数据的动态图卷积算子，通过构建动态图来描述点之间的连接关系，利用这些边来聚合每个点的邻域信息，并更新点的特征表示。图是使用特征空间中的最近邻居动态构建的 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| batch | 批向量，将每个节点分配一个批量 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | nn | 映射成对节点串联特征。 | Model |

表94　DynamicEdgeConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DynamicEdgeConv | 一种处理点云数据的动态图卷积算子，通过构建动态图来描述点之间的连接关系，利用这些边来聚合每个点的邻域信息，并更新点的特征表示。图是使用特征空间中的最近邻居动态构建的 | Attributes | k | 最近邻居数量 | int |
| aggr | 要使用的聚合方法 | string |
| num\_workers | k-NN计算的进程数 | int |

XConv运算操作定义见表95。

表95　XConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| XConv | 构建X变换点上的卷积运算符，对点云中的每个点进行变换，将其转换到局部坐标系中。然后，它利用这些局部坐标系来构建卷积核，从而捕捉点云中点之间的局部关系 | Input | X | 节点特征矩阵 | tensor |
| pos | 点坐标矩阵 | tensor |
| batch | 每一个节点的批次所属 | tensor  Tuple [tensor, tensor] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| kernel\_size | 卷积核的大小，即包括自循环在内的邻居数 | int |
| hidden\_channel | 隐含层点特征维度 | int |
| dilation | 邻居扩散因子 | int |
| bias | 是否添加偏置项 | bool |
| num\_workers | k-NN计算的进程数 | int |

PPFConv运算操作定义见表96。

表96　PPFConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PPFConv | 一种有效且鲁棒的方式来处理3D点云数据，使得神经网络能够更好地理解和利用点云中的局部信息，并在点云匹配任务中取得更好的性能 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| pos | 点坐标矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| normal | 法线矩阵 | tensor  Tuple [tensor, tensor] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | local\_nn | 映射节点特征和相对空间坐标的神经网络 | Model |
| global\_nn | 映射聚合后的节点特征神经网络 | Model |
| add\_self\_loops | 是否添加自环 | bool |

FeatStConv运算操作定义见表97。

表97　FeaStConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FeaStConv | 根据输入的特征信息来动态调整卷积核的权重。这些权重可以根据输入特征的情况来调整，从而使网络能够更好地适应不同形状的特征分布 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |

表97　FeaStConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FeaStConv | 根据输入的特征信息来动态调整卷积核的权重。这些权重可以根据输入特征的情况来调整，从而使网络能够更好地适应不同形状的特征分布 | Attributes | heads | 注意力头的数量 | int |
| add\_self\_loops | 是否添加自环 | bool |
| bias | 是否添加偏置项 | bool |

HypergraphConv运算操作定义见表98。

表98　HypergraphConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HypergraphConv | 通过引入超边权重，用于控制超边对节点特征的影响程度，从而更好地捕捉超图中节点之间的关系 | Input | X | 节点特征矩阵 | tensor |
| hyperedge\_index | 超边索引 | tensor |
| hyperedge\_weight | 超边特征矩阵 | tensor |
| num\_edges | 超边数量 | int |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| use\_attention | 是否使用注意力机制 | bool |
| attention\_mode | 计算注意力机制的模式 | string |
| heads | 注意力头的数量 | int |
| concat | 多头注意力机制进行串联，否则进行平均 | int |
| negative\_slope | 负斜率的LeakyReLU角度 | float |
| dropout | 丢弃率 | float |
| bias | 是否添加偏置项 | bool |

LEConv运算操作定义见表99。

表99　LEConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| LEConv | 在每个节点上对局部邻域进行卷积操作，并通过自适应的结构感知机制加强图的层次表示 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| bias | 是否添加偏置项 | bool |

PNAConv运算操作定义见表100。

表100　PNAConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PNAConv | 通过将节点特征进行多项式变换来捕捉节点自身的信息。利用聚合函数将变换后的节点特征与邻居节点的特征进行结合，以获得节点的新特征表示 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边属性矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| aggr | 聚合函数的标识符集合 | List [string] |
| scalers | 缩放功能标识符集合 | List [string] |
| deg | 训练集中的节点的度数直方图，供缩放器用来进行归一化 | tensor |
| delta | 度正则化有关的因子 | float |

表100　PNAConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PNAConv | 通过将节点特征进行多项式变换来捕捉节点自身的信息。利用聚合函数将变换后的节点特征与邻居节点的特征进行结合，以获得节点的新特征表示 | Attributes | edge\_dim | 边特征维度 | int |
| act | 激活函数名称 | string |
| tower | 塔数 | int |
| pre\_layers | 聚合之前的转换层数 | int |
| post\_layers | 聚合之后的转换层数 | int |
| train\_norm | 正则化参数是否可训练 | bool |
| residual | 是否添加残差连接 | bool |
| dropout | 丢弃率 | float |
| divided\_Input | 输入特征是否应在塔之间划分 | bool |

ClusterGCNConv运算操作定义见表101。

表101　ClusterGCNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ClusterGCNConv | 将图数据分解成多个子图（cluster），然后在每个子图上进行独立的GCN训练，从而减少计算和内存需求 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| diag\_lambda | 对角增强值 | float |
| add\_self\_loops | 是否添加自环 | bool |
| bias | 是否添加偏置项 | bool |

GENConv运算操作定义见表102。

表102　GENConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GENConv | 使用一系列可学习的参数化函数来对节点之间的消息传递进行建模。这些参数化函数可以动态地调整以适应不同的图结构和特征分布 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| size | 邻接矩阵大小 | Tuple [int, int] |
| edge\_weight | 边特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| aggr | 使用的聚合方法 | string |
| t | softmax聚集的初始反向温度。 | float |
| learn\_t | softmax的反向温度是否可学习 | bool |
| expansion | MLP中隐藏层的扩散因子 | int |
| p | 功率均值聚合的初始功率值 | float |
| learn\_p | 功率均值聚合是否为动态可学习 | bool |
| msg\_norm | 是否使用消息归一化 | bool |
| learn\_msg\_scale | 消息归一化的比例因子是否可学习 | bool |
| normalize | 多层感知器的归一化方式 | string |
| bias | 是否添加偏置项 | bool |
| edge\_dim | 边特征维度 | int |
| num\_layers | 多层感知器层数 | int |
| eps | 消息构造函数的epsilon值 | float |

GCNIIConv运算操作定义见表103。

表103　GCNIIConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GCNIIConv | 通过利用节点的自身特征和邻居节点的特征，结合多项式参数化函数来进行信息传递和聚合。从而学习节点特征表示 | Input | X | 节点特征矩阵 | tensor |
| X0 | 初始化特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | channels | 输入/输出特征的维度 | int |
| alpha | 初始残差连接的强度 | float |
| theta | 计算身份映射的强度的超参 | float |
| layer | 在该模块中执行的层数 | int |
| shared\_weights | 共享平滑表示和初始残差连接的权重矩阵 | bool |
| activation | 激活函数名称 | string |
| bias | 是否添加偏置项 | bool |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |
| cached | 是否缓存第一次执行的对称归一化邻接矩阵的结果 | bool |
| normalize | 是否添加自环并应用对称归一化 | bool |
| add\_self\_loops | 是否添加自环 | bool |

WLConv运算操作定义见表104。

表104　WLConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| WLConv | 构建Weisfeiler Lehman运算符，对每个节点及其邻居节点的特征使用WL图同构测试对编码后的图进行处理，最后通过卷积操作将编码后的节点特征进行聚合和更新 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor |

FiLMConv运算操作定义见表105。

表105　FiLMConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FiLMConv | 在传统的图卷积操作中引入了FiLM机制，通过对每个节点的特征进行线性调制，以增强或减弱特定维度的特征表示，从而提高网络对图数据的建模能力和泛化能力 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_type | 边种类矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| num\_relation | 关系的数量 | int |
| nn | 节点特征映射的神经网络 | Model |
| act | 激活函数 | Model |
| aggr | 要使用的聚合方法 | string |

SuperGATConv运算操作定义见表106。

表106　SuperGATConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SuperGATConv | 利用图注意力机制对节点之间的关系进行建模，以确定每个节点与其邻居节点之间的重要性。然后，利用自监督学习的方法来引导网络学习更有意义的节点表示，使得节点表示能够更好地捕捉图的结构和特征 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| neg\_edge\_index | 负向边索引 | tensor  SparseTensor |
| batch | 每一个节点的批次所属 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| heads | 多头注意力的数量 | int |
| concat | 如果是False，对多头注意力进行average操作，如果是True，则进行concat操作 | bool |
| negative\_slope | 负斜率的LeakyReLU角度 | float |
| dropout | 丢弃率 | float |
| bias | 是否添加偏置项 | bool |
| add\_self\_loops | 是否添加自环 | bool |
| attention\_type | 注意力机制的种类 | string |
| neg\_sample\_ratio | 采样的负边数与正边数之比 | float |
| edge\_sample\_ratio | 训练边数中用于训练的样本比例 | float |
| is\_undirected | 输入图是否是无向的。如果未给出，则在执行负采样时将使用输入图自动计算 | bool |

CFConv运算操作定义见表107。

表107　CFConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| CFConv | 使用可学习的滤波器函数来对原子之间的相互作用进行建模，将原子间的相对位置和特征映射到一个连续的函数空间中，这些滤波后的特征表示会根据原子之间的相对位置进行聚合，以捕获分子的局部结构和化学环境 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 输入的边特征 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征维度 | int |
| edge\_dim | 边特征维度 | int |
| hidden\_channels | 隐藏表示的维度 | int |
| out\_channels | 输出表示的维度 | int |

DOTGATConv操作定义见表108。

表108　DOTGATConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DOTGATConv | 构建GAT中应用点积方法计算注意力权重系数的图卷积层 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| get\_attention | 是否返回注意力 | bool |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出表示的维度 | int |
| num\_heads | 多头注意头数 | int |
| bias | 是否带偏置 | bool |
| allow\_zero\_in\_degree | 是否允许入度为0的节点出现 | bool |

RGCNConv运算操作定义见表109。

表109　RGCNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| RGCNConv | 利用每种关系类型的参数化权重来进行节点特征的聚合和更新，从而更好地捕捉图中不同类型的关系信息 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_type | 一维边类型 | tensor |
| norm | 表示边规范值的一维张量 | tensor |
| is\_sorted | 表示输入图的边是否已经按照其类型排序 | bool |
| Output | Y | 输出张量 | tensor |
| Attributes | in\_channels | 输入节点特征的维度 | int |
| out\_channels | 输出节点表示的维度 | int |
| num\_relations | 边关系的数量 | int |
| num\_bases | 基数分解正则化的基数，None为不使用基数正则化方案 | int |
| num\_blocks | 块对角分解正则化的块数，None为不使用块对角正则化方案 | int |
| aggr | 使用的聚合方法 | string |
| root\_weight | 是否将转换后的根节点特征添加到输出中 | bool |
| bias | 是否添加偏置项 | bool |
| activation | 激活函数名称 | string |
| add\_self\_loops | 是否添加自环 | bool |
| dropout | 丢弃率 | float |
| layer\_norm | 是否进行归一化 | bool |

HANConv运算操作定义见表110。

表110　HANConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HANConv | 使用注意力机制和元路径来动态地计算节点之间的关系权重，并利用这些权重对节点特征进行聚合和更新 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息。 | Dict [string, tensor] |
| g | 输入图 | Graph |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| get\_attention | 是否额外返回每种目标节点类型的语义级注意力权重 | bool |
| Output | Y | 输出张量 | Dict [string, tensor] |
| Attributes | in\_channels | 输入特征的维度 | int  Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| metadata | 异质图的元数据，即由一个字符串列表给出的节点类型和由一个字符串三元组列表给出的边类型。 | Tuple [List [string]，List [Tuple [string, string, string]]] |
| heads | 多头注意力的数量 | int |
| negative\_slope | 负斜率的LeakyReLU角度 | float |
| dropout | 丢弃率 | float |

HGTConv运算操作定义见表111。

表111　HGTConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HGTConv | 构建异质图transformer运算符，自注意力机制来动态地计算节点之间的关系权重，并利用这些权重对节点特征进行聚合和更新 | Input | X | 节点特征矩阵 | Dict [string, tensor] |
| g | 输入图 | Graph |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| node\_type | 一维节点类型索引 | tensor |
| edge\_type | 一维边类型索引 | tensor |
| is\_sorted | 表示输入图的边是否已经按照其类型排序 | bool |
| Output | Y | 输出张量 | Dict [string, tensor] |
| Attributes | in\_channels | 输入特征的维度 | int  Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| metadata | 异质图的元数据，即由一个字符串列表给出的节点类型和由一个字符串三元组列表给出的边类型 | Tuple [List [string], List [Tuple [string, string, string]]] |
| heads | 多头注意力的数量 | int |
| group | 用来对不同关系产生的节点嵌入分组的聚合方式 | string |
| num\_ntypes | 节点类型的数量 | int |

表111　HGTConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HGTConv | 构建异质图transformer运算符，自注意力机制来动态地计算节点之间的关系权重，并利用这些权重对节点特征进行聚合和更新 | Attributes | num\_etypes | 边类型的数量 | int |
| dropout | 丢弃率 | float |
| layer\_norm | 是否进行归一化 | bool |

MetaPath2Vec运算操作定义见表112。

表112　MetaPath2Vec运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MetaPath2Vec | 构建元路径随机游走嵌入学习运算符，利用这些关系模式来生成节点序列。然后，它使用基于深度学习的技术，如Word2Vec，来学习节点的嵌入， 以保持嵌入空间中相似节点的相似性 | Input | node\_type | 想要获得嵌入的节点类型 | string |
| batch | 想要获取嵌入的节点的索引 | tensor |
| Output | Y | 输出张量（节点嵌入） | tensor |
| Attributes | edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| g | 输入图 | Graph |
| embedding\_dim | 嵌入向量的大小 | int |
| metapath | 元路径 | List [Tuple [string, string, string]] |
| walk\_length | 游走的长度 | int |
| context\_size | 正例的上下文窗口大小 | int |
| walks\_per\_node | 对于每个节点进行游走的次数 | int |

表112　MetaPath2Vec运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MetaPath2Vec | 构建元路径随机游走嵌入学习运算符，利用这些关系模式来生成节点序列。然后，它使用基于深度学习的技术，如Word2Vec，来学习节点的嵌入， 以保持嵌入空间中相似节点的相似性 | Attributes | num\_negative\_samples | 对于每个正例，负采样的数量 | int |
| num\_nodes\_dict | 节点数量字典，保存每种节点类型的节点数量 | Dict [string, int] |
| sparse | 反向传播时权重矩阵的梯度是否设置为稀疏矩阵 | bool |

HERec运算操作定义见表113。

表113　HERec运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HERec | 利用一个交互式表示学习框架，将用户和商品的隐含表示进行交叉融合，以产生更加准确的用户-商品匹配表示。最后，HERec利用这些匹配表示来进行推荐，以提高推荐系统的性能和准确度 | Input | X | 节点特征矩阵 | Dict [string, tensor] |
| edge\_index\_dict | 边索引 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| g | 输入图 | Graph |
| node\_type | 节点类型 | tensor |
| edge\_type | 边类型 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | embedding\_node\_type | 元路径节点类型 | string |
| embedding\_dim | 嵌入向量的大小 | int |
| metapath | 元路径 | List [Tuple [string, string, string]] |

表113　HERec运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HERec | 利用一个交互式表示学习框架，将用户和商品的隐含表示进行交叉融合，以产生更加准确的用户-商品匹配表示。最后，HERec利用这些匹配表示来进行推荐，以提高推荐系统的性能和准确度 | Attributes | walk\_length | 游走的长度 | int |
| context\_size | 正例的上下文窗口大小 | int |
| walks\_per\_node | 对于每个节点进行游走的次数 | int |
| num\_negative\_samples | 对于每个正例，负采样的数量 | int |

SetTransformerEncoder运算操作定义见表114。

表114　SetTransformerEncoder运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SetTransformerEncoder | 基于注意力机制的置换不变神经网络的编码运算符 | Input | g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| X | 输入特征向量 | tensor |
| Output | Y | 输出特征向量 | tensor |
| Attributes | d\_model | 模型的隐藏维度 | int |
| n\_heads | 注意力头的数目 | int |
| d\_heads | 每个注意力头的隐藏大小 | int |
| d\_ff | FFN层的核大小 | int |
| n\_layers | 模型层数 | int |
| block\_type | 包含SAB模块或ISAB模块 | string |
| m | Induced向量的数目（如果包含ISAB模块） | int |
| dropout | 每个子层的dropout率 | float |
| dropout\_head | 注意力头的dropout率 | float |

SetTransformerDecoder运算操作定义见表115。

表115　SetTransformerDecoder运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SetTransformerDecoder | 基于注意力机制的置换不变神经网络的解码运算符 | Input | g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| X | 输入特征向量 | tensor |
| Output | Y | 输出特征向量 | tensor |
| Attributes | d\_model | 模型的隐藏维度 | int |
| num\_heads | 注意力头的数目 | int |
| d\_head | 每个注意力头的隐藏大小 | int |
| d\_ff | FFN层的核大小 | int |
| n\_layers | 模型层数 | int |
| k | PMA层中种子向量的数目 | int |
| dropout | 每个子层的dropout率 | float |
| dropout\_head | 注意力头的dropout率 | float |

GeomConv运算操作定义见表116。

表116　GeomConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GeomConv | 几何图卷积神经网络，利用图表征空间中的几何信息，提取或“重构”传统聚合算子丢失的信息。输入特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 输出张量（节点特征） | tensor |
| Attributes | in\_channels | 输入特征数量 | int |
| out\_channels | 输出类别 | int |
| hidden\_channels | 隐藏层数量 | int |
| num\_divisions | 子图数量 | int |
| dropout | dropout的比率 | float |
| num\_heads\_layer\_one | 第一层GAT注意力头数 | int |

表116　GeomConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GeomConv | 几何图卷积神经网络，利用图表征空间中的几何信息，提取或“重构”传统聚合算子丢失的信息。输入特征张量和邻接矩阵，输出特征张量 | Attributes | num\_heads\_layer\_two | 第二层GAT注意力头数 | int |
| layer\_one\_ggcn\_merge | 对多子图的聚合操作 | string |
| layer\_one\_channel\_merge | 对多注意力头的聚合操作 | string |
| layer\_two\_ggcn\_merge | 对多子图的聚合操作 | string |
| layer\_two\_channel\_merge | 对多注意力头的聚合操作 | string |

EGNNConv运算操作定义见表117。

表117　EGNNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| EGNNConv | 构建等变图神经网络运算符，使用特征张量、邻接矩阵、节点坐标特征矩阵和边特征矩阵，输出特征张量和节点坐标特征张量 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| coord\_feat | 节点的坐标特征矩阵 | tensor |
| edge\_weight | 边特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| coord\_feat\_out | 节点的坐标特征输出 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| out\_channels | 输出特征的维度 | int |
| edge\_dim | 边特征维度 | int |

TWIRLSConv运算操作定义见表118。

表118　TWIRLSConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TWIRLSConv | 构建基于迭代重加权最小二乘的图卷积运算符，使用特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| prop\_step | 传播步数 | int |
| num\_mlp\_before | 传播前的多层感知机层数 | int |
| num\_mlp\_after | 传播后的多层感知机层数 | int |
| normalize | 多层感知机内的规范化类型 | string |
| precond | 是否使用预处理和归一化拉普拉斯矩阵 | bool |
| alp | 梯度迭代步数超参数 | float |
| lam | 权衡全局信息和局部信息的超参数 | float |
| attention | 是否在传播中添加注意力层 | bool |
| tau | 能量函数参数 | float |
| T | 能量函数参数 | float |
| p | 能量函数参数 | float |
| use\_eta | 是否在注意力的每个维度上添加可学习权重 | bool |
| attn\_bef | 是否在传播之前添加注意力层 | bool |
| dropout | 多层感知机的丢弃率 | float |

表118　TWIRLSConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TWIRLSConv | 构建基于迭代重加权最小二乘的图卷积运算符，使用特征张量和邻接矩阵，输出特征张量 |  | attn\_dropout | 注意力值的丢弃率 | float |
| inp\_dropout | 输入特征的丢弃率 | float |

GDO运算操作定义见表119。

表119　GDO运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GDO | 采用分离提纯的思想，从节点的原始特征中抽离共有特征获得蒸馏增强特征，使用特征张量和邻接矩阵，输出特征张量（节点增强表示嵌入） | Input | X | 节点原始特征矩阵 | tensor  Tuple [tensor, tensor] |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出蒸馏后增强特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| cached | 是否缓存第一次执行的对称归一化邻接矩阵的结果 | bool |
| add\_self\_loops | 是否给输入图添加自环 | bool |

Shift-GCNConv运算操作定义见表120。

表120　Shift-GCNConv运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Shift-GCNConv | 使用移位方法+11卷积进行图卷积运算，使用特征张量和邻接矩阵，输出表示张量 | Input | X | 节点特征矩阵 | tensor  Tuple [tensor, tensor] |
| A | 输入图的邻接矩阵 | tensor |

表120　Shift-GCNConv运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Shift-GCNConv | 使用移位方法+11卷积进行图卷积运算，使用特征张量和邻接矩阵，输出表示张量 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Tuple [int, int] |
| out\_channels | 输出特征的维度 | int |
| global | 是否进行全局移位操作 | bool |
| bias | 是否添加偏置项 | bool |

Shift-GCNConv算法伪代码见表121。

表121　Shift-GCNConv算法伪代码

|  |
| --- |
| Shift-GCNConv算法 |
| 输入：in\_channels, out\_channels, X, A,global，bias |
| 输出：Y |
| if in\_channels != out\_channels: |
| self.down = nn.Sequential(  nn.Conv2d(in\_channels, out\_channels, kernel\_size=1,  W\_init=nn.initializers.XavierNormal(),  b\_init='zeros', data\_format='channels\_first'),  nn.BatchNorm2d(out\_channels, gamma\_init=nn.initializers.Constant(1),  data\_format='channels\_first')  ) |
| else: |
| self.down = lambda x: x |
| index\_array\_in=np.empty(in\_channels).astype(np.int) |
| index\_array\_out=np.empty(out\_channels).astype(np.int) |
| if global: |
| for j in range(in\_channels): |
| index\_array\_in[in\_channels + j] = (in\_channels + j + j\*in\_channels)%(in\_channels) |

表121　Shift-GCNConv算法伪代码（续）

|  |
| --- |
| Shift-GCNConv算法 |
| for j in range(out\_channels): |
| index\_array\_out[out\_channels + j] = (out\_channels + j - j\*out\_channels)%(out\_channels) |
| self.shift\_in = nn.Parameter(torch.from\_numpy(index\_array),requires\_grad=False) |
| self.shift\_out = nn.Parameter(torch.from\_numpy(index\_array),requires\_grad=False) |
| else: |
| A[A==0] = 0 |
| A[A!=0] = 1 |
| partition = np.sum(A[i]) |
| channel\_per\_partition\_in = in\_channels//partition |
| channel\_per\_partition\_out = out\_channels//partition |
| current\_A = A[0] |
| current\_A[0] = 0 |
| neighbors = np.nonzero(current\_A)[0] |
| for j in range(int(partition)): |
| if j == 0: |
| index\_array\_in[int(channels):int(channels + channel\_per\_partition\_in)] = np.arange(channels,channels +  channel\_per\_partition\_in).astype(np.int) |
| index\_array\_out[int(channels):int(channels + channel\_per\_partition\_out)] = np.arange(channels,channels +  channel\_per\_partition\_out).astype(np.int) |
| else: |
| index\_array\_in[int(channels + j\*channel\_per\_partition\_in) :  int(channels + (j+1)\*channel\_per\_partition\_in)] =  np.arange(neighbors[j-1]\*channels +  j\*channel\_per\_partition\_in, neighbors[j-1]\*channels +  (j+1)\*channel\_per\_partition\_in).astype(np.int) |
| index\_array\_out[int(channels + j\*channel\_per\_partition\_out) :  int(channels + (j+1)\*channel\_per\_partition\_out)] =  np.arange(neighbors[j-1]\*channels +  j\*channel\_per\_partition\_out, neighbors[j-1]\*channels +  (j+1)\*channel\_per\_partition\_out).astype(np.int) |
| self.shift\_in = nn.Parameter(torch.from\_numpy(index\_array\_in),requires\_grad=False) |

表121　Shift-GCNConv算法伪代码（续）

|  |
| --- |
| Shift-GCNConv算法 |
| self.shift\_out = nn.Parameter(torch.from\_numpy(index\_array\_out),requires\_grad=False) |
| X = torch.index\_select(X, 1, self.shift\_in) |
| if bias: |
| self.linear\_bias = nn.Parameter(torch.zeros(1, 1, out\_channels, requires\_grad=True, device='cuda'), requires\_grad=True) |
| X = X+self.linear\_bias |
| X = torch.index\_select(X, 1, self.shift\_out) |
| X = X + down(X) |
| Y = relu(X) |
| return Y |

* + 1. 池化算子

图池化算子允许图神经网络处理可变大小的图，并能够在图级别进行有效的信息汇总，用于图分类、图嵌入、图回归等任务。图池化主要分为局部池化和全局池化两种形式：

1. 局部池化：局部池化层通过下采样来粗化图，类似于在卷积神经网络中的池化层；
2. 全局池化：全局池化层，也称为读出层，提供整个图的固定大小表示。全局池化层是排列不变的，使图节点和边的排列顺序的变化不会改变最终输出。

池化算子运算操作定义具体见表122~表143。

ASA\_pool运算操作定义见表122。

表122　ASA\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ASA\_pool | 自适应结构感知池化运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph  BatchGraph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| batch | 每一个节点的批次所属 | tensor |
| Output | Y | 池化后的表示 | tensor |
| edge\_index | 粗化后的边索引 | tensor |

表122　ASA\_pool运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ASA\_pool | 自适应结构感知池化运算符 | Output | edge\_weight | 粗化后的边权重矩阵 | tensor |
| batch | 粗化后的批次向量 | tensor |
| perm | 池化后保留的节点的前Topk节点索引 | tensor |
| Attributes | in\_channels | 每个输入节点类型的特征维度 | int |
| ratio | 图池化比率 | float  int |
| GNN | 使用聚类内部属性的图神经网络层 | Model |
| dropout | 归一化注意力系数的丢弃概率 | float |
| negative\_slope | 负斜率的LeakyReLU角 | float |
| add\_self\_loops | 是否添加自环 | bool |

edge\_pool运算操作定义见表123**。**

表123　edge\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| edge\_pool | 利用计算边分数进行池化的边池化运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| batch | 批量向量，用于将每个节点分配给一个特定批量 | tensor |
| Output | Y | 池化后的表示 | tensor |
| edge\_index | 粗化后的边索引 | tensor |
| batch | 批量向量，用于将每个节点分配给一个特定批量 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |

表123　edge\_pool运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| edge\_pool | 利用计算边分数进行池化的边池化运算符 | Attributes | edge\_score\_method | 用于从原始边分数计算边分数的函数方法 | string |
| dropout | 丢弃率 | float |
| add\_to\_edge\_score | 被添加到边分数上的一个值 | float |

SAG\_pool运算操作定义见表124。

表124　SAG\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SAG\_pool | 通过边分数进行计算的边收缩运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| edge\_weight | 边特征矩阵 | tensor |
| batch | 每一个节点的批次所属 | tensor |
| attn | 当这个参数被提供时，该矩阵会被用来计算注意力分数，而不是使用节点特征矩阵 | tensor |
| Output | Y | 池化后的表示 | tensor |
| edge\_index | 粗化后的边索引 | tensor |
| edge\_weight | 粗化后的边权重矩阵 | tensor |
| batch | 粗化后的批次向量 | tensor |
| perm | 池化后保留的节点的前Topk节点索引 | tensor |
| score[perm] | Topk索引对应分数 | tensor |
| Attributes | in\_channels | 每个输入节点类型的特征维度 | float |
| ratio | 图池比率 | float  int |

表124　SAG\_pool运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SAG\_pool | 通过边分数进行计算的边收缩运算符 | Attributes | GNN | 用于计算投影得分的图神经网络层 | Model |
| min\_score | 最小节点得分 | float |
| multiplier | 在池化后特征被乘以的系数 | float |
| nonlinearity | 使用的非线性函数 | string |

TopK\_pool运算操作定义见表125。

表125　TopK\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TopK\_pool | Top-k池化运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| edge\_weight | 边特征矩阵 | tensor |
| batch | 每一个节点的批次所属 | tensor |
| attn | 当这个参数被提供时，该矩阵会被用来计算注意力分数，而不是使用节点特征矩阵 | tensor |
| Output | Y | 池化后的节点嵌入 | tensor |
| edge\_index | 粗化后的边索引 | tensor |
| edge\_weight | 粗化后的边权重矩阵 | tensor |
| batch | 粗化后的批次向量 | tensor |
| perm | 池化后保留的节点的前Topk节点索引 | tensor |
| score | Topk索引对应分数 | tensor |
| Attributes | in\_channels | 每个输入节点类型的特征维度 | float |

表125　TopK\_pool运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TopK\_pool | Top-k池化运算符 | Attributes | ratio | 图池比率 | float |
| GNN | 用于计算投影得分的图神经网络层 | Model |
| min\_score | 最小节点得分 | float |
| multiplier | 在池化后特征被乘以的系数 | float |
| nonlinearity | 使用的非线性函数名称 | string |

average\_pool运算操作定义见表126**。**

表126　average\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| average\_pool | 根据聚类定义的平均节点特征池化运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| cluster | 每一个节点的所属集群 | tensor |
| Output | g | 输出池化后的图 | Graph |

max\_pool运算操作定义见表127。

表127　max\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| max\_pool | 对节点做最大池化的运算，最终节点特征由同一聚类内所有节点的最大特征定义，节点位置被平均 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| cluster | 每一个节点的所属集群 | tensor |
| Output | g | 输出池化后的图 | Graph |

min\_pool运算操作定义见表128。

表128　min\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| min\_pool | 对节点做最小池化的运算，最终节点特征由同一聚类内所有节点的最小特征定义，节点位置被平均 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| cluster | 每一个节点的所属集群 | tensor |
| Output | g | 输出池化后的图 | Graph |

sum\_edges聚合操作定义见表129。

表129　sum\_edges聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| sum\_edges | 聚合边特征，并按聚合类型“和”生成图级表示 | Input | X | 边特征 | tensor |
| Output | graph\_attr | 图特征 | tensor |

max\_edges聚合操作定义见表130。

表130　max\_edges聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| max\_edges | 聚合边特征，并按聚合类型“最大”生成图级表示 | Input | X | 边特征 | tensor |
| Output | graph\_attr | 图特征 | tensor |

avg\_edges聚合操作定义见表131。

表131　avg\_edges聚合操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| avg\_edges | 聚合边特征，并按聚合类型“平均”生成图级表示 | Input | X | 边特征 | tensor |
| Output | graph\_attr | 图特征 | tensor |

节点特征topk\_nodes操作定义见表132。

表132　节点特征topk\_nodes操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| topk\_nodes | 通过节点特征上的图top-k返回图级表示 | Input | X | 输入的节点特征 | tensor |
| k | topk的节点数量 | int |
| sortby | 所依照的排序基础。如果为null或-1，则按所有特征独立排序 | int |
| Output | graph\_attr | 图特征 | tensor |

边特征topk\_edges操作定义见表133。

表133　边特征topk\_edges操作

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| topk\_edges | 通过边特征上的图top-k返回图级表示 | Input | X | 输入的边特征 | tensor |
| k | topk的边数量 | int |
| sortby | 所依照的排序基础。如果为null或-1，则按所有特征独立排序 | int |
| Output | graph\_attr | 图特征 | tensor |

global\_add\_pool运算操作定义见表134。

表134　global\_add\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| global\_add\_pool | 在整个节点维度上添加节点特征的运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| batch | 每一个节点的批次所属 | tensor |
| size | 批次数目 | int |
| Output | Y | 输出表示 | tensor |

global\_max\_pool运算操作定义见表135。

表135　global\_max\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| global\_max\_pool | 通过获取整个节点维度上的通道级最大值进行最大池化的运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| batch | 每一个节点的批次所属 | tensor |
| size | 批次数目 | int |
| Output | Y | 输出表示 | tensor |

global\_mean\_pool运算操作定义见表136。

表136　global\_mean\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| global\_mean\_pool | 通过对整个节点维度上的节点特征求平均进行均值池化的运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| batch | 每一个节点的批次所属 | tensor |
| size | 批次数目 | int |
| Output | Y | 输出表示 | tensor |

global\_sort\_pool运算操作定义见表137。

表137　global\_sort\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| global\_sort\_pool | 通过对节点特征进行排序进行全局池化的运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| batch | 每一个节点的批次所属 | tensor |
| Output | Y | 输出表示 | tensor |
| Attributes | k | 每个图要保留的节点数 | int |

global\_min\_pool运算操作定义见表138。

表138　global\_min\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| global\_min\_pool | 通过获取整个节点维度上的通道级最小值进行最小池化的运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| batch | 每一个节点的批次所属 | tensor |
| Output | Y | 输出表示 | tensor |

global\_attention运算操作定义见表139。

表139　global\_attention运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| global\_attention\_pool | 全局soft attention池化运算符 | Input | X | 输入节点特征 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| get\_attention | 是否获取注意力值 | bool |
| Output | Y | 输出表示 | tensor |
| Attributes | gate\_nn | 通过对节点特征进行映射来计算注意力得分的神经网络 | Model |
| feat\_nn | 将特征结合之前对每个特征进行处理的神经网络 | Model |

Set2Set运算操作定义见表140。

表140　Set2Set运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Set2Set | 一种基于内容的迭代注意力机制下的全局池化运算符 | Input | X | 节点特征矩阵 | tensor |
| index | 用于应用聚合操作的元素的索引，index和ptr必须至少定义一个 | tensor |

表140　Set2Set运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Set2Set | 一种基于内容的迭代注意力机制下的全局池化运算符 | Input | ptr | 如果提供，将基于CSR表示的排序输入来计算聚合。index和ptr必须至少定义一个。 | tensor |
| dim\_size | 输出节点类型的特征维度 | int |
| dim | 在哪个维度上进行聚合 | int |
| max\_num\_elements | 单个聚合组内的最大元素数 | int |
| g | 输入图 | Graph |
| Output | Y | 输出表示 | tensor |
| Attributes | in\_channels | 输入节点类型的特征维度 | int |
| num\_iters | 迭代次数 | int |
| num\_layers | 模型层数 | int |

WeightAndSum运算操作定义见表141。

表141　WeightAndSum运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| WeightAndSum | 用于计算原子的重要性权重并执行加权总和的运算符 | Input | X | 节点特征矩阵 | tensor |
| g | 用于并行处理多个分子的图对象 | Graph |
| Output | Y | 各分子的特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |

dense\_diff\_pool运算操作定义见表142。

表142　dense\_diff\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dense\_diff\_pool | 可微分池化运算符 | Input | X | 节点特征矩阵 | tensor |
| adj | 邻接矩阵 | tensor |
| s | 分配张量 | tensor |

表142　dense\_diff\_pool运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dense\_diff\_pool | 可微分池化运算符 | Input | mask | mask矩阵 | tensor |
| normalize | 链接预测损失是否会被 adj.numel() 除 | bool |
| Output | Y | 输出节点特征矩阵 | tensor |
| Y\_adj | 输出邻接矩阵 | tensor |
| link\_loss | 节点预测损失 | tensor |
| ent\_loss | 正交损失 | tensor |

dense\_mincut\_pool运算操作定义见表143。

表143　dense\_mincut\_pool运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dense\_mincut\_pool | 最小切割池化运算符 | Input | X | 输入张量 | tensor |
| adj | 邻接矩阵 | tensor |
| s | 分配张量 | tensor |
| mask | mask矩阵 | tensor |
| temp | softmax的温度参数 | float |
| Output | Y | 输出节点特征矩阵 | tensor |
| Y\_adj | 输出邻接矩阵 | tensor |
| link\_loss | Mincut损失 | tensor |
| ent\_loss | 正交损失 | tensor |

* + 1. 归一化算子

归一化算子用以调整和规范化数据的分布，提高模型的训练效率、稳定性和性能，主要包含节点特征归一化和消息归一化。节点特征归一化确保不同节点特征的值在相似的范围内；消息归一化平衡不同节点传递的信息量，提高学习效率和模型稳定性。具体定义见表144~表151。

HeteroBatchNorm运算操作定义见表144。

表144　HeteroBatchNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HeteroBatchNorm | 按照批归一化的方式对异质图上的特征进行归一化，即每种点或边类型单独归一化 | Input | X | 节点特征矩阵 | tensor |
| type\_vec | 节点类型矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| num\_types | 类型的数量 | int |
| eps | 加到归一化分母上的值，用于保证稳定性 | float |
| momentum | 用于控制均值和方差更新速度 | float |
| affine | 是否加入可学习参数 | bool |
| track\_running\_stats | 是否记录运行中的均值与方差 | bool |

HeteroLayerNorm运算操作定义见表145。

表145　HeteroLayerNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HeteroLayerNorm | 按照层归一化的方式对异质图上的特征进行归一化，即每种点或边类型单独归一化 | Input | X | 节点特征矩阵 | tensor |
| type\_vec | 节点类型矩阵 | tensor |
| type\_ptr | 定义节点类型的边界 | tensor  List [int] |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| num\_types | 类型的数量 | int |
| eps | 加到归一化分母上的值，用于保证稳定性 | float |
| affine | 是否加入可学习参数 | bool |
| mode | 默认为“node”：每个节点特征被当做一个元素；若选择“graph”:每张图被当做一个元素 | string |

GraphSizeNorm运算操作定义见表146。

表146　GraphSizeNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphSizeNorm | 按照Graph Size Normalization的方式，根据节点数对每张图中的特征进行归一化 | Input | X | 节点特征矩阵 | tensor |
| batch | 描述每个节点从属的batch | tensor |
| batch\_size | batch的大小 | int |
| Output | Y | 输出特征表示 | tensor |

PairNorm运算操作定义见表147。

表147　PairNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PairNorm | 按照Pair Norm的方式，对节点特征进行归一化 | Input | X | 节点特征矩阵 | tensor |
| batch | 描述每个节点从属的batch | tensor |
| batch\_size | batch的大小 | int |
| Output | Y | 输出特征表示 | tensor |
| Attributes | scale | 归一化规模参数 | float |
| scale\_individually | 是否独立计算归一化的scale | bool |
| eps | 加到归一化分母上的值，用于保证稳定性 | float |

MessageNorm运算操作定义见表148。

表148　MessageNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MessageNorm | 按照Message Normalization的方式，对聚合的消息进行归一化 | Input | X | 节点特征矩阵 | tensor |
| msg | 消息矩阵 | tensor |
| p | 用于归一化的归一化常量 | float |
| Output | Y | 输出特征表示 | tensor |
| Attributes | learn\_scale | 默认为False；若设置为True，则规模常量会调整为可学习参数 | bool |

DiffGroupNorm运算操作定义见表149。

表149　DiffGroupNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DiffGroupNorm | 按照Diff Group Normalization的方式，通过一个分组为基础的软聚类节点划分来归一化节点特征 | Input | X | 节点特征矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| groups | 分组的数量 | int |
| lamda | 用于平衡输入表示和归一化表示的参数 | float |
| eps | 加到归一化分母上的值，用于保证稳定性 | float |
| momentum | 用于控制均值和方差更新速度 | float |
| affine | 是否加入可学习参数 | bool |
| track\_running\_stats | 是否记录运行中的均值与方差 | bool |

MeanSubtractionNorm运算操作定义见表150。

表150　MeanSubtractionNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MeanSubtractionNorm | 结合层归一化的方式，以减去邻居节点特征均值的方式对特征进行归一化 | Input | X | 节点特征矩阵 | tensor |
| batch | 描述每个节点从属的batch | tensor |
| dim\_size | batch的大小 | int |
| Output | Y | 输出特征表示 | tensor |

GraphNorm运算操作定义见表151。

表151　GraphNorm运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphNorm | 按照Graph Normalization的方式，对每张图中的特征进行归一化 | Input | X | 节点特征矩阵 | tensor |
| batch | 描述每个节点从属的batch | tensor |
| batch\_size | batch的大小 | int |

表151　GraphNorm运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphNorm | 按照Graph Normalization的方式，对每张图中的特征进行归一化 | Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| eps | 加到归一化分母上的值，用于保证稳定性 | float |

* 1. 点级模型
     1. 概述

点级模型是应用于点级任务的模型。点级图神经网络模型主要利用图的连通关系以及特征矩阵，将存在关系的节点特征进行聚合并得到新的节点表征。根据其节点聚合的范围，将图神经网络分为一阶聚合、多阶聚合以及任意聚合三种类型。

* + 1. 一阶聚合

一阶聚合是指在聚合过程中仅仅包含一阶邻居，将与节点直接相连的其他节点的表征作为信息聚合，而后与节点的原始表征进行组合得到新的节点表征。

一阶聚合模型定义见表152~表168。

GCN模型对原始的邻接矩阵添加自环，而后计算其拉普拉斯矩阵并进行归一化，最后通过得到的矩阵进行消息传递。模型定义见表152。

表152　GCN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GCN | 根据添加自环以及归一化后的拉普拉斯矩阵进行消息传递 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |

GAT模型在对节点信息进行聚合时，首先计算其各邻居节点对其的注意力分数，然后根据得到的注意力分数加权求和以进行信息的聚合。模型定义见表153。

表153　GAT模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GAT | 利用注意力机制进行带权重的消息传递 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| attention\_weights | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| heads | 注意力头数量 | int |

GraphSAGE模型对节点的邻居进行采样，根据采样得到的节点进行消息传递。模型定义见表154。

表154　GraphSAGE模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphSAGE | 采样部分一阶邻居进行聚合操作 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| size | 采样邻接矩阵大小 | int |

GIN模型通过构造单射函数，使GNN获得了与WL-test相当的性能。它通过串联（CONCAT）的方式，将节点在所有迭代步骤中的表示聚合起来，以此来计算节点的最终表征。模型定义见表155。

表155　GIN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GIN | 设计单射函数以及聚合过程使其达到WL-test的表达能力 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| aggregation\_type | 聚合方法 | string |

GGNN模型借鉴类似于LSTM的门控机制，以实现图神经网络中节点与邻居特征的聚合。模型定义见表156。

表156　GGNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GGNN | 构建门控图卷积模型，使用特征矩阵和邻接矩阵，输出节点表征 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| aetype | 边的类型 | string |

ResGatedgraph模型将残差连接引入到门控图卷积神经网络中，以实现性能的增益。定义见表157。

表157　ResGatedgraph模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ResGatedgraph | 构建残差门控图卷积模型，使用特征矩阵和邻接矩阵，输出特征表征 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |

GINE模型将边的特征与残差连接引入到图卷积神经网络中，以实现性能的增益。定义见表158。

表158　GINE模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GINE | 改进的图同构模型，使用特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边特征矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| size | 邻接矩阵的大小 | Tuple [int, int] |

GaAN模型在GAT的基础上，对不同的注意力头添加不同的权重参数以获取更优的性能。定义见表159。

表159　GaAN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GaAN | 对不同的注意力头添加不同的权重系数 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| heads | 注意力头数量 | int |

AM-GCN模型根据节点特征构建一个KNN图。然后通过两个特有图卷积网络，分别在原始拓扑图和构建属性图上提取各自节点特征。再通过一个共同图卷积网络，在原始拓扑图和构建属性图上分别提取节点特征求平均得到共同节点特征。最后采用注意力机制对三种节点特征进行自适应聚合，从而达到自适应提取网络拓扑信息以及节点特征信息的目的。模型定义见表160。

表160　AM-GCN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| AM-GCN | 自适应多通道图卷积网络 | Input | X | 节点特征矩阵 | tensor |
| edge\_index\_s | 结构图边索引 | tensor  SparseTensor |
| edge\_index\_f | 特征图边索引 | tensor  SparseTensor |
| Output | Y | 输出张量 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| dropout | 特征丢弃率 | float |
| hidden\_channels1 | 第一隐藏层特征的维度 | int |
| hidden\_channels2 | 第二隐藏层特征的维度 | int |

AM-GCN算法伪代码见表161。

表161　AM-GCN算法伪代码

|  |
| --- |
| AM-GCN算法 |
| 输入：in\_channels, out\_channels, hidden\_channels1, hidden\_channels2, dropout, X, edge\_index\_s, edge\_index\_f |
| 输出：Y |
| SGCN1 = Sequential([GCNConv(in\_channels,hidden\_channels1),  Dropout(dropout),  GCNConv(hidden\_channels2,hidden\_channels1)]) |
| SGCN2 = Sequential([GCNConv(in\_channels,hidden\_channels1),  Dropout(dropout),  GCNConv(hidden\_channels2,hidden\_channels1)]) |
| CGCN = Sequential([GCNConv(in\_channels,hidden\_channels1),  Dropout(dropout),  GCNConv(hidden\_channels2,hidden\_channels1)]) |
| attention = Attention(hidden\_channels2) |
| MLP = Sequential([Linear(hidden\_channels2, out\_channels),LogSoftmax(dim=1)]) |
| emb1 = SGCN1(X, edge\_index\_s) |
| emb2 = SGCN2(X, edge\_index\_f) |
| com1 = CGCN(X, edge\_index\_s) |
| com2 = CGCN(X, edge\_index\_f) |
| Xcom = (com1 + com2) / 2 |
| emb = stack([emb1, emb2, Xcom], dim=1) |
| emb = attention(emb) |
| Y = MLP(emb) |
| return Y |

FAGCN模型使用一个低通滤波器以及一个高通滤波器，用于分离节点特征中的低频信号以及高频信号。使用注意力机制来学习低频信号以及高频信号的比例并实现频率自适应地聚合低频信号以及高频信号。模型定义见表162。

表162　FAGCN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FAGCN | 频率自适应图卷积网络模型 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出张量 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| dropout | 特征丢弃率 | float |
| hidden\_channels | 隐藏层特征的维度 | int |
| layer\_num | FAConv的层数 | int |
| eps | FAConv的epsilon值 | float |

FAGCN算法伪代码见表163。

表163　FAGCN算法伪代码

|  |
| --- |
| FAGCN算法 |
| 输入：in\_channels, out\_channels, hidden\_channels, dropout, X, edge\_index, layer\_num |
| 输出：Y |
| layers = [] |
| lin1 = Linear(in\_channels, hidden\_channels) |
| for i in range(layer\_num): |
| layers.append(FAConv(hidden\_channels, eps, dropout)) |
| lin2 = Linear(hidden\_channels, out\_channels) |
| h = dropout(X, p=dropout) |
| h = relu(lin1(h)) |
| h = dropout(X, p=dropout) |
| raw = h |

表163　FAGCN算法伪代码（续）

|  |
| --- |
| FAGCN算法 |
| for i in range(num\_layers): |
| h = layers[i](h) |
| h = eps\*raw + h |
| Y = log\_softmax(lin2(h), dim=1) |
| return Y |

GeomGCN以集合神经网络获取图表征，保证连续空间的平稳性、局部性和组合型。其运算操作定义见表164。

表164　GeomGCN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GeomGCN | 几何图卷积神经网络，利用图表征空间中的几何信息，提取或“重构”传统聚合算子丢失的信息。输入特征张量和邻接矩阵，输出特征张量 | Input | X | 节点特征矩阵 | tensor |
| g | 图 | Graph |
| Output | Y | 输出张量（节点特征） | tensor |
| Attributes | num\_input\_features | 输入特征数量 | int |
| num\_output\_classes | 输出类别 | int |
| num\_hidden | 隐藏层数量 | int |
| num\_divisions | 子图数量 | int |
| dropout\_rate | dropout的比率 | float |
| num\_heads\_layer\_one | 第一层GAT注意力头数 | int |
| num\_heads\_layer\_two | 第二层GAT注意力头数 | int |
| layer\_one\_ggcn\_merge | 对多子图的聚合操作 | string |
| layer\_one\_channel\_merge | 对多注意力头的聚合操作 | string |
| layer\_two\_ggcn\_merge | 对多子图的聚合操作 | string |
| layer\_two\_channel\_merge | 对多注意力头的聚合操作 | string |

HGAT基于一种HIN框架的异质图注意力网络模型来对短文本进行建模。其具体运算操作见表165。

表165　HGAT模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HGAT | 基于双层注意力机制的半监督短文本分类模型 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息。 | Dict [string, tensor] |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| num\_nodes\_dict | 节点数量字典，用于储存每个类型的节点的数量 | Dict [string, int] |
| Output | out\_dict | 输出概率值 | Dict [string, tensor] |
| Attributes | in\_channels | 输入特征的维度 | int |
| Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| metadata | 异质图的元数据，即由一个字符串列表给出的节点类型和由一个字符串三元组列表给出的边类型。 | Tuple [List [string]，List [Tuple [string, string, string]]] |

HGAT算法伪代码见表166。

表166　HGAT算法伪代码

|  |
| --- |
| HGAT算法 |
| 输入：x\_dict, edge\_index\_dict, num\_nodes\_dict |
| 输出：out\_dict |

表166　HGAT算法伪代码（续）

|  |
| --- |
| HGAT算法 |
| Type\_Attention\_Value\_dict = {} |
| for edge\_type, edge\_index in edge\_index\_dict.items(): |
| Type\_Attention\_Value\_dict[edge\_type] = [] |
| for edge\_type, edge\_index in edge\_index\_dict.items(): |
| src\_type, \_, dst\_type = edge\_type |
| src = edge\_index[0,:] |
| dst = edge\_index[1,:] |
| h\_l = self.Linear\_dict\_l[src\_type](x\_dict[src\_type]) |
| h\_r = self.Linear\_dict\_r[dst\_type](x\_dict[dst\_type]) |
| h\_l = tlx.gather(h\_l,src) |
| h\_r = tlx.gather(h\_r,dst) |
| Type\_Attention\_Value = self.leakyReLu(h\_l + h\_r) |
| Type\_Attention\_Value\_dict[edge\_type].append(Type\_Attention\_Value) |
| x\_value, edge\_index, edge\_value = to\_homograph(x\_dict,edge\_index\_dict,num\_nodes\_dict, Type\_Attention\_Value\_dict) |
| alpha = segment\_softmax(edge\_value,edge\_index[1,:],num\_segments=None) |
| alpha = self.dropout(alpha) |
| src = edge\_index[0,:] |
| dst = edge\_index[1,:] |
| h\_l = self.Linear\_l(x\_value) |
| h\_r = self.Linear\_r(x\_value) |
| h\_l = tlx.gather(h\_l,src) |
| h\_r = tlx.gather(h\_r,dst) |
| Node\_Attention\_Value = (h\_l + h\_r)\*alpha |
| beta = segment\_softmax(Node\_Attention\_Value,edge\_index[1,:],num\_segments=None) |
| out = unsorted\_segment\_sum(beta\*tlx.gather(x\_value,edge\_index[0,:]),edge\_index[0,:]) |
| out\_dict = to\_heterograph(self.dropout(out),edge\_index,num\_nodes\_dict) |
| return out\_dict |

GCNII在原有的图卷积神经网络上增加初始连接和恒等变换两个模块以缓解过平滑现象。模型定义见表167。

表167　GCNII模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GCNII | GCNII利用 增加初始连接和恒等变换两个模块以缓解GCN中的过平滑现象 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| Output | Y | 输出张量 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| dropout | 特征丢弃率 | float |
| hidden\_channels | 隐藏层特征的维度 | int |
| num\_layers | GCNIIConv的层数 | int |
| alpha | GCNIIConv中初始连接比例 | float |
| lamda | GCNIIConv中恒等映射比例 | float |

GNNII算法伪代码见表168。

表168　GCNII算法伪代码

|  |
| --- |
| GCNII算法 |
| 输入：in\_channels, out\_channels, hidden\_channels, dropout, X, edge\_index, nlayer, alpha, lamda |
| 输出：Y |
| layers = [] |
| lin1 = Linear(in\_channels, hidden\_channels) |

表168　GCNII算法伪代码（续）

|  |
| --- |
| GCNII算法 |
| for i in range(layer\_num): |
| layers.append(GCNIIConv(hidden\_channels, hidden\_channels)) |
| lin2 = Linear(hidden\_channels, out\_channels) |
| h = dropout(X, p=dropout) |
| h = relu(lin1(h)) |
| h0 = h |
| for i in range(nlayer): |
| h = dropout(X, p=dropout) |
| beta = log(lamda / (i+1) + 1) |
| h = layers[i](X, edge\_index, alpha, h0, beta) |
| h = relu(h) |
| h = dropout(h, p=dropout) |
| Y = log\_softmax(self.lin2(h), dim=1) |
| return Y |

* + 1. 多阶聚合

多阶聚合是在聚合过程中不只包含一阶邻居，根据其拓扑结构，选择更高阶的邻居进行聚合。

多阶聚合模型定义见表169~表184。

SGC模型将邻接矩阵进行K次幂，获取了节点的K阶邻居信息，而后进行消息传递。模型定义见表169。

表169　SGC模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SGC | 去除GCN中的非线性层，仅保留k层消息传递模型以进行信息聚合 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引，与g选一 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |

表169　SGC模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SGC | 去除GCN中的非线性层，仅保留k层消息传递模型以进行信息聚合 | Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 跳数（即取到K阶邻居） | int |

APPNP模型使用personalized PageRank算法进行随机游走，得到节点的邻域信息以进行消息传递以及信息聚合。模型定义见表170。

表170　APPNP模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| APPNP | 利用随机游走得到节点邻域以进行消息传递 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 随机游走迭代次数 | int |
| alpha | 重启传送概率参数 | float |

GPRGNN模型使用generalized PageRank方法使图神经网络适用于异配图，先计算K轮消息传递，而后使用K层特征的线性组合作为表征，线性组合的参数通过学习获得。模型定义见表171。

表171　GPRGNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GPRGNN | 使用K轮消息传递表征的线性组合得到最终表征 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 消息传递的轮数 | int |

ChebNet模型中使用切比雪夫多项式代替谱域的卷积核，谱域的卷积核的取值是与特征值相关的函数，以避免显式的特征值分解，然后来用切比雪夫多项式来逼近这个函数。模型定义见表172。

表172　ChebNet模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ChebNet | 使用切比雪夫多项式近似卷积核 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 切比雪夫滤波器大小 | int |

JKnet模型使用节点进行聚合时，极大受到所在子结构影响（若为中心节点如微博大V，则很少几步就可以得到大量邻居信息），不同的节点应该具有不同的聚合步数。该模型先计算聚合1-K步的所有结果，而后自适应的选择需要的步数进行组合。模型定义见表173。

表173　JKnet模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| JKnet | 不同节点采用自适应的聚合步数 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 最大聚合步数 | int |

DCNN模型使用diffusion对每一个节点采用H个hop的矩阵进行表示，每一个hop表示其邻近信息，以更好得获取局部信息。模型定义见表174。

表174　DCNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DCNN | 利用不同hop的信息进行聚合得到节点表征 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| K | Hop数 | int |

Line模型使用节点与其共同邻居的相似性表示两节点的相似性，以构造二阶的相邻关系。模型定义见表175。

表175　Line模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Line | 利用节点与其共同邻居的相似性表示两节点的相似性 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |

HPN模型通过语义传播机制和语义融合机制来缓解语义混淆。语义传播机制通过赋予节点的局部语义适当权重，改善节点层级聚合过程，使节点嵌入即使在更深的HeteroGNN架构中也能保持区分度。语义融合机制致力于学习元路径的重要性，并对其进行合理融合。模型定义见表176。

表176　HPN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HPN | 异质图传播网络，改善元路径和注意力机制在消息传递过程中的聚合过程，以获得更鲁棒的表示 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息 | Dict [string, tensor] |
| g | 输入图 | HeteroGraph |

表176　HPN模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HPN | 异质图传播网络，改善元路径和注意力机制在消息传递过程中的聚合过程，以获得更鲁棒的表示 | Input | edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| Output | Y | 输出张量 | Dict [string, tensor] |
| Attributes | in\_channels | 输入特征的维度 | int  Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| heads | 多头注意力的数量 | int |

HPN模型伪代码见表177。

表177　HPN算法伪代码

|  |
| --- |
| HPN算法 |
| 输入：in\_channels, out\_channels, metadata, iter\_K, alpha, negative\_slope, drop\_rate, x\_dict, edge\_index\_dict, num\_nodes\_dict |
| 输出：out\_dict |
| for edge\_type in metadata[1]: |
| src\_type, \_, dst\_type = edge\_type |
| edge\_type = '\_\_'.join(edge\_type) |
| self.appnp\_dict[edge\_type] = APPNPConv(in\_channels[src\_type],  out\_channels,  iter\_K,  alpha,  drop\_rate) |

表177　HPN算法伪代码（续）

|  |
| --- |
| HPN算法 |
| out\_dict = {} |
| for node\_type, \_ in x\_dict.items(): |
| out\_dict[node\_type] = [] |
| for edge\_type, edge\_index in edge\_index\_dict.items(): |
| src\_type, \_, dst\_type = edge\_type |
| edge\_type = '\_\_'.join(edge\_type) |
| out = self.appnp\_dict[edge\_type](x\_dict[src\_type],  edge\_index,  num\_nodes\_dict[dst\_type]) |
| out = relu(out) |
| out\_dict[dst\_type].append(out) |
| for node\_type, outs in out\_dict.items() |
| outs = stack(outs) |
| w = mean(linear(out\_channels, 1)(outs, dim=1)) |
| beta = softmax(w, dim=0) |
| beta = beta.unsqueeze(-1) |
| out\_dict[node\_type] = sum(beta \* outs, dim=0) |
| return out\_dict |

GAMLP模型通过recursive attention或JK attention，以学习的方式为各个节点提供个性化的针对K轮消息传递结果的聚合权重。模型定义见表178。

表178　GAMLP模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GAMLP | 为不同节点学习个性化的线性组合K轮消息传递结果的权重 | Input | X | 节点特征矩阵 | tensor |
| y\_train | 独热编码的节点标签矩阵，仅含训练集节点信息 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |

表178　GAMLP模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GAMLP | 为不同节点学习个性化的线性组合K轮消息传递结果的权重 | Input | g | 输入图，与edge\_index选一 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 消息传递的轮数 | int |
|  | 节点标签分支的权重 | float |
| attention\_type | 0表示使用Recursive，1表示使用JK | bool |

GNN-LF/HF是以统一框架理解和分析不同GNN，开发两个考虑可调节图核的新颖目标函数，分别显示低通或高通滤波能力。模型定义见表179。

表179　GNN-LF/HF模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GNN\_LF/HF | 多组件图的卷积协同过滤模型 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | X | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |

GNN-LF/HF伪代码见表180。

表180　GNN-LF/HF算法伪代码

|  |
| --- |
| GNN-LF/HF算法 |
| 输入：in\_channels, out\_channels, g, alpha, beta, mu, x, niter |
| 输出：Y |
| layer\_inner = self.act\_fn(self.fcs[0](self.dropout(x))) |
| for fc in self.fcs[1:-1]: |
| layer\_inner = self.act\_fn(fc(layer\_inner)) |
| local\_logits = self.fcs[-1](dropout(layer\_inner)) |
| propagation = LFExact(graph.adj\_matrix, alpha, mu) |
| propagation = LFPowerIteration(g.adj\_matrix, alpha, mu) |
| propagation = HFExact(g.adj\_matrix, alpha, beta) |
| propagation = HFPowerIteration(g.adj\_matrix, alpha, beta) |
| final\_logits = self.propagation(local\_logits, idx) |
| Return log\_softmax(final\_logits, dim=-1) |

HeCo是利用对比学习的自监督方式训练异质图神经网络，采用交叉视图对比机制，从两个视图中提取正、负嵌入，使两个视图能够协作监督，学习更高级别的节点嵌入。模型定义见表181。

表181　HeCo模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HeCo | 融合对比学习的自监督异质图神经网络 | Input | X | 存储每一种节点类型的节点特征信息。 | List [tensor] |
| pos | 正样本 | tensor |
| mps | 元路径邻接矩阵 | List [tensor] |
| nei\_index | 邻居索引列表 | List [tensor] |
| Output | loss | 输出损失值 | tensor |
| Attributes | hidden\_channels | 隐藏层维度 | int |
| feats\_dim\_list | 各类型特征维度 | List [int] |
| feat\_drop | 特征丢弃率 | float |
| sample\_rate | 采样率 | float |
| nei\_num | 邻居类型数量 | int |

表181　HeCo模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HeCo | 融合对比学习的自监督异质图神经网络 | Attributes | tau | 温度值 | float |
| lam | lamda值 | float |
| attn\_drop | attention聚合都i去哦率 | float |
| p | 元路径数量 | int |

HeCo算法伪代码见表182。

表182　HeCo算法伪代码

|  |
| --- |
| HeCo算法 |
| 输入：feats, pos, mps, nei\_index |
| 输出：loss |
| h\_all = [] |
| for i in range(len(feats)): |
| h\_all.append(F.elu(self.feat\_drop(self.fc\_list[i](feats[i])))) |
| z\_mp = self.mp(h\_all[0], mps) |
| z\_sc = self.sc(h\_all, nei\_index) |
| loss = self.contrast(z\_mp, z\_sc, pos) |
| return loss |
| embeds = [] |
| for i in range(self.P): |
| mps[i] = mps[i] |
| embeds.append(self.node\_level[i](h, mps[i])) |
| z\_mp = self.att(embeds) |
| return z\_mp |
| embeds = [] |
| for i in range(self.nei\_num): |
| sele\_nei = [] |
| sample\_num = self.sample\_rate[i] |
| for per\_node\_nei in nei\_index[i]: |

表182　HeCo算法伪代码

|  |
| --- |
| HeCo算法 |
| if len(per\_node\_nei) >= sample\_num: |
| select\_one = torch.tensor(np.random.choice(per\_node\_nei,  sample\_num, replace=False))[np.newaxis] |
| else: |
| select\_one = torch.tensor(np.random.choice(per\_node\_nei,  sample\_num, replace=True))[np.newaxis] |
| sele\_nei.append(select\_one) |
| sele\_nei = torch.cat(sele\_nei, dim=0).cuda() |
| one\_type\_emb = F.elu(self.intra[i](sele\_nei, nei\_h[i + 1], nei\_h[0])) |
| embeds.append(one\_type\_emb) |
| z\_mc = self.inter(embeds) |
| return z\_mc |
| z\_proj\_mp = self.proj(z\_mp) |
| z\_proj\_sc = self.proj(z\_sc) |
| matrix\_mp2sc = self.sim(z\_proj\_mp, z\_proj\_sc) |
| matrix\_sc2mp = matrix\_mp2sc.t() |
| matrix\_mp2sc = matrix\_mp2sc/(torch.sum(matrix\_mp2sc, dim=1).view(-1, 1) + 1e-8) |
| lori\_mp = -torch.log(matrix\_mp2sc.mul(pos).sum(dim=-1)).mean() |
| matrix\_sc2mp = matrix\_sc2mp / (torch.sum(matrix\_sc2mp, dim=1).view(-1, 1) + 1e-8) |
| lori\_sc = -torch.log(matrix\_sc2mp.mul(pos).sum(dim=-1)).mean() |
| return self.lam \* lori\_mp + (1 - self.lam) \* lori\_sc |

HACUD是引入基于层次注意的套现用户检测模型，以解决套现欺诈问题。模型定义见表183。

表183　HACUD模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HACUD | 基于分层注意力机制的 Cash-out用户检测模型 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息 | Dict [string, tensor] |
| g | 输入图 | HeteroGraph |

表183　HACUD模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HACUD | 基于分层注意力机制的 Cash-out用户检测模型 | Input | edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| metadata | 异质图的元数据，即由一个字符串列表给出的节点类型和由一个字符串三元组列表给出的边类型 | Tuple [List [string], List [Tuple [string, string, string]]] |
| Output | p | 输出概率值 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int  Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |

HACUD算法伪代码见表184。

表184　HACUD算法伪代码

|  |
| --- |
| HACUD算法 |
| 输入：hidden\_channels, metadata, node\_type, node\_id , x\_dict, edge\_index\_dict |
| 输出：e\_u |
| aggregated\_features = {} |
| for meta\_path, edge\_index in edge\_index\_dict.items(): |
| \_, edge\_types = metadata |
| for edge\_type in edge\_types: |
| src\_type, \_, dst\_type = edge\_type |
| if src\_type == node\_type: |

表184　HACUD算法伪代码（续）

|  |
| --- |
| HACUD算法 |
| if edge\_type in edge\_index\_dict: |
| edge\_indices = edge\_index\_dict[edge\_type] |
| node\_edge\_indices = (edge\_indices[0] == node\_id).nonzero(as\_tuple=True)[0] |
| if len(node\_edge\_indices) == 0: |
| continue |
| neighbors = edge\_indices[1][node\_edge\_indices] |
| neighbor\_features = x\_dict[dst\_type][neighbors] |
| aggregated\_features[edge\_type] = neighbor\_features.sum(dim=0) |
| x\_features = x\_dict[node\_type][node\_id] |
| h\_u = Linear(x\_features.shape(0), hidden\_channels)(x\_features) + Parameter(hidden\_channels) |
| fusion\_features = {} |
| for type, feature in aggregated\_features: |
| h\_rou = Linear(feature.shape(0), hidden\_channels)(feature) + Parameter(hidden\_channels) |
| aggregated\_features[type] = h\_rou |
| cat\_feature = cat(h\_u, h\_rou) |
| fusion\_feature = relu(Linear(2\*hidden\_channels, hidden\_channels)(cat\_feature) + Parameter(hidden\_channels)) |
| fusion\_features[type] = fusion\_feature |
| att\_features = {} |
| for type, feature in fusion\_features: |
| cat\_feature = cat(h\_u, feature) |
| v\_rou = relu(Linear(hidden\_channels, hidden\_channels)(cat\_feature) + Parameter(hidden\_channels)) |
| alpha\_rou = relu(Linear(hidden\_channels, hidden\_channels)(v\_rou) + Parameter(hidden\_channels)) |
| alpha\_rou = softmax(alpha\_rou, dim=0) |
| att\_features[type] = alpha\_rou \* feature |
| attention\_vectors = Parameter((len(att\_features), hidden\_channels)) |
| f\_concat = torch.cat(list(att\_features.values()), dim=1) |
| attention\_scores = torch.matmul(attention\_vectors, f\_concat) |

表184　HACUD算法伪代码（续）

|  |
| --- |
| HACUD算法 |
| beta = F.softmax(attention\_scores, dim=1) |
| e\_u = stack([beta[p] \* f\_pu\_tilde\_dict[meta\_path] for p, meta\_path in enumerate(f\_pu\_tilde\_dict)]).sum(dim=0) |
| return e\_u |

* + 1. 任意聚合

任意聚合是无视图的连通关系，任意的节点之间可进行信息的传递，根据节点本身的特性进行聚合操作，无视拓扑结构或者仅使用拓扑结构进行辅助。

任意聚合模型定义见表185~表190。

GTN模型是用transformer进行全局聚合，以拉普拉斯特征向量进行位置编码。模型定义见表185。

表185　GTN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GTN | 利用transformer进行消息聚合，并采用拉普拉斯作为特征编码 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引，与g而选一 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| attention\_weights | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| heads | 注意力头数量 | int |

Graphormer模型是用transformer进行全局的消息传递以及聚合，在计算注意力分数时，加入两个节点之间的一些信息如最短路径、边特征等，以进行空间位置编码来影响注意力权重。模型定义见表186。

表186　Graphormer模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Graphormer | 利用transformer进行消息聚合，并采用几种不同的空间编码，如最短路径等 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| attention\_weights | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| heads | 注意力头数量 | int |
| position\_encoding | 位置编码方式 | string |

GraphTrans模型是在标准GNN层上加入transformer，用Bert中的[CLS]token，引入readout机制，抛弃位置编码以保持图的置换不变性。模型定义见表187。

表187　GraphTrans模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphTrans | 在标准GNN层后添加transformer并利用[CLS]进行全局聚合 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| attention\_weights | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| heads | 注意力头数量 | int |

SAT模型是使用结构提取器专门用于抓取子图的表征以计算注意力分数。允许模型在特征聚合过程中融入图结构的信息，增强模型对图结构的敏感度和表征能力。模型定义见表188。

表188　SAT模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SAT | Structure-Aware Transformer通过在每个节点提取以其为中心的子图表示，然后计算注意力，从而将结构信息整合到原始自注意力中 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| attention\_weights | 注意力权重 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| heads | 注意力头数量 | int |
| k\_hop | 抽取子图的Hop数 | int |
| struct\_extractor | 提取子图表征的方式 | string |

GraphGPS模型是广泛使用，且适用不同规模大小的图的Graph Transformer框架。模型定义见表189。

表189　GraphGPS模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphGPS | 构建可以广泛使用，且适用不同规模大小的图的Graph Tr-ansformer框架 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| attention\_weights | 注意力权重 | tensor |

表189　GraphGPS模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphGPS | 构建可以广泛使用，且适用不同规模大小的图的Graph Tr-ansformer框架 | Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| heads | 注意力头数量 | int |
| type\_mpnn | 局部注意力计算方式 | string |
| type\_transformer | 全局注意力计算方式 | string |
| position\_encoding | 位置编码方式 | string |

GloGNN模型是以聚合图中全局节点的信息生成节点的嵌入，用优化问题的闭式解求出全局聚合中发挥作用的系数矩阵。模型定义见表190。

表190　GloGNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GloGNN | 利用系数矩阵对全局节点的信息进行聚合 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| k\_hop | 系数矩阵近似的邻接矩阵的阶数 | int |

* 1. 边级模型
     1. 概述

边级模型是以边为主体对图进行建模和处理的模型。根据边表示信息的编码方式，将边级模型分为3类：基于节点的边级模型、基于子图的边级模型和基于路径的边级模型。

* + 1. 基于节点的边级模型

基于节点表征的边级模型是通过聚合目标边的首尾节点表征作为输入，构建出边级表征的模型。模型首先通过图神经网络对节点的领域信息做聚合，获取节点的向量表征，再将边上的首尾节点的表征聚合，输入到链接预测器中，得到边级的表征模型。

基于节点的边级模型定义见表191~表198。

GraphSage Link Predictor 用于同质图的链路预测，基于GraphSage模型计算节点表征，根据点积（以下简称“dot”）或mlp融合头尾节点表征，得到边概率。模型定义见表191。

表191　GraphSage Link Predictor模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraphSage Link Predictor | 基于GraphSage计算节点表征，根据dot或mlp融合头尾节点表征，得到边概率 | Input | X | 节点特征 | tensor |
| g | 输入图 | Graph |
| query\_edge | 待预测边列表 | List [string, string] |
| Output | edge\_prob | 边概率 | tensor |
| edge\_label | 边标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| link\_combinator | 特征融合方式 | string |

R-GCN Link Predictor是接受异质边作为输入，基于R-GCN模型计算节点表征，根据dot或mlp融合头尾节点表征，得到边概率。模型定义见表192。

表192　R-GCN Link Predictor模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| R-GCN Link Predictor | 基于R-GCN计算节点表征，根据dot或mlp融合头尾节点表征，得到边概率 | Input | node\_feature | 节点特征 | tensor |
| g | 输入图 | graph |
| query\_edge | 待预测边列表 | List [string, string] |
| Output | edge\_prob | 边概率 | tensor |
| edge\_label | 边标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| link\_combinator | 特征融合方式 | string |

MCCF模型是多组件图卷积协同过滤方法，包含两个模块：分解器和组合器。前者首先分解“用户-物品”图中的边，以识别可能引起购买关系的潜在组件；后者自动重新组合这些潜在组件，从而获得用于预测“用户-物品”的边级表征。模型定义见表193。

表193　MCCF模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MCCF | 多组件图的卷积协同过滤模型 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| Output | Y | 输出特征表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| heads | 多头注意力的数量 | int |

MCCF算法伪代码见表194。

表194　MCCF算法伪代码

|  |
| --- |
| MCCF算法 |
| 输入：in\_channels, out\_channels, nodes\_u, nodes\_i, droprate |
| 输出：Y |
| nodes\_u\_embed = self.u\_embed(nodes\_u, nodes\_i) |
| nodes\_i\_embed = self.i\_embed(nodes\_u, nodes\_i) |
| x\_u = relu(self.u\_bn(self.u\_layer1(nodes\_u\_embed)), inplace = True) |
| x\_u = dropout(x\_u, p = droprate) |
| x\_u = self.u\_layer2(x\_u) |
| x\_i = relu(self.i\_bn(self.i\_layer1(nodes\_i\_embed)), inplace = True) |
| x\_i = dropout(x\_i, p = droprate) |
| x\_i = self.i\_layer2(x\_i) |

表194　MCCF算法伪代码（续）

|  |
| --- |
| MCCF算法 |
| x\_ui = cat((x\_u, x\_i), dim = 1) |
| x = relu(self.ui\_bn(self.ui\_layer1(x\_ui)), inplace = True) |
| Y = dropout(x, p = droprate) |

MEIRec是利用元路径来指导不同步骤邻居的选择，设计一个异质的GNN，以获得“用户-查询”的边级表征。模型定义见表195。

表195　MEIRec模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MEIRec | 基于元路径的异质图神经意图推荐模型 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息 | Dict [string, tensor] |
| g | 输入图 | HeteroGraph |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| Output | Y | 输出张量 | Dict [string, tensor] |
| Attributes | in\_channels | 输入特征的维度 | int  Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |

MEIRec算法伪代码见表196。

表196　MEIRec算法伪代码

|  |
| --- |
| MEIRec算法 |
| 输入：in\_channels, out\_channels, g, num\_nodes\_dict, metapath |
| 输出：out\_dict |
| out = {} |
| one\_step\_dict = g.sample\_neighbors(num\_nodes\_dict, -1) |
| one\_step\_dict = one\_step\_dict.edges()[0] |
| two\_step\_dict = g.sample\_neighbors(one\_step\_dict , -1) |
| two\_step\_dict = two\_step\_dict.edges()[0] |
| for meta\_path in metapath: |
| out[meta\_path] = mean(two\_step\_dict[meta\_path]) |
| out[meta\_path] = LSTM(in\_channels, out\_channels, out[meta\_path]) |
| out\_dict = Aggregate(out) |
| return out\_dict |

HERec是一种基于异质网络嵌入的基于HIN的推荐方法，通过基于元路径的随机游走策略，以生成有意义的节点序列用于网络嵌入。学习到的节点表征首先通过一组融合函数进行变换，然后集成到扩展矩阵分解（MF）模型中。模型定义见表197。

表197　HERec模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HERec | 基于元路径的异质图推荐模型 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息 | Dict [string, tensor] |
| g | 输入图 | HeteroGraph |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |

表197　HERec模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| HERec | 基于元路径的异质图推荐模型 | Output | Y | 输出张量 | Dict [string, tensor] |
| Attributes | in\_channels | 输入特征的维度 | int  Dict [string, int] |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |

HERec算法伪代码见表198。

表198　HERec算法伪代码

|  |
| --- |
| HERec算法 |
| 输入：in\_channels, out\_channels, metadata, x\_dict, edge\_index\_dict, edge\_type, w, M, b |
| 输出：out\_dict |
| outs = {} |
| for edge\_type in metadata: |
| node\_sequence = RandomWalk(x\_dict, edge\_index\_dict, edge\_type) |
| outs[edge\_type] = Node2Vec(node\_sequence) |
| for edge\_type, out in outs: |
| out\_dict = sum(w[edge\_type] \* M \* out + b, dim=0) |
| return out\_dict |

* + 1. 基于子图的边级模型

基于子图的边级模型，是通过构造子图、提取特征、子图嵌入、链接推理和预测等步骤，利用图形中节点和边之间的局部结构信息，预测节点之间的关系。

基于子图的边级模型见表199~表200。

SEAL是应用于同质图的边级模型，提取每个目标边(u,v)周围K阶范围内的封闭子图，应用图级GNN来分类判断子图标签对应于边存在或类别。模型定义见表199。

表199　SEAL模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SEAL | 基于K阶子图表征的同质图边级模型 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息 | Dict [string, tensor] |
| g | 输入图 | HeteroGraph |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| query\_edge | 目标边 | List [string, string] |
| Output | Y | 目标边的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| gcn\_type | GCN编码器(GCN/GraphSage) | string |
| k-hop | 子图抽取的阶数，默认为2 | int |
| nbr\_num | 各阶聚合的邻居数量 | string |
| sampler | 邻居采样方式 | string |

GraIL是针对异质图/知识图谱的边级模型，提取目标边(u,v)周围K阶范围的子图，应用R-GCN作为图级GNN判断边的存在或类别。GraIL的封闭子图不包括只是一个目标节点的邻居但不是另一个目标节点的邻居的节点。模型定义见表200。

表200　GraIL模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GraIL | 基于子图推理的异质图边级模型 | Input | X\_dict | 节点特征字典，用于存储每一种节点类型的节点特征信息。 | Dict [string, tensor] |
| g | 输入图 | HeteroGraph |
| edge\_index\_dict | 边索引字典 | Dict [Tuple [string, string, string], tensor]  Dict [Tuple [string, string, string], SparseTensor] |
| query\_edge | 目标边 | List [string, string] |
| Output | Y | 目标边的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| k-hop | 子图抽取的阶数，默认为 2 | int |
| nbr\_num | 各阶聚合的邻居数量 | string |
| sampler | 邻居采样方式 | string |

* + 1. 基于路径的边级模型

基于路径的边级模型是利用目标边首尾节点间的路径作为信息编码对象，通过路径传播的信息聚合方式，利用路径上的所有节点和边信息进行高效编码后学习其表征。

基于路径的边级模型定义见表201~表203。

NBFNet是将Bellman-Ford算法神经网络化，结合路径传播与图神经网络算子，应用于同质图和异质图链路预测。模型定义见表201。

表201　NBFNet模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| NBFNet Link Predictor | 将Bellman-Ford算法神经网络化，结合路径传播与图神经网络算子，可应用于同质图和异质图链路预测 | Input | X | 节点特征 | tensor |
| g | 输入图 | Graph |
| query\_edge | 待预测边列表 | List [string, string] |
| query\_edge\_type | 待预测边类型 | int |
| Output | edge\_prob | 边概率 | tensor |
| edge\_label | 边标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| num\_relation | 默认10 | int |
| message\_func | 默认distmult | string |
| aggregate\_func | 默认pna | string |
| symmetric | 是否计算对称边，默认False | bool |

PAGNN是通过利用广播和聚合操作，对目标边的两个关联节点之间的所有交互（即路径）和邻域信息进行建模，并基于这些信息生成边级的表征用于链路预测。模型定义见表202。

表202　PAGNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PAGNN | 根据pagnn算法进行边级别子图预测 | Input | g | 输入图数据 | Graph |
| egde\_index | 待预测的边集合 | tenor |
| batch\_size | 每一批次目标节点数目 | int |
| Output | Y | 边属于各个类别标签的概率 | tensor |
| Attribute | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| layer\_num | pagnn层数 | int |

PAGNN算法伪代码见表203。

表203　PAGNN算法伪代码

|  |
| --- |
| PAGNN算法 |
| 输入：in\_channels, out\_channels, hidden\_channels, X, edge\_index, layer\_num |
| 输出：Y |
| G’= enclosing\_subgraph\_extract(edge\_index) |
| h = Linear(in\_channels, hidden\_channels)(X) |
| N = {u} |
| for i in range(layer\_num): |
| N’ = {p | (q, p) G’, q N} |
| for p in N’: |
| h = LSTM(ATT\_OP(h, N’)) |
| N = N’ |
| h’ = Linear(in\_channels, hidden\_channels)(X) |
| r = [h’,h] |
| for i in range(layer\_num): |
| r = ATT\_OP(r, N’) |
| s = concat(r, edge\_index) |
| Y = log\_softmax(Linear(hidden\_channels, out\_channels)(h), dim=1) |
| return Y |

* 1. 图级模型
     1. 概述

图级模型是用于图级任务的模型。图级模型根据任务方式划分为6类，分别为图分类、图回归、图聚类、图匹配和图生成模型。

* + 1. 图分类

图神经网络的图分类是将多图数据中的每一个单图分配到预定义的类别中，单图所对应类别可以是一个或是多个。图级模型通过在图上进行消息传递来学习节点表示，并根据节点特征通过池化等方式学习到图表示，基于学习到的图表示来计算单图的分类概率。

图分类模型见表204~表205。

DiffPool的是通过提供一个可微模块对图节点进行分层池化，构建深度、多层的GNN模型。模型定义见表204。

表204　DiffPool模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DiffPool | 能够对图节点进行分层池化的可微模块K轮消息传递结果 | Input | X | 节点特征矩阵 | tensor |
| y\_train | 节点标签矩阵， | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 图级标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| K | 消息传递的轮数 | int |

MPSN模型是基于代数拓扑中的单纯复形概念扩展图神经网络中的邻域特征聚合概念。该模型将原始图提升为2-复形（包含图中的所有三角结构），基于单纯形的边界邻接、联合边界邻接、下邻接、上邻接这四种单纯复形中的“邻域”概念进行特征聚合，通过读出函数获得最终结果。模型定义见表205。

表205　MPSN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MPSN | 基于代数拓扑中的单纯复形概念扩展图神经网络中的邻域特征聚合概念。首先将原始图提升为2-复形（包含图中的所有三角结构），接着基于单纯形的边界邻接、联合边界邻接、下邻接、上邻接这四种单纯复形中的“邻域”概念进行特征聚合，最后通过读出函数获得最终结果 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边特征矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 图级标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| n\_layers | 隐藏层层数 | int |
| read\_out | 读出函数选择 | string |

* + 1. 图回归

图神经网络中的图回归是通过学习图结构中包含节点的特征、相邻节点的信息以及边的信息的图表征，预测该图所对应的连续性目标值或属性。与图分类不同，图回归的目标是预测数值型的图属性，而不是离散的类别。

图回归模型见表206~表208。

MPNN模型是消息传递和读出两阶段模型框架。对于图中任意节点，消息传递考虑所有与该点相连的节点，基于节点特征与边特征共同计算消息向量，求和后与原有特征向量结合得出新的节点特征向量，通过读出函数得出整个图的特征表示。MPNN为图级一阶模型提供了统一的抽象形式。模型定义见表206。

表206　MPNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MPNN | 通过消息传递和读出两阶段实现图级任务学习。为图级一阶模型提供了统一的抽象形式 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边特征矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 图级表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| n\_layers | 隐藏层层数 | int |
| message\_size | 信息函数输出维度 | int |
| type | 任务类型 | string |

D-MPNN模型通过一个有向图卷积网络对分子性质进行预测。模型定义见表207。

表207　D-MPNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| D-MPNN | 一个对分子性质进行预测的有向图卷积网络 | Input | X | 节点特征矩阵 | tensor |
| edge\_weight | 边特征矩阵 | tensor |
| y\_train | 分子性质的标签矩阵，仅含训练集节点信息 | tensor |

表207　D-MPNN模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| D-MPNN | 一个对分子性质进行预测的有向图卷积网络 | Input | edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | Y | 分子性质的预测值 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 消息传递的轮数 | int |
| dropout\_r | 丢弃比率 | float |

PPGN模型是一种简单的2阶张量网络架构，基于边特征变换、矩阵乘法、堆叠构建基础网络模块，每层包含两个特征变换操作，对图同构问题具有3-WL判别能力。模型定义见表208。

表208　PPGN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PPGN | 基于边特征变换、矩阵乘法、堆叠构建基础网络模块，每层包含两个特征变换操作。对图同构问题具有3-WL判别能力 | Input | edge\_feature | 边特征张量 | tensor |
| Output | Y | 图级表示 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels\_1 | 隐藏层特征的维度1 | int |
| hidden\_channels\_2 | 隐藏层特征的维度2 | int |
| n\_layers | 隐藏层层数 | int |
| type | 任务类型 | string |

* + 1. 图聚类

图神经网络中的图聚类是指通过识别具有相似拓扑结构、相似节点属性分布或其他关联性的图数据，从而将每个图划分到预先定义的类别中的任务。图神经网络利用节点的属性、图的拓扑结构，通过学习图的表征并用于计算属于不同类别的概率或分数。

DMoN是通过聚类质量模块度度量搭配坍缩正则化使得优化目标更易实现，应对现实图数据中的聚类结构恢复问题的监督池化方法。模型定义见表209。

表209　DMoN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DMoN | 学习一种启发性无监督池化算法 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | F | 聚类表征 | tensor |
| C | 聚类分配矩阵 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| n\_clusters | 模型内的聚类个数 | int |
| collapse\_regularization | 坍塌正则化权重 | float |
| dropout | 丢弃率 | float |
| do\_unpooling | 1表示使用反池化，0表示不使用反池化 | bool |

* + 1. 图匹配

图神经网络中的图匹配是指通过图神经网络对图中的节点和边进行表征学习，基于节点和边的表征同时考虑节点相似性和边相似性，在两个或多个图之间建立节点有意义的结构对应关系。

图匹配模型见表210~表211。

GMN的是以两张图作为输入，通过交叉图注意力匹配机制，联合计算两张图之间的相似性得分。与。模型定义见表210。

表210　GMN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GMN | 基于交叉图注意力匹配机制的图匹配模块K轮消息传递结果 | Input | X | 节点特征矩阵 | tensor |
| y\_train | 独热编码的节点标签矩阵，仅含训练集节点信息 | tensor |

表210　GMN模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GMN | 基于交叉图注意力匹配机制的图匹配模块K轮消息传递结果 | Input | edge\_index\_1 | 边索引 | tensor  SparseTensor |
| edge\_weight\_1 | 边权重矩阵 | tensor |
| g\_1 | 输入图 | Graph |
| edge\_index\_2 | 边索引 | tensor  SparseTensor |
| edge\_weight\_2 | 边权重矩阵 | tensor |
| g\_2 | 输入图，与二edge\_index选一 | Graph |
| Output | Y | 图匹配度 | float |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 消息传递的轮数 | int |
| similarity | 相似度计算方式， dot\_product，euclidean\_distance等 | str |

FGNN模型是使用张量结构对图进行表示，引入folklore graph layer操作实现对图特征的置换等变性变换，在最后一层加入置换不变性函数。模型定义见表211。

表211　FGNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FGNN | 引入folklore graph layer操作构建具有置换不变性的图神经网络，在图同构判别问题上具有比MPNN更强的表达能力 | Input | X | 节点特征矩阵 | tensor |
| edge\_index\_1 | 边索引 | tensor  SparseTensor |
| g\_1 | 输入图 | Graph |
| edge\_index\_2 | 边索引 | tensor  SparseTensor |

表211　FGNN模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FGNN | 引入folklore graph layer操作构建具有置换不变性的图神经网络，在图同构判别问题上具有比MPNN更强的表达能力 | Input | g\_2 | 输入图，与二edge\_index选一 | Graph |
| Output | Y | 图匹配度 | float |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| n\_layers | 隐藏层层数 | int |

* + 1. 图生成

图神经网络中的图生成是基于给定的任意数量节点和边的图样本，图神经网络从中学习图样本的分布，并从此分布中抽取新的图。根据生成图的大小，图生成任务可以被分成两类：图节点数量固定的图生成和图节点数量可变的图生成。

图生成模型见表212~表213。

MolGAN模型是基于生成对抗思想的分子图生成框架。随机产生一个高斯噪声后送入一个生成器产生一个邻近张量A和节点特征矩阵X。其中A的形状为(N,N,Y)，其中Y表示边的类型数；X的形状为(N,T)，其中T表示每个节点（原子）的类型数。(A,X)共同刻画了一个分子图结构，将其送入由R-GCN构成的判别器中进行监督训练。模型定义见表212。

表212　MolGAN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| MolGAN | 基于生成对抗思想的分子图生成网络架构，判别器采用了Relational-GCN | Input | z | 随机噪声向量 | tensor |
| Output | A | 图邻接张量 | tensor |
| X | 图节点特征矩阵 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| n\_layers | 隐藏层层数 | int |
| n\_nodes | 输出图节点个数 | int |
| edge\_dim | 边特征维度 | int |

GRANs模型将图生成过程划分成多个时间块，采用注意力机制计算新加入图结构的节点与已存在节点间产生连边的概率。模型定义见表213。

表213　GRANs模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GRANs | 该模型将图生成过程划分成多个时间块，采用注意力机制计算新加入图结构的节点与已存在节点间产生连边的概率 | Input | X | 节点特征矩阵 | tensor |
| y\_train | 节点之间的连接情况的标签，仅含训练集节点信息 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g1 | 输入图 | Graph |
| Output | g2 | 基于g1生成的图 | Graph |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| B | 每次图生成的块大小 | int |
|  | 允许的最大图尺寸 | int |
|  | 节点生成顺序 | List [int] |

1. 图神经网络压缩和加速
   1. 图数据压缩
      1. 概述

图数据压缩是对图形数据进行压缩，可以减少图数据存储和传输开销，提高图数据处理的效率和可扩展性。图数据压缩可以归类为图拓扑结构压缩和图特征数据压缩。具体定义见表214~表217。

* + 1. 图拓扑结构压缩

图拓扑结构压缩是将图的拓扑结构表示方式转化为更紧凑、更高效的形式。对图的拓扑结构进行压缩，可以减小图拓扑数据的存储空间占用并提高处理效率。

Cluster-GCN是一种图拓扑结构压缩的方法，Cluster-GCN利用图聚类结构改进训练算法，可以降低内存占用并提高计算效率。模型定义见表214。

表214　Cluster-GCN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Cluster-GCN | 利用图聚类结构改进训练算法，从而改善内存使用和提高计算效率 | Input | X | 节点特征矩阵 | tensor |
| Y | 节点标签矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Module | 输入初始化模型 | Model |
| Output | Y | 类（标签）矩阵 | tensor |
| H | 预测矩阵及传送集 | tensor |
| Module | 输出训练后的模型 | Model |
| Attributes | dropout\_r | 丢弃比率 | float |
| clustering-method | 聚类方法 | string |
| cluster-number | 聚类数量 | int |

WebGraph是一种图拓扑结构压缩的方法，通过充分利用图拓扑数据的相似性和局部性特点改进图拓扑结构存储方式，可以降低存储占用并提高计算效率。模型定义见表215。

表215　WebGraph模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| WebGraph | 利用图拓扑数据的相似性和局部性特点改进图拓扑结构存储，从而降低存储占用并提高计算效率 | Input | is\_compress | True时表示压缩操作；Flase时表示解压操作； | bool |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |

* + 1. 特征数据压缩

图特征数据压缩是指对图数据中节点、边或全图的特征数据进行压缩，以减小特征数据的空间占用并提高图神经网络处理特征数据的计算效率的方法。图数据的节点和边特征能够包含多个数值，如节点的特征向量、边的特征向量等。特征数据压缩通过使用合适的压缩方法和算法，将向量或数组的属性表示方式转化为更紧凑的形式，以减少数据的存储空间。

VQ-GNN是一个图特征数据压缩框架，VQ-GNN扩展了基于卷积的GNN，使用矢量量化(VQ)压缩图特征数据。模型定义见表216。

表216　VQ-GNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| VQ-GNN | 为不同节点学习个性化的线性组合K轮消息传递结果的权重 | Input | X | 节点特征矩阵 | tensor |
| R | 码字分配矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| edge\_weight | 边权重矩阵 | tensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征维度 | int |
| K | 消息传递的轮数 | int |
| C | 卷积矩阵 | tensor |

VQ-GNN算法伪代码见表217。

表217　VQ-GNN算法伪代码

|  |
| --- |
| VQ-GNN算法 |
| 输入：X |
| 输出：Y |
| For l in range(L): |
| initialize GNN learnable parameters and and |
| For indices sampled from : |
| Load the mini-batch features , labels |
| For |

表217　VQ-GNN算法伪代码（续）

|  |
| --- |
| VQ-GNN算法 |
| Compute the approximate message passing weight matrix using , , |
| Estimate next layer’s features with and feature codewords |
| For |
| Estimate lower layer’s gradients and |
| For |
| Update the concatenated codewords = |

* + 1. 图采样
       1. 概述

根据图采样的层次不同，图采样方法分为三类：

1. 节点采样：节点采样对单个节点的局部邻居采样。节点采样为图中的每个节点选择固定数量的邻居以计算该节点的特征。此类方法允许模型在考虑每个节点的局部结构的同时，降低由于考虑全部邻居带来的计算负担；
2. 层级采样：层级采样在GNN的每一层对多个节点的邻居进行采样。与节点采样相比，此类方法能够在保证采样效率的同时避免邻居数量的指数级增长问题。层级采样的节点数量与层数成线性关系，从而降低整体的内存开销，提高整体计算速度，可以在计算资源受限的情况下，有效地处理深层的GNN模型；
3. 子图采样：子图采样在原图中采样得到一个包含选定节点和边的子图，并在这个子图上进行全图式GNN的训练。此类方法可以将大规模图分割成多个小的、更易于处理的子图，可以减少计算负载，同时保持图的全局特性。
   * + 1. 节点采样

节点采样是对目标节点的邻居进行采样以减少输入图数据量，可以降低内存消耗，并加速模型训练。此类方法实现流程和细节为：

1. 输入：包括节点特征、邻接矩阵、边特征等信息的图数据，选定若干目标节点；
2. 使用采样策略对邻居进行采样；
3. 输出：采样后的图数据，各节点在原图中的节点序号，目标节点在输出图中的序号。

节点采样的具体定义见表218~表220。

node\_sampler运算操作定义见表218。

表218　node\_sampler运算操作定义

| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| --- | --- | --- | --- | --- | --- |
| node\_sampler | 根据起始节点，对输入的图数据进行采样 | Input | g | 输入图数据 | Graph  HeteroGraph |

表218　node\_sampler运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| node\_sampler | 根据起始节点，对输入的图数据进行采样 | Input | target\_nodes | 目标节点的编号 | tensor |
| num\_neighbors | 每层邻居节点数 | List [int]  Dict [Tuple [string, string, string], List [int]] |
| batch\_size | 每一批次目标节点数目 | int |
| shuffle | 是否每轮采样采用随机洗牌策略 | bool |
| drop\_last | 是否丢弃最后一个不完整的批次 | bool |
| Output | blocks | 输出子图列表 | List [Graph]  List [HeteroGraph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

graphsage\_sampler运算操作定义见表219。

表219　graphsage\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| graphsage\_sampler | 按照graphsage的方式进行邻居采样 | Input | g | 输入图数据 | Graph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_neighbors | 每层邻居节点数 | List [int] |
| batch\_size | 每一批次目标节点数目 | int |
| Output | blocks | 输出图列表 | List [Graph] |
| sample\_index | 输出图在原图的节点序号 | tensor |
| node\_index | 起始节点的序号 | tensor |
| neighbors | 每层邻居数 | List [int] |

pinsage\_sampler运算操作定义见表220。

表220　pinsage\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| pinsage\_sampler | 根据起始节点，依据pinsage算法对输入的图数据进行采样 | Input | g | 输入图数据 | HeteroGraph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_neighbors | 每层邻居节点数 | List [int]  Dict [Tuple [string, string, string], List [int]] |
| node\_type | 目标节点的类型 | string |
| other\_type | 其他节点类型 | string |
| num\_traversals | 单次进行基于元路径随机游走的最大遍历次数 | int |
| termination\_prob | 每次基于元路径遍历后的终止概率 | float |
| num\_random\_walks | 对每一个给定节点的随机游走次数 | int |
| batch\_size | 每一批次目标节点数目 | int |
| shuffle | 是否每轮采样采用随机洗牌策略 | bool |
| drop\_last | 是否丢弃最后一个不完整的批次 | bool |
| Output | block | 输出图列表 | List [HeteroGraph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

* + - 1. 层级采样

层级采样是在GNN的每一层采样固定数量的邻居节点。层级采样可以解决随着网络深度增加，邻居数量指数式地增长的问题，降低内存消耗并加速模型训练。

四种具体层级采样运算操作见表221~表224。

layer\_sampler运算操作定义见表221。

表221　layer\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| layer\_sampler | 根据起始节点，对输入的图数据进行层级采样 | Input | g | 输入图数据 | Graph  HeteroGraph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_samples | 每层采样节点数 | List [int]  Dict [Tuple [string, string, string], List [int]] |
| batch\_size | 每一批次目标节点数目 | int |
| shuffle | 是否每轮采样采用随机洗牌策略 | bool |
| drop\_last | 是否丢弃最后一个不完整的批次 | bool |
| Output | block | 输出图 | List [Graph]  List [HeteroGraph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

hgt\_sampler运算操作定义见表222。

表222　hgt\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| hgt\_sampler | 异质图按层级采样的方式进行邻居采样 | Input | g | 输入图数据 | HeteroGraph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_samples | 每层采样节点数 | Dict [Tuple [string, string, string], List [int]] |

表222　hgt\_sampler运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| hgt\_sampler | 异质图按层级采样的方式进行邻居采样 | Input | batch\_size | 每一批次目标节点数目 | int |
| Output | graphs | 输出图 | List [HeteroGraph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

fastgcn\_sampler运算操作定义见表223。

表223　fastgcn\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| fastgcn\_sampler | 按照FastGCN的方式进行邻居采样 | Input | g | 输入图数据 | Graph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_samples | 每层采样节点数 | List [int] |
| batch\_size | 每一批次目标节点数目 | int |
| Output | block | 输出图列表 | List [graph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

ladies\_sampler运算操作定义见表224。

表224　ladies\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ladies\_sampler | 按照LADIES的方式进行邻居采样 | Input | g | 输入图数据 | Graph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_samples | 每层采样节点数 | List [int] |
| batch\_size | 每一批次目标节点数目 | int |
| Output | block | 输出图列表 | List [Graph] |

表224　ladies\_sampler运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ladies\_sampler | 按照LADIES的方式进行邻居采样 | Output | sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

* + - 1. 子图采样

子图采样是对整张图数据进行采样得到若干子图，可以降低内存消耗并加速模型训练。

此类方法实现流程为：

1. 输入节点特征、邻接矩阵、边特征等信息的图数据，子图数量、子图节点最大数量；
2. 使用随机游走等方式采样子图，并根据子图数量、子图节点最大数量等限制子图的规模；
3. 输出子图集合，子图各节点在原图中的节点序号。

五种具体子图采样运算操作见表225~表229。

子图采样运算操作定义见表225。

表225　subgraph\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| subgraph\_sampler | 根据输入图数据进行子图采样 | Input | X | 输入图数据 | Graph |
| target\_nodes | 目标节点的编号 | tensor |
| num\_neighbors | 每层邻居节点数 | List [int]  Dict [Tuple [string, string, string], List [int]] |
| batch\_size | 每一批次目标节点数目 | int |
| shuffle | 是否每轮采样采用随机洗牌策略 | bool |
| drop\_last | 是否丢弃最后一个不完整的批次 | bool |
| Output | block | 输出图列表 | List [Graph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

clustergcn\_sampler运算操作定义见表226。

表226　clustergcn\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| clustergcn \_sampler | 根根clustergcn算法进行子图采样 | Input | g | 输入图数据 | Graph |
| num\_cluster | 聚类个数 | int |
| clustering\_method | 聚类算法 | string |
| batch\_size | 每一批次目标节点数目 | int |
| Output | block | 输出图列表 | List [Graph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

saint\_sampler运算操作定义见表227。

表227　saint\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| saint\_sampler | 根据SAINT算法进行子图采样 | Input | g | 输入图数据 | Graph |
| saint\_method | 采样算法 | string |
| batch\_size | 每一批次目标节点数目 | int |
| Output | block | 输出图列表 | List [Graph] |
| sample\_index | 输出图在原图的节点编号 | tensor |
| node\_index | 起始节点在输出中的编号 | tensor |

random\_walk\_sampler运算操作定义见表228。

表228　random\_walk\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| random\_walk\_sampler | 从起始节点生成随机游走序列而采样子图的接口 | Input | g | 图 | Graph  HeteroGraph |
| nodes | 起始节点 | tensor |

表228　random\_walk\_sampler运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| random\_walk\_sampler | 从起始节点生成随机游走序列而采样子图的接口 | Input | metapath | 元路径 | List [Tuple [string, string, string]] |
| prob | 存储每条边的目标节点的转移概率 | tensor |
| restart\_prob | 结束本次随机游走的概率 | float |
| length | 随机游走路径长度 | int |
| return\_eids | 是否返回边编号 | bool |
| Output | traces | 随机游走序列的节点编号 | tensor |
| eids | 随机游走序列的边编号 | tensor |
| types | 随机游走序列的节点类型编号 | tensor |

node2vec\_sampler运算操作定义见表229。

表229　node2vec\_sampler运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| node2vec\_sampler | 从起始节点生成基于node2vec随机游走序列的采样接口 | Input | g | 图 | Graph |
| nodes | 起始节点编号 | tensor |
| prob | 存储每条边的目标节点转移概率 | tensor |
| p | 在随机游走过程中节点转移控制参数 | float |
| q | 在随机游走过程中节点转移控制参数 | float |
| walk\_len | 随机游走长度 | int |
| return\_eids | 是否返回边编号 | bool |
| Output | traces | 随机游走节点编号轨迹张量 | tensor |
| eids | 随机游走边编号轨迹张量 | tensor |

* + 1. 图数据压缩度量指标

图数据压缩度量指标是用来评估图数据压缩算法的效果和性能的指标。图数据压缩算法的效果主要体现在压缩后的空间占用、数据的表征能力以及可恢复性，图数据压缩算法的的性能主要体现在压缩算法的时间开销和规模可扩展。常用的图数据压缩的度量指标有图编辑距离、压缩比、信息熵和压缩时间。

* 1. 模型量化与剪枝
     1. 量化

图模型的量化是将模型中的参数转换为定点数或浮点数。量化可以减少模型的存储空间和计算复杂度，提高模型的效率和推理速度。

EXACT是一种使用量化方法结合随机投影压缩节点嵌入的GNN内存优化技术，可以显著降低在大型图上训练GNN的内存需求而近乎不损失精度。模型定义见表230。

表230　EXACT模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| EXACT | 一个使用压缩激活训练图神经网络的内存框架 | Input | X | 节点特征矩阵 | tensor |
| edge\_weight | 边特征矩阵 | tensor |
| y\_train | 独热编码的节点标签矩阵，仅含训练集节点信息 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | Y | 节点属于各个类别标签的概率 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| activation\_compression\_bits | 压缩激活比特 | int |
| dropout\_r | 丢弃率 | float |

EXACT算法伪代码见表231。

表231　EXACT算法伪代码

|  |
| --- |
| EXACT算法 |
| 输入：, layers L |
| 输出： |
| ← {} |
|  |
| Add and to |
| ← SPMM (, ) |
| Add (in CSR format) and to |
| If : |
| Add to |
| = ReLU() |
| return |

1. 量化方法有较强的兼容性，可以与其他压缩方法联合使用。
2. 当量化方法和剪枝压缩联合使用，量化方法可以应用于剪枝后的模型以实现进一步的优化。结构化剪枝后，可以直接量化精简后的模型。非结构化剪枝后，需要量化未被剪枝掉的权重。
3. 当量化方法和结构化矩阵压缩联合使用，量化方法可以应用于结构化矩阵压缩之前。先进行量化操作，简化矩阵数据中的内容，再按照结构化矩阵的方法进行处理。
   * 1. 剪枝
        1. 概述

剪枝是减少对图模型的性能没有显著影响的参数，可以提高模型的效率和推理速度。图模型剪枝技术可根据其在图神经网络的不同功能层的应用分类，分为信息传递层剪枝和信息聚合层剪枝。

* + - 1. 信息传递层剪枝

信息传递层剪枝是在图神经网络中稀疏化信息传递层的参数。具体方法包括：基于敏感性的剪枝、基于权重值的剪枝、稀疏剪枝等。此类方法可以根据具体的信息传递层和应用场景单独应用或结合使用。

SGC模型把剪枝应用在传统GNN信息传递方式上，移除了连续层之间的非线性和折叠权重矩阵，可以去除非必要的模型复杂性和冗余计算。模型定义见表232。

表232　SGC模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SGC | 通过连续移除连续层之间的非线性和折叠权重矩阵,对传统GNN信息传递方式进行剪枝，以去除不必要的复杂性和冗余计算 | Input | X | 节点特征矩阵 | tensor |
| S | 带有自环边的对称归一化邻接矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | Y | 类（标签）矩阵 | tensor |
| H | 预测矩阵及传送集 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 消息传递的轮数 | int |
| dropout\_r | 丢弃率 | float |

* + - 1. 信息聚合层剪枝

信息聚合层剪枝是指稀疏化图神经网络中信息聚合步骤中的参数，具体方法包括执行部分邻居信息聚合、激活操作剪枝、维度变换简化等。

PPNP结合了GCN和PageRank算法来更新消息传递的步骤，可以聚合大范围内的邻居信息。模型定义见表233。

表233　PPNP模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PPNP | 通过结合GCN和PageRank算法对传统GNN信息聚会方式进行剪枝，以达到聚合大范围内邻居信息的目的。 | Input | X | 节点特征矩阵 | tensor |
| A | 带有自环边的对称归一化邻接矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | Y | 类（标签）矩阵 | tensor |

表233　PPNP模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PPNP | 通过结合GCN和PageRank算法对传统GNN信息聚会方式进行剪枝，以达到聚合大范围内邻居信息的目的。 | Input | H | 预测矩阵及传送集 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| K | 消息传递的轮数 | int |
| dropout\_r | 丢弃率 | float |

* + - 1. 剪枝评估标准

剪枝评估标准是用于衡量剪枝算法效果的标准。图模型常用剪枝评估标准包括：

1. 精度准确度：剪枝后模型在验证集或测试集上的精度准确度以及能否保持与原始模型相近的准确度；
2. 参数数量：剪枝后模型的参数数量是否减少；
3. 计算量：剪枝后模型的计算量是否减少；
4. 网络稀疏度：剪枝后模型的网络稀疏度是否增加。
   1. 模型蒸馏
      1. 概述

模型蒸馏是用一个小型学生模型学习一个大型、训练好的教师模型的知识来提高模型性能。过程中教师模型的综合知识可以转化为更精简、更有效的表示。模型蒸馏可以保持较高预测性能的同时，极大地降低模型的复杂性和计算资源需求，实现模型的轻量化和高效化。图神经网络的模型蒸馏可以分为离线蒸馏、在线蒸馏以及自蒸馏三种类型。

* + 1. 离线蒸馏

离线蒸馏采用单向知识转移和两阶段的训练过程。第一阶段为教师模型的预训练，教师模型可以充分学习训练数据来捕获和编码丰富的知识；第二阶段为知识迁移，学生模型能够通过有效的知识传递机制从教师模型中学习到关键信息。例如LSP、TinyGNN、LTD、GRACED为离线蒸馏模型。

离线蒸馏模型见表234~表239。

LSP在图神经网络领域进行离线蒸馏，使用局部结构保留模块，可以显式传递拓扑差异信息，实现拓扑感知的知识转移。LSP适用于动态图，能在图结构变化时有效蒸馏知识。模型定义见表234。

表234　LSP模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| LSP | LSP是首次针对GNN进行知识蒸馏的专门方法,通过提出的局部结构保留模块，可以显式地衡量并传递拓扑差异信息，从而实现拓扑感知的知识转移 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| teacher\_logits | 教师模型的输出，即软标签 | tensor |
| Output | Y | 节点标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| heads | 多头注意力的数量，默认为1 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| beta | 通过beta加权知识蒸馏中的硬损失和软损失，默认为0.1 | float |
| dropout | 在训练过程中，神经网络每个节点的丢弃概率，默认为0.2 | float |

TinyGNN使用邻居蒸馏策略和对等节点信息弥补了小型GNN与深层GNN间的信息差距，可以在维持高性能的同时加速GNN推理。模型定义见表235。

表235　TinyGNN模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TinyGNN | TinyGNN是为了在保持高性能的同时，实现GNN的快速推理而提出的模型。该模型通过利用对等节点信息和采用邻居蒸馏策略，来弥补小型GNN与深层GNN之间的邻居信息差距 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| teacher\_logits | 教师模型的输出，即软标签 | tensor |
| Output | Y | 节点标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |

表235　TinyGNN模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TinyGNN | TinyGNN是为了在保持高性能的同时，实现GNN的快速推理而提出的模型。该模型通过利用对等节点信息和采用邻居蒸馏策略，来弥补小型GNN与深层GNN之间的邻居信息差距 | Attributes | out\_channels | 输出特征的维度 | int |
| hidden\_channels | 隐藏层特征的维度 | int |
| dropout | 在训练过程中，神经网络每个节点的丢弃概率，默认为0.2 | float |

LTD是一个适用于多种预训练图神经网络模型的知识蒸馏框架，LTD通过学习节点特定的蒸馏温度来提升性能，可以提高蒸馏模型的性能。模型定义见表236。

表236　LTD模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| LTD | 提出一个通用的知识蒸馏框架，可以应用于任何预训练的图神经网络模型以进一步提高它们的性能。为了解决分离问题，LTD提出学习节点特定的蒸馏温度, 以提高蒸馏模型的性能 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| teacher\_logits | 教师模型的输出，即软标签 | tensor |
| Output | Y | 节点标签 | tensor |
| Attributes | in\_channels | 每个输入样本的大小 | int |
| out\_channels | 每个输出样本的大小 | int |
| heads | 多头注意力的数量，默认为1 | int |
| hidden\_dim | 隐藏层维度，默认为64 | int |
| beta | 通过beta加权知识蒸馏中的硬损失和软损失，默认为0.1 | float |
| dropout | 在训练过程中，神经网络每个节点的丢弃概率，默认为0.2 | float |

LTD算法伪代码见表237。

表237　LTD算法伪代码

|  |
| --- |
| LTD算法 |
| 输入：X, g, teacher\_logits, y |
| 输出：Y |
| While warmup do: |
| student\_logits = self.gnn\_student(g, X) |
| loss\_distill = self.ce\_loss(student\_logits, teacher\_logits, y) |
| self.gnn\_student.update(loss\_distill) |
| student\_logits = self.gnn\_student(g, X) |
| While not converge do: |
| temperature = LTDTemp(student\_logits, teacher\_logits) |
| loss\_distill = self.ce\_loss(student\_logits, teacher\_logits, y) |
| self.gnn\_student.update(loss\_distill) |
| student\_logits = self.gnn\_student(g, X) |
| loss\_temp = self.ce\_loss(student\_logits, y) |
| self.LTDTemp.update(loss\_temp) |
| final\_logits = self.gnn\_student(g, X) |
| Return log\_softmax(final\_logits, dim=-1) |

GRACED模型是一种基于预聚合的图神经网络蒸馏推理加速模型。GRACED通过定制化知识蒸馏策略来适应性地修改从教师图神经网络模型继承的知识，可以减轻图异配性对蒸馏的不良影响。模型定义见表238。

表238　GRACED模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GRACED | 提出一种基于预聚合的图神经网络蒸馏推理加速方案 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| teacher\_logits | 教师模型的输出，即软标签 | tensor |

表238　GRACED模型定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GRACED | 提出一种基于预聚合的图神经网络蒸馏推理加速方案 | Output | Y | 节点标签 | tensor |
| Attributes | in\_channels | 每个输入样本的大小 | int |
| out\_channels | 每个输出样本的大小 | int |
| hidden\_dim | 隐藏层维度，默认为64 | int |
| dropout | 在训练过程中，神经网络每个节点的丢弃概率，默认为0.2 | float |

GRACED算法伪代码见表239。

表239　GRACED算法伪代码

|  |
| --- |
| GRACED算法 |
| 输入：X, g, teacher\_logits, y |
| 输出：final\_loss |
| While not converge do: |
| GA\_Input = self.GA\_Func(X,g) |
| student\_logits = slef.GAMLP(GA\_Input) |
| loss\_student\_pred = self.ce\_loss(student\_logits, y) |
| loss\_teacher\_pred = self.ce\_loss(teacher\_logits, y) |
| distll\_weight = self.weight\_func(loss\_student\_pred, loss\_teacher\_pred) |
| loss\_distill = self.ce\_loss(student\_logits, teacher\_logits) |
| final\_loss = distll\_weight \* loss\_distill + (1- distll\_weight) \* loss\_student\_pred |
| final\_loss.backward() |
| Return final\_loss |

* + 1. 在线蒸馏

在线蒸馏同时更新教师模型和学生模型，整个框架端到端可训练，可以应对无法有效获得参数量大且高性能的教师模型的情况。

例如ROD、FreeKD为在线蒸馏模型，见表240~表241。

ROD使用同行教学和接收感知多尺度知识的在线蒸馏，优化稀疏图学习中学生模型的互补性与多样性，可以高效地传递知识。模型定义见表240。

表240　ROD模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ROD | 一种新颖的面向稀疏图学习的接收感知在线知识蒸馏方法，设计了多尺度接收感知图知识、基于任务的监督和丰富的蒸馏知识三种监督信号，输出多个学生模型在多个不同下游任务上的预测结果 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | prediction | 输出张量 | tensor |
| Attributes | hidden\_units | 隐藏层维度 | int |
| k | 学生数量 | int |
| learning\_rate | 优化器的学习率 | float |
| alpha | 损失函数中多任务间的平衡系数 | float |
| beta | 损失函数中多任务间的平衡系数 | float |

FreeKD是一种结合了强化学习的在线知识蒸馏框架，FreeKD使用分层策略协同构建两个较浅的GNN进行有效知识交换，可以实现自由方向的知识蒸馏。模型定义见表241。

表241　FreeKD模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| FreeKD | 可以动态调整知识传递方向的在线图神经网络知识蒸馏方法，使用特征张量、标签张量、邻接矩阵，输出两个模型预测的节点标签 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| L | 训练轮次 | int |
| Output | prediction | 输出张量 | tensor |
| Attributes | gamma | Reward中的超参数 | float |
| alpha | 损失函数中的平衡系数 | float |
| beta | 损失函数中的平衡系数 | float |

* + 1. 自蒸馏

自蒸馏是教师模型和学生模型使用相同的网络结构的知识蒸馏方法。

例如GNN-SD、IGSD为自蒸馏模型，见表242~表243。

GNN-SD是通过邻域差异率和自适应差异率保留正则化的自蒸馏方法，GNN-SD无需外部教师模型即可在GNN层间有效传递关键知识。模型定义见表242。

表242　GNN-SD模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GNN-SD | 可以通过在GNN不同层之间传递知识来缓解过平滑问题的自蒸馏方法，使用特征张量、标签张量、邻接矩阵，输出学生模型在不同下游任务上的预测结果 | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | prediction | 输出张量 | tensor |
| Attributes | hidden\_dim | 隐藏层维度 | int |
| dropout | 神经网络每个节点的丢弃概率 | float |
| learning\_rate | 优化器的学习率 | float |
| alpha | 损失函数中的平衡系数 | float |
| beta | 损失函数中的平衡系数 | float |

IGSD是一种迭代图自蒸馏方法。IGSD使用自监督对比学习和图扩散增强，无监督学习图级表示，在半监督场景中结合有监督和自监督损失优化，可以提升图表示的性能和区分度。模型定义见表243。

表243　IGSD模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| IGSD | 用于图表示学习的图神经网络自蒸馏方法，通过对图实例的增强视图的实例鉴别来迭代执行蒸馏过程，使用特征张量、邻接矩阵，输出学习到的图表示（自监督场景）或图预测情况（半监督场景） | Input | X | 节点特征矩阵 | tensor |
| edge\_index | 边索引 | tensor  SparseTensor |
| g | 输入图 | Graph |
| Output | prediction | 输出张量 | tensor |
| Attributes | proj\_hidden\_size | 映射隐藏层维度 | int |
| proj\_size | 映射层维度 | int |
| learning\_rate | 优化器的学习率 | float |
| label\_thre | 伪标签阈值 | float |
| Y | 图标签 | tensor |

* 1. 模型加速
     1. 概述

模型加速是通过优化算法、模型结构或硬件加速等手段提升模型的推理速度和运行效率，同时保持或尽可能减小对模型预测性能的影响。模型加速可以不显著损害模型准确度的前提下，缩短计算时间、降低计算资源需求和能耗，实现模型的快速部署和高效运行。

图神经网络的加速策略分为：并行加速策略，迭代加速策略，图划分策略和通信加速策略。

* + 1. 并行加速策略
       1. 流水线并行

流水线并行是将模型按层或模块划分为多个部分，并分配给不同的设备进行并行处理。每一个设备都执行前向和反向的计算，最后在其他设备上进行更新参数。常用的小批量流水线并行将数据切分为不同的批次，第一个批次的数据首先在第一个设备上进行前向传播，然后将中间结果传递给下一个设备。同时，第二个批次的数据可以在第一个设备上进行前向传播，不同的设备可以同时进行计算，等所有数据计算完成后进行反向计算。每个设备根据计算出的梯度更新其部分的模型参数。重复上述步骤，直到完成所有数据的训练批次。模型定义见表244　流水线并行模型定义。

表244　流水线并行模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| PipelineParallel | 不同的GNN层被分配给不同的设备。每个设备负责处理一层GNN层的数据。数据从一个设备流向下一个设备，每个设备在其负责的层上进行计算 | Input | module | 模型 | Model |
| device\_num | 设备数量 | int |
| g | 图拓扑 | Graph |
| feature | 顶点特征 | tensor |
| feature\_dim | 顶点特征维度 | int |
| Output | output\_device | 输出设备的id | int |

* + - 1. 数据并行

数据并行模型根据设备数量对数据进行划分，把每份数据分配给不同的计算设备。每个计算设备上都有完整的模型副本。在训练过程中，每个设备独立地进行前向和反向传播，计算梯度。最后汇总所有设备上的梯度，更新模型参数。重复上述过程，直至训练完成。模型定义见表245。

表245　数据并行模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DataParallel | 数据集划分到不同的设备上，每个设备都具有完整的模型副本。设备之间各自完成前向和后向计算 | Input | module | 模型 | Model |
| device\_num | 设备数量 | int |
| g | 图拓扑 | Graph |
| feature | 顶点特征 | tensor |
| batch\_size | 批量大小 | int |
| Output | output\_device | 输出设备的id | int |

* + - 1. 张量并行

张量并行是指将单个张量，例如顶点特征和顶点嵌入，按照维度切分到多个设备上，每一个设备处理数据部分维度的并行计算方法。

首先，将顶点特征或嵌入张量按特定维度切分，每个子张量分配到不同设备上。每个设备独立进行前向传播，处理其负责的子张量部分，同时在计算过程中各设备间进行必要的信息传递，以确保非线性变换和其他依赖完整张量信息的操作能够正确执行。在使用完整张量进行下游任务与损失函数计算后，得到的梯度张量继续按照特定维度切分到不同设备上进行反向传播。最后汇总所有设备上的模型参数梯度，更新模型参数。模型定义见表246　张量并行模型定义。

表246　张量并行模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| TensorParallel | 根据设备数量切分张量，每个设备计算部分数据 | Input | module | 模型 | Model |
| device\_num | 设备数量 | int |
| g | 图拓扑 | Graph |
| feature | 顶点特征 | tensor |
| Output | Module | 并行化的模块 | Model |

* + 1. 迭代加速策略

迭代加速可以在大规模图数据的处理过程中提高图神经网络的训练和推理效率。分布式训练过程中，迭代机制分为同步机制，异步机制，以及同步异步混合形式。同步机制要求所有计算节点在进入下一计算周期前达成一致，能够保持全局参数的一致性；异步机制中各节点独立更新其局部参数，不必等待其他节点，能够提高训练的并行性和效率；同步异步混合模式可以优化训练过程中的效率与模型准确性之间的权衡。该模式综合运用了同步和异步两种基本的训练机制，能够适应不同的网络状况和计算需求，训练性能更高。

SSP是一种混合训练模型。该模型以异步的方式执行训练，每隔固定迭代次数强制进行同步更新。SSP能够根据训练的具体情况灵活调整同步和异步操作，提高模型训练的效率，保证模型的收敛性并适应不同的硬件和网络环境。模型定义见表247。

表247　SPP模型定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| SSP | 使用有界陈旧性的同步异步混合模式并行训练 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| ssp\_flag | 是否使用同步异步混合模式 | bool |
| staleness\_bound | 有界陈旧值 | int |
| Output | Y | 节点标签 | tensor |
| Attributes | in\_channels | 输入特征的维度 | int |
| out\_channels | 输出特征的维度 | int |
| Module | 隐藏层特征维度 | int |
| n\_layers | 隐藏层层数 | int |

* + 1. 图划分策略

应用在分布式图神经网络系统中的图划分方法有四种：

* 1. 哈希划分通过定义不同的映射规则如顶点或边的哈希值，随机分配顶点以平衡计算和通信负载。；
  2. Metis可以快速高效地将大型稀疏图划分为多个大小几乎相等的子图，同时最小化割边；
  3. Metis-extend方法使用聚类算法和添加各种约束，如顶点掩码和顶点度数，优化传统的Metis算法。该方法能够使带标签顶点的邻居集中，平衡顶点和边的数量，减小计算和通信负载；
  4. 流式划分采用动态划分策略，通过调整评分函数灵活应对不同的划分任务。

partitiong\_graph结合了不同的图划分方法，可以优化图的划分效果，适应不同的GNN系统需求和运行环境。partitiong\_graph运算操作定义见表248。

表248　partitiong\_graph运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| partitiong\_graph | 对原始图拓扑进行图划分 | Input | X | 节点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor  SparseTensor |
| num\_parts | 图分区数量 | int |
| part\_method | 分区方法 | string |
| Output | partition\_g | 分区图 | List [Graph] |

表248　partitiong\_graph运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| partitiong\_graph | 对原始图拓扑进行图划分 | Output | partition\_X | 分区图节点特征 | List [tensor] |
| Attributes | balance\_ntypes | 根据节点类型平衡划分图 | bool |
| balance\_nodes | 根据节点数量平衡划分图 | bool |
| balance\_edges | 根据边数量平衡划分图 | bool |
| balance\_weight | 根据节点或边权重平衡划分图 | bool |
| function\_eval | 应用评分函数评价划分图 | bool |

* + 1. 通信加速策略
       1. 无损通信优化

无损通信优化是图神经网络分布式训练中的关键技术，可以提高图神经网络的通信效率，同时保证模型的准确率不受影响。无损通信优化主要通过缓存实现，具体分为：

1. 没有缓存则称为DepComm；
2. 优先级缓存可以分为度数大优先缓存和采样率高优先缓存：
   1. 度数大优先缓存优先存储度数较大的，对训练影响较大的节点数据；
   2. 采样率高优先缓存则优先存储被高频采样的节点数据，以减少频繁通信带来的开销。
3. 全量缓存（DepCache）是将所有节点的数据全部缓存，以避免重复的通信操作；
4. 部分缓存根据实际需求和缓存资源，选择性地缓存部分节点数据，可以在通信和存储资源之间找到平衡，是DepComm和DepCache的结合。

度数大优先缓存以更高的概率采样高出度节点进行缓存。度数大优先缓存运算操作定义见表249。

表249　度数大优先缓存运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| 度数大优先缓存 | 选择高出度顶点来填充缓存 | Input | g | 输入图 | Graph |
| embed\_names | 特征名称列表 | List [string] |
| sort\_nid | 按照出度排序的顶点ID | tensor |
| node\_num | 顶点数量 | int |
| dims | 特征维度 | int |
| nid\_map | 本地顶点ID到全局ID的映射 | tensor |
| data\_frame | 包含特征名称和对应特征数据的字典 | Dict [string, value] |
| Output | cached\_feature | 缓存的特征 | tensor |

表249　度数大优先缓存运算操作定义（续）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| 度数大优先缓存 | 选择高出度顶点来填充缓存 | Attributes | peak\_allocated\_mem | GPU上分配的最大内存 | int |
| peak\_cached\_mem | GPU上缓存的最大内存 | int |

采样率高优先通过GPU内存使用的一次性采样来决定缓存大小。采样率高优先缓存运算操作定义见表250　采样率高优先缓存运算操作定义。

表250　采样率高优先缓存运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| 采样率高优先缓存 | 在第一个 mini-batch 的训练周期结束后，检查空闲 GPU 内存的大小，并相应地分配可用的 GPU 内存来缓存图数据 | Input | g | 输入图 | Graph |
| embed\_names | 特征名称列表 | List [string] |
| node\_num | 顶点数量 | int |
| dims | 特征维度 | int |
| nid\_map | 本地顶点ID到全局ID的映射 | tensor |
| data\_frame | 包含特征名称和对应特征数据的字典 | Dict [string, value] |
| Output | cached\_feature | 缓存的特征 | tensor |
| Attributes | peak\_allocated\_mem | GPU上分配的最大内存 | int |
| peak\_cached\_mem | GPU上缓存的最大内存 | int |

DepCache将所有节点的数据全部缓存。DepCache运算操作定义见表251　DepCache运算操作定义。

表251　DepCache运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| DepCache | 每个worker在训练开始前，让依赖它们的邻居在本地做好准备 | Input | g | 输入图 | Graph |
| L | 模型层数 | int |
| label | 顶点标签 | tensor |
| h | 初始节点特征 | tensor |
| W | 初始参数 | tensor |
| Output | W | 更新后的参数 | tensor |

部分缓存是指在内存有限，特别是GPU内存有限的情况下，只缓存图的一部分节点及其特征数据。部分缓存运算操作定义见表252。

表252　部分缓存运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| 部分缓存 | 使用贪婪的启发式方法来划分 DepCache 和 DepComm 的依赖关系 | Input | vertices\_subset | 顶点子集 | set |
| edge\_subset | 边子集 | set |
| remote\_dependent\_neighbors | 远程依赖邻居 | List [set] |
| Output | partitions\_of\_ \_dependencies | 依赖关系的划分集合 | List [set] |
| Attributes | S | 内存限制 | int |

* + - 1. 通信压缩与有损通信优化

概述

图神经网络的通信压缩是指在分布式GNN训练和推理过程中减少节点之间的数据传输量，降低通信开销，提高整体计算效率的过程。通信压缩可以在确保GNN模型准确性和性能的前提下，显著降低分布式计算中的通信负载，提高分布式GNN训练和推理的效率。通信压缩和有损通信优化可以分为边界的随机采样，数据随机丢弃和通信数据量化：

* 1. 边界的随机采样随机选取部分远程邻居顶点以减少传输数据量；
  2. 数据随机丢弃将要传输的数据进行随机丢弃以减少传输数据量；
  3. 通信数据量化将浮点数数据转换为低精度格式以减少传输数据量。

边界的随机采样/数据随机丢弃

边界节点采样BNS是用于GNN训练的方法。该方法能够有效处理大规模图数据，提升训练效率，同时保持模型的性能。

BNS选择性地关注图中的边界节点，只采样和处理具有较大的信息量和代表性的一部分节点及其邻居，能够减少计算开销，实现高效的训练过程。BNS的运算操作定义见表253　BNS运算操作定义。

表253　BNS运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| BNS | 用于图神经网络 (GNN) 的分布式训练方法，其通过在每个训练迭代中随机采样边界顶点，可以减少通信开销，提高训练效率 | Input | X | 顶点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| Output | New\_X | 通信后拥有部分边界邻居信息的节点特征矩阵 | tensor |
| Attributes | Fanout | 边界点采样率 | float |

通信数据量化

通信数据量化是在分布式GNN训练中减少通信开销和加速训练过程的技术。

通信数据量化将高精度的浮点数表示简化为低精度表示，如8位或16位整数，从而减少数据传输量。尽管量化会引入一定的误差，但适当的量化策略可以在保持模型性能的前提下，大幅减少通信开销。通信数据量化的运算操作定义见表254。

表254　通信数据量化运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Comm-quantization | 将节点通信的数据转换为精度更低的定点数或浮点数，减少数据传输量。/用更少的比特位量化传出数据，以减少分布式通信量 | Input | X | 顶点特征矩阵 | tensor |
| g | 输入图 | Graph |
| edge\_index | 边索引 | tensor |
| Output | Quantization\_data | 量化后要传输的嵌入矩阵 | tensor |
| Attributes | n\_bits | 量化后的数据比特位数 | int |

1. 图神经网络计算框架
   1. 基于深度学习平台的图神经网络计算框架
      1. 系统体系结构

基于深度学习平台的图神经网络计算框架由多个组件构成，这些组件协同工作以支持图数据的表示学习、模型训练和推理等任务。框架的组成和相关组件之间的关系如下：

1. 系统的整体组件逻辑结构
   1. 图数据预处理层：提供全图加载和批量加载的图数据导入方式，负责图数据的加载、格式化和预处理，以便后续的模型训练和推理，包括图的构建、特征的归一化、标签的准备等；
   2. 图模型构建层：通过图卷积网络（GCN）或其他类型的GNN模型来学习节点、边或图的表示。这一层是GNN的核心，负责从图数据中提取有用的信息；
   3. 模型训练层：包含优化算法和反向传播机制，用于调整模型参数以最小化损失函数。这一层与深度学习平台紧密集成，以利用高效的计算资源。整个训练过程一般包括前向传播、损失计算、反向传播和参数更新等；
   4. 模型推理和评测层：在训练完成后，用于在新数据上执行预测和分类等任务，评测模型性能，通常包括准确性、F1分数、AUC和其它度量标准。
2. 各个组件的职责、交互方式和关系
   1. 数据预处理层与图模型构建层：数据预处理层输出格式化的图数据，供图模型构建层使用。这两层之间的交互是通过数据流进行的，预处理后的数据作为模型输入；
   2. 图模型构建层与模型训练层：图模型构建层的输出（节点或图的嵌入）被用于模型训练层中的损失计算。这两层通过梯度下降等优化算法进行交互，以改进模型性能；
   3. 模型训练层与推理层：训练好的模型参数被保存并部署，以进行模型推理。

框架图如图3所示。

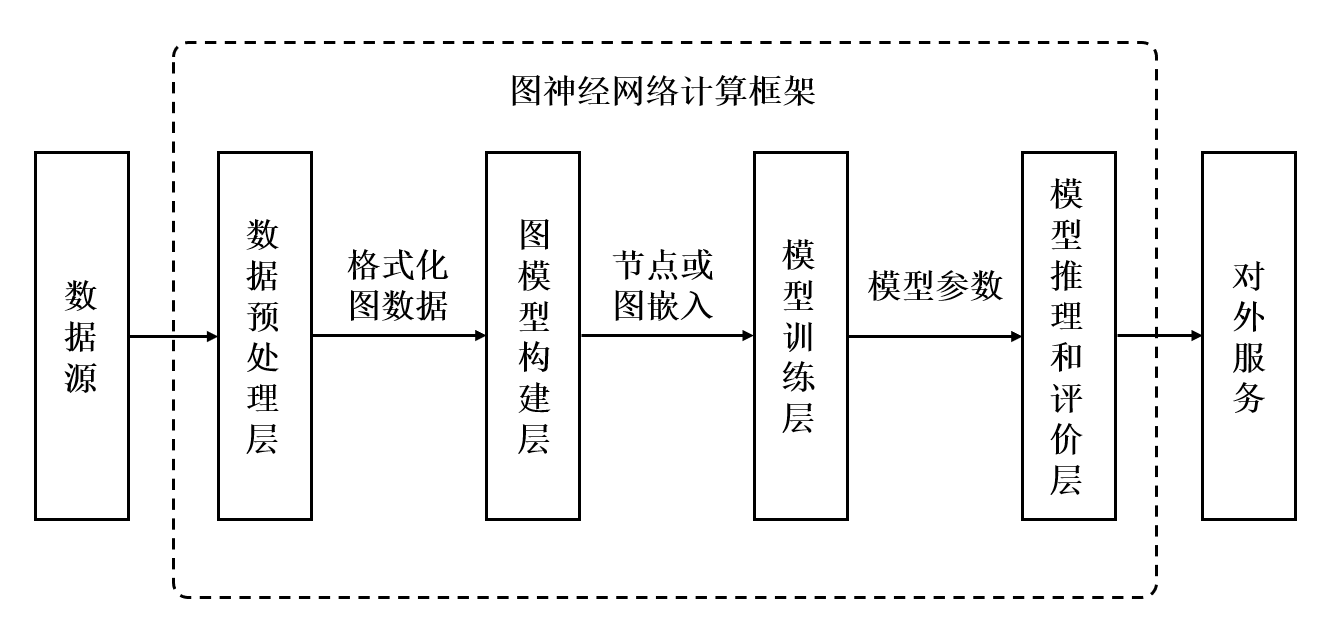


图3　基于深度学习平台的图神经网络计算框架系统结构示意图

* + 1. 与深度学习平台的接口规范
       1. 张量计算接口

张量计算接口是用于执行张量基本运算的接口，支持包括随机张量的生成、张量的四则运算以及逻辑运算在内的多种操作。其中，部分接口信息对应如下：

abs：返回张量中每一个元素的绝对值。abs运算操作定义见GB/T 42382.1-2023表18。

add：用于两个张量相加，如果张量形状不同，会进行广播。add运算操作定义见GB/T 42382.1-2023表20。

arrange：用于创建一个在指定区间内等间隔排列的一系列数值。arrange运算操作定义见GB/T 42382.1-2023表93。

argmax：返回张量中最大值的索引。argmax运算操作定义见GB/T 42382.1-2023表23。

argmin：返回张量中最小值的索引。argmin运算操作定义见GB/T 42382.1-2023表24。

cast：修改张量的数据类型。cast运算操作定义见GB/T 42382.1-2023表32。

ceil：对张量中每一个元素向上取整。ceil运算操作定义见GB/T 42382.1-2023表33。

concat：沿指定维度对几个张量进行拼接。concat运算操作定义见GB/T 42382.1-2023表34。

cos：计算张量中每个元素的余弦值。cos运算操作定义见GB/T 42382.1-2023表41。

diag：从一个一维张量创建一个对角矩阵或从矩阵提取对角元素。diag运算操作定义见GB/T 42382.1-2023表49。

divide：张量逐元素相除。divide运算操作定义见GB/T 42382.1-2023表50。

equal：判断两个张量是否逐元素相等。equal运算操作定义见GB/T 42382.1-2023表57。

exp：计算张量中每个元素的exp值。exp运算操作定义见GB/T 42382.1-2023表58。

gather：根据索引从输入张量抽取值。gather运算操作定义见GB/T 42382.1-2023表62。

logical\_and：对两个张量进行逐元素逻辑与。logical\_and运算操作定义见GB/T 42382.1-2023表22。

logical\_not：对两个张量进行逐元素逻辑非。logical\_not运算操作定义见GB/T 42382.1-2023表83。

logical\_or：对两个张量进行逐元素逻辑或。logical\_or运算操作定义见GB/T 42382.1-2023表85。

logical\_xor：对两个张量进行逐元素逻辑异或。logical\_xor运算操作定义见GB/T 42382.1-2023表125。

matmul：对两个满足要求的张量进行矩阵乘。matmul运算操作定义见GB/T 42382.1-2023表78。

multiply：两个张量的逐元素乘。multiply运算操作定义见GB/T 42382.1-2023表81。

pow：对张量中每个元素进行指数运算。pow运算操作定义见GB/T 42382.1-2023表90。

reduce\_mean：计算指定维度上张量的均值。reduce\_mean运算操作定义见GB/T 42382.1-2023表96。

reduce\_max：计算指定维度上张量的最大值。reduce\_max运算操作定义见GB/T 42382.1-2023表97。

reduce\_min：计算指定维度上张量的最小值。reduce\_min运算操作定义见GB/T 42382.1-2023表98。

reduce\_sum：计算指定维度上张量的数值和。reduce\_sum运算操作定义见GB/T 42382.1-2023表100。

reshape：用于改变张量的形状。reshape运算操作定义见GB/T 42382.1-2023表104。

shape：用于获取张量的形状。shape运算操作定义见GB/T 42382.1-2023表108。

sin：计算张量中每个元素的正弦值。sin运算操作定义见GB/T 42382.1-2023表110。

sqrt：计算张量中每个元素的平方根。sqrt运算操作定义见GB/T 42382.1-2023表115。

stack：沿一个新维度堆叠一系列张量。stack运算操作定义见GB/T 42382.1-2023表117。

substract：张量逐元素相减。substract运算操作定义见GB/T 42382.1-2023表118。

squeeze：移除张量中所有长度为1的维度。squeeze运算操作定义见GB/T 42382.1-2023表121。

transpose：用于交换张量的两个维度。transpose运算操作定义见GB/T 42382.1-2023表122。

unsqueeze：在张量中添加一个长度为1的维度。unsqueeze运算操作定义见GB/T 42382.1-2023表123。

zeros：生成形状一定，数值为0的张量。zeros运算操作定义见GB/T 42382.1-2023表129。

本文件包括额外的张量计算接口，具体见表255~表266。

atan运算操作定义见表255。

表255　atan运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| atan | 计算张量中每一个元素的反正切值 | Input | X | 输入张量 | tensor |
| Output | Y | 输出张量 | tensor |

cumsum运算操作定义见表256。

表256　cumsum运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| cumsum | 计算张量中沿指定维度的累积和 | Input | X | 输入张量 | tensor |
| dim | 求和维度 | int |
| Output | Y | 输出张量 | tensor |

eye运算操作定义见表257。

表257　eye运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| eye | 创建一个二维的单位矩阵 | Input | m | 行数 | int |
| n | 列数 | int |
| dtype | 生成张量的数据类型 | string |
| Output | Y | 输出张量 | tensor |

mask\_select运算操作定义见表258。

表258　mask\_select运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| mask\_select | 根据一个布尔掩码张量筛选输入张量元素 | Input | X | 输入张量 | tensor |
| mask | 掩码张量 | tensor |
| axis | 选择维度 | int |
| Output | Y | 输出张量 | tensor |

ones运算操作定义见表259。

表259　ones运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ones | 生成形状一定，数值为1的张量 | Input | shape | 输出张量的维度 | List [int] |
| dtype | 输出张量的数据类型 | string |
| Output | Y | 输出张量 | tensor |

ones\_like运算操作定义见表260。

表260　ones\_like运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ones\_like | 创建一个形状与输入张量相同、所有元素为1的张量 | Input | X | 输入张量 | tensor |
| dtype | 输出张量的数据类型 | string |
| Output | Y | 输出张量 | tensor |

tile运算操作定义见表261。

表261　tile运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| tile | 用于沿各个维度生成重复的张量 | Input | X | 输入张量 | tensor |
| multiples | 重复次数 | tensor  Tuple [int, int]  List [int] |
| Output | Y | 输出张量 | tensor |

tan运算操作定义见表262。

表262　tan运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| tan | 计算张量中每个元素的正切值 | Input | X | 输入张量 | tensor |
| Output | Y | 输出张量 | tensor |

to\_tensor运算操作定义见表263。

表263　to\_tensor运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| to\_tensor | 将数据转换为张量 | Input | X | 输入张量 | tensor |
| dtype | 输出张量的数据类型 | string |
| Output | Y | 输出张量 | tensor |

to\_numpy运算操作定义见表264。

表264　to\_numpy运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| to\_numpy | 将张量转换为numpy.array | Input | X | 输入张量 | tensor |
| Output | Y | 输出张量 | tensor |

where运算操作定义见表265。

表265　where运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| where | 根据一个条件张量筛选两个张量的元素 | Input | X | 输入张量1 | tensor |
| Y | 输入张量2 | tensor |
| condition | 条件 | tensor |
| Output | Z | 输出张量 | tensor |

zeros\_like运算操作定义见表266。

表266　zeros\_like运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| zeros\_like | 创建一个形状与输入张量相同、所有元素为0的张量 | Input | X | 输入张量 | tensor |
| dtype | 输出张量类型 | string |
| Output | Y | 输出张量 | tensor |

* + - 1. 数据加载与预处理接口

数据加载与预处理接口是用于加载和预处理数据的接口，支持文本或序列数据的加载与处理。包括数据加载接口等，具体见表267。

表267　dataloader操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| dataloader | 加载和预处理数据集，共神经网络模型使用 | Input | dataset\_path | 输入数据集路径 | string |
| batch\_size | 批量大小 | int |
| shuffle | 是否打乱 | bool |
| drop\_last | 是否丢弃最后一组数据 | bool |
| Output | Y | 划分好的数据 | List [tensor] |

* + - 1. 模型构建接口

模型构建接口是用于构建神经网络模型的接口，包括模型结构的定义、参数的初始化等。其中，部分接口定义信息见GB/T 42382.1-2023，具体如下：

Linear：线性层，全连接层。Linear运算操作定义见GB/T 42382.1-2023表130。

random\_unifrom：根据均匀分布初始化网络权重。random\_unifrom运算操作定义见GB/T 42382.1-2023表92。

random\_normal：根据正态分布初始化网络权重。random\_normal运算操作定义见GB/T 42382.1-2023表91。

本文件包括额外的模型构建接口，具体见表268~表273。

Module定义见表268。

表268　Module操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Module | 神经网络模块的基类，自定义的模型应当继承这个类 | Input | X | 输入数据 | tensor |
| Output | Y | 输出张量 | tensor |
| Attributes | params | 模型参数 | map<string, tensor> |

表268　Module操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| Module | 神经网络模块的基类，自定义的模型应当继承这个类 | Attributes | layers | 神经网络层 | map<string, Module>  map<string, None> |
| params\_status | 模型参数是否为可学习参数 | map<string, bool> |
| trainable\_weights | 可学习参数列表 | List [tensor] |
| nontrainable\_weights | 不可学习参数列表 | List [tensor] |
| all\_weights | 所有参数列表 | List [tensor] |
| is\_train | 模型是否为训练模式 | bool |

ModuleList定义见表269。

表269　ModuleList操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ModuleList | 专门存储神经网络模块的列表 | Input | X | 输入数据 | tensor |
| Output | Y | 输出张量 | tensor |
| Attributes | layer\_list | 神经网络列表 | List [Module] |

ModuleDict定义见表270。

表270　ModuleDict操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| ModuleDict | 专门存储神经网络模块的字典 | Input | X | 输入数据 | tensor |
| Output | Y | 输出张量 | tensor |
| Attributes | layer\_dict | 神经网络列表 | map<string, Module> |

truncated\_normal运算操作定义见表271。

表271　truncated\_normal运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| truncated\_normal | 根据截断正态分布中抽取的样本，初始化网络权重 | Input | shape | 输出张量的形状 | Tuple [int, int] |
| mean | 正态分布的期望 | float |
| std | 正态分布的标准差 | float |
| dtype | 输出张量的数据类型 | string |
| seed | 随机数种子 | int |
| Output | Y | 输出张量 | tensor |

xavier\_uniform运算操作定义见表272。

表272　xavier\_uniform运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| xavier\_uniform | 从均匀分布中初始化网络权重，分布范围根据输入和输出神经元数量自动调整 | Input | shape | 输出张量的形状 | Tuple [int, int] |
| gain | 均匀分布参数 | float |
| dtype | 输出张量的数据类型 | string |
| seed | 随机数种子 | int |
| Output | Y | 输出张量 | tensor |

xavier\_normal运算操作定义见表273。

表273　xavier\_normal运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| xavier\_normal | 从正态分布中初始化网络权重，分布范围根据输入和输出神经元数量自动调整 | Input | shape | 输出张量的形状 | Tuple [int, int] |
| gain | 正态分布参数 | float |
| dtype | 输出张量的数据类型 | string |
| seed | 随机数种子 | int |
| Output | Y | 输出张量 | tensor |

* + - 1. 激活函数接口

激活函数接口是调用非线性变换函数的接口，包括ReLU、Sigmoid等。其中，部分接口定义信息见GB/T 42382.1-2023，具体如下：

ELU：为负输入提供负值缓解梯度消失问题。ELU运算操作定义见GB/T 42382.1-2023表134。

ReLU：最常用的激活函数之一。ReLU运算操作定义见GB/T 42382.1-2023表102。

ReLU6：ReLU的变体，通过设置上限为6以提供输出数值的范围限制。ReLU6运算操作定义见GB/T 42382.1-2023表103。

LeakyReLU：类似于PReLU，但斜率是预先设定，不可学习。LeakyReLU运算操作定义见GB/T 42382.1-2023表140。

PReLU：ReLU的一种变体，负数部分的斜率是可学习的参数，不同的输入特征，斜率可以不同。PReLU运算操作定义见GB/T 42382.1-2023表142。

Sigmoid：将输入值压缩到[0, 1]范围内，通常用于二分类问题的输出层。Sigmoid运算操作定义见GB/T 42382.1-2023表143。

Softmax：归一化指数函数，通常用于多分类问题的输出层，将一组值转换为概率分布。Softmax运算操作定义见GB/T 42382.1-2023表145。

Softplus：ReLU的平滑版本。Softplus运算操作定义见GB/T 42382.1-2023表146。

Tanh：双曲正切激活函数。Tanh运算操作定义见GB/T 42382.1-2023表148。

本文件包括额外的激活函数接口，具体见表274。

GeLU运算操作定义见表274。

表274　GeLU运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| GeLU | 该激活函数常见于Transformer模型，可以看作ReLU和ELU的一种折中选择 | Input | X | 输入张量 | tensor |
| Output | Y | 输出张量 | tensor |

* + - 1. 损失函数接口

损失函数接口时用于计算预测值和真实值之间的差距的接口。其中，部分接口定义信息见GB/T 42382.1-2023，具体如下：

cross\_entropy：适用于多分类任务的损失函数，对于每个样本，该函数会计算正确类别概率的负对数。cross\_entropy运算操作定义见GB/T 42382.1-2023表43。

simoid\_cross\_entropy：适用于二分类任务，会通过sigmoid函数将输出压缩到[0, 1]范围，随后计算交叉熵损失函数。simoid\_cross\_entropy运算操作定义见GB/T 42382.1-2023表144。

本文件包括额外的损失函数接口，具体见表275~表276。

binary\_cross\_entropy运算操作定义见表275。

表275　binary\_cross\_entropy运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| binary\_cross\_entropy | 用于二元分类任务的损失函数，计算真实标签和预测标签之间的交叉熵 | Input | output | 输入张量 | tensor |
| target | 标签 | tensor |
| reduction | 计算模式，可以选择“mean”、“sum”、 “none” | string |
| Output | Y | 输出张量 | tensor |

mean\_squared\_error运算操作定义见表276。

表276　mean\_squared\_error运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| mean\_squared\_error | 适用于回归任务的损失函数，计算预测值和真实值之间的均方误差 | Input | output | 输入张量 | tensor |
| target | 标签 | tensor |
| reduction | 计算模式，可以选择“mean”、“sum”、 “none” | string |
| Output | Y | 输出张量 | tensor |

* + - 1. 优化器接口

优化器接口是用于模型训练中参数优化的接口。以下优化器部分定义及描述信息见GB/T 42382.1-2023表154归一化表达以及后续接口支持情况说明。

SGD：最传统的优化方法，旨在有效地调整网络参数以最小化损失函数。

Adagrad：适用于处理稀疏数据，频繁更新的参数学习率会降低，较少更新的参数学习率会增加。

Adadelta：主要解决Adagrad训练后期学习率急剧下降的问题。

Adam：计算梯度的一阶和二阶矩估计，并根据这些估计调整学习率。

Adamax：Adam的一个变体，主要基于无穷范数，提供一种更稳定的优化方法，特别是在非稳定参数更新的情况。

RMSprop：通过维持一个平均值来调整每一个参数的学习率，避免Adagrad学习率持续下降的问题。

* + - 1. 正则化接口

正则化接口是用于提高模型的泛化能力并防止模型过拟合的接口。其中，部分接口定义信息见GB/T 42382.1-2023，具体如下：

batchnorm：将输入规范化，使其均值为0，方差为1。batchnorm运算操作定义见GB/T 42382.1-2023表132。

dropout：在训练过程中随机丢弃一些神经元，使得网络不依赖于某一特定特征，提升其泛化能力。dropout运算操作定义见GB/T 42382.1-2023表51。

layernorm：类似于batchnorm，但归一化是在单个样本层面而非整个批次。layernorm运算操作定义见GB/T 42382.1-2023表72。

l2\_normalize：将输入特征向量除以其L2范数，通常用于约束网络权重和特征向量，使其不会变得过大。l2\_normalize运算操作定义见GB/T 42382.1-2023表76。

本文件包括额外的正则化接口，具体见表277。

gaussian\_noise运算操作定义见表277。

表277　gaussian\_noise运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| gaussian\_noise | 在输入添加服从高斯分布的随机噪声，可以增加模型对输入噪声的鲁棒性，提高泛化能力 | Input | mean | 噪声均值 | float |
| std | 噪声方差 | float |
| is\_always | 是否在训练和测试中都添加噪声 | bool |
| seed | 随机数种子 | int |
| Output | Y | 输出张量 | tensor |

* + - 1. 下游评测接口

下游评测接口是用于模型的评估和测试的接口。其中，部分接口定义信息见GB/T 42382.1-2023，具体如下：

accuracy：表示模型预测正确的样本占总样本的比例。accuracy运算操作定义见GB/T 42382.1-2023表19。

auc：衡量模型在不同阈值设置下的性能，AUC越高，代表模型区分不同类别的能力越强。auc运算操作定义见GB/T 42382.1-2023表29。

本文件包括额外的下游评测接口，具体见表278~表279。

precision运算操作定义见表278。

表278　precision运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| precision | 衡量模型预测为正类的样本有多少是真正的正类 | Input | pred | 预测结果 | tensor |
| label | 标签 | tensor |
| Output | Y | 输出张量 | tensor |

recall运算操作定义见表279。

表279　recall运算操作定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| recall | 衡量模型正确识别正类样本的能力 | Input | pred | 预测结果 | tensor |
| label | 标签 | tensor |
| Output | Y | 输出张量 | tensor |

* + - 1. 其他接口

其他深度学习接口，包括日志、模型保存、可视化功能等。

* 1. 图神经网络计算框架与第三方数据源接口
     1. 图数据库接口规范

图神经网络计算框架与图数据库的接口规范定义了图神经网络框架与图数据库之间进行数据交互的方式和规则。将图数据库作为外部数据源接入图神经网络框架的基本过程包括：

1. 图神经网络框架连接数据库；
2. 从图数据库中拉取数据，并构造图神经网络框架支持的图数据类型；
3. 使用图神经网络框架的采样、消息传递等模块进行训练/推理；
4. 将得到的嵌入/预测结果写回图数据库。

同时，考虑到图神经网络框架获取部分子图的需求，还应该提供图数据库的查询接口。

据此，图数据库接口规范应该主要包含以下模块：

1. 图数据库连接接口：实现图神经网络计算框架与图数据库的连接，使计算框架能够与图数据库通过接口进行交互；
2. 数据查询接口：用于向图数据库查询并获取图数据。这些接口可能包括指定节点或边的查询，按条件过滤数据，获取节点/边的属性信息等并输入至图神经网络框架；
3. 数据获取接口：用于将数据从图数据库导入到计算框架中，并进行图神经网络模型的训练和推断。这些接口可以将从图数据库中获取的数据转化为规定的中间标准数据形式；
4. 数据写回接口：在图神经网络计算过后，可能需要对图数据库中的数据进行写回操作，如将训练结果写回图数据库中。
   * + 1. 图数据库连接接口

图数据库连接接口是用于维护图数据库连接的接口，图数据库的连接信息定义见表280。

表280　图数据库的连接信息定义

|  |  |  |
| --- | --- | --- |
| GraphDBConnection | 指定参数 | |
| 字段 | 类型 | 定义 |
| graph\_address | string | 图数据库地址 |
| user\_name | string | 用户名 |
| password | string | 密码 |

* + - 1. 获取图数据接口

获取图数据接口是用于从图数据获取图数据的接口。先规定点、边的导出配置，然后通过导出配置来控制图数据的输出，可以是整图也可以是规定了点类型或者边类型的子图。具体定义见表281~表283。

点导出配置决定了如何从图数据库拉取节点数据。点导出配置定义见表281。除了节点本身外，节点上可能带有特征和标签的信息。表中x\_property\_names定义了需要从图数据库中获取哪些属性用来构建特征信息，而y\_property\_names表示标签对应的属性名。

表281　点导出配置定义

|  |  |  |
| --- | --- | --- |
| NodeExportConfig | 指定参数 | |
| 字段 | 类型 | 定义 |
| label\_name | string | 节点类型名 |
| x\_property\_names | List [string] | 节点特征对应的属性名 |
| y\_property\_names | List [string] | 节点标签对应的属性名 |

类似地，边导出配置定义见表282。

表282　边导出配置定义

|  |  |  |
| --- | --- | --- |
| EdgeExportConfig | 指定参数 | |
| 字段 | 类型 | 定义 |
| label\_name | string | 边类型名 |
| src\_dst\_label | Tuple [string, string] | 源节点与目的节点类型 |
| x\_property\_names | List [string] | 边特征对应的属性名 |
| y\_property\_names | List [string] | 边标签对应的属性名 |

通过点、边的导出配置，可从图数据库中导出图神经网络框架需要的图数据结构。同时可以通过点、边的不同导出配置，规定点或边的类型来实现获取相应子图。图数据的详细定义参见本标准第5.2章节中的图数据类型定义。本文件使用一个统一的列表来表示点、边导出配置，当图数据为同质图时，列表元素只有一个，而为异质图时则可以有多个元素。获取图数据接口定义见表283。

表283　获取图数据接口定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| get\_graph | 从图数据库中导出异质图 | Input | conn | 图数据库连接信息 | GraphDBConnection |
| graph\_name | 图名 | string |
| node\_export\_config | 点导出配置 | List [NodeExportConfig] |
| edge\_ export\_config | 边导出配置 | List [EdgeExportConfig] |
| Output | graph | 图神经网络框架的图数据 | HeteroGraph |

* + - 1. 查询接口

查询接口是用于向图数据库查询并获取图数据信息的接口。这些接口可以按类型、属性值等条件过滤数据以获取具体的节点/边，同时返回符合条件的点或边的全部信息。具体接口的定义见表284。

表284　查询接口定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| match | 在图数据库中查询 | Input | graph\_name | 图名 | string |
| label\_name | 点或边的类型 | List [string] |
| src\_dst\_label | 对于边要限制源节点与目的节点类型 | Tuple [string, string] |
| x\_property\_names | 点或边的属性及其值 | Dict [string, value] |
| y\_property\_names | 点或边的标签及其值 | Dict [string, value] |
| Output | node\_all或edge\_all | 符合条件的点或边的全部信息 | Dict[string,List] |

* + - 1. 写回接口

写回接口是用于向图数据库写回属性的接口，写回的属性包括训练得到的节点嵌入，以及节点或边的预测标签等。写回接口定义见表285。

表285　写回接口定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| write\_property | 向图数据库写入属性 | Input | label\_name | 类型名 | string |
| src\_dst\_label | 对于边要限制源节点与目的节点类型 | Tuple [string, string] |
| property\_name | 写入属性名 | string |
| property\_values | 写入值 | List [value] |
| Output | success | 写入成功标志 | bool |

* + 1. 批处理系统接口规范
       1. 批处理输入

图数据接口需要分别定义点、边数据格式，同时，还需要定义标签数据格式，用于描述需要对哪些点、边生成图样本。具体见表286~表291。

点数据格式定义见表286。

表286　点数据格式

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| id | 点编号 | 用于唯一标识图中的一个点 |
| feature | 点属性 | 用于描述节点的属性信息 |

feature字段用于存储类型信息和特征信息，用于表示点类型和各种类型的特征，定义见表287。

表287　点数据feature字段定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| type | 类型名 | 表示点类型，如果是同质图，可以为空 |
| sparseKey | 稀疏键 | 表示一组特征ID，如：1，2，3 |
| sparseKV | 稀疏键值对 | 表示一组特征ID-Value对，如：1:0.1，2:0.2，4:0.01 |
| dense | 稠密数组 | 表示一个特征向量，如：0.1，0.2，0.002，0.4 |
| raw | 原始数据 | 表示一个字符串 |

边数据格式定义见表288。

表288　边数据格式定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| src\_id | 起点编号 | 用于标识一条边的起点 |
| dst\_id | 终点编号 | 用于标识一条边的终点 |
| feature | 边属性 | 用于描述边的属性信息 |

相较于点上的feature，边上的feature字段在类型信息上有所区别，具体定义见表289。

表289　边数据feature字段定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| src\_type | 起点类型名 | 表示起点类型，如果是同质图，可以为空 |
| edge\_type | 边类型名 | 表示边类型，如果是同质图，可以为空 |
| dst\_type | 终点类型名 | 表示终点类型，如果是同质图，可以为空 |
| sparseKey | 稀疏键 | 表示一组特征ID, 如：1，2，3 |
| sparseKV | 稀疏键值对 | 表示一组特征ID-Value对，如：1:0.1，2:0.2，4:0.01 |
| dense | 稠密数组 | 表示一个特征向量，如：0.1，0.2，0.002，0.4 |
| raw | 原始数据 | 表示一个字符串 |

点标签数据格式定义见表290。

表290　点标签数据格式定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| id | 起点编号 | 用于标识一条输入样本的点 |
| label | 标签 | 用于标识一条输入样本的类型 |

边标签数据格式定义见表291。

表291　边标签数据格式定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| src\_id | 起点编号 | 用于标识一条样本边的起点 |
| dst\_id | 终点编号 | 用于标识一条样本边的终点 |
| label | 标签 | 用于标识一条样本边的标签 |

* + - 1. 批处理输出

批处理系统的输出可以描述为输入和子图，具体见表292~表294。

点样本数据格式定义见表292。

表292　点样本数据格式定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| id | 起点编号 | 用于标识一条输入样本的点 |
| subgraph | 子图 | 用于表示点id的子图，按照本标准第5.2节所示结构组织数据 |
| label | 标签 | 用于标识一条输入样本的标签，对于inference样本，可以为空 |

边样本数据格式定义见表293。

表293　边样本数据格式定义

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段名（中文） | 说明 |
| src\_id | 起点编号 | 用于标识一条样本边的起点 |
| src\_subgraph | 起点子图 | 用于表示起点src\_id的子图，按照本标准第5.2节所示结构组织数据 |
| dst\_id | 终点编号 | 用于标识一条样本边的终点 |
| dst\_subgraph | 终点子图 | 用于表示终点dst\_id的子图，按照本标准第5.2节所示结构组织数据 |
| label | 标签 | 用于标识一条样本边的标签，对于Inference样本，可以为空 |

批处理接口的输入主要分为两部分，即输入图数据和采样参数，前者是加工对象，后者描述加工约束，详细定义见表294。

表294　批处理接口定义

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算操作 | 描述 | 参数 | 子参数 | 定义 | 数据类型 |
| BatchProcess | 批处理接口 | Input | nodes | 输入点数据 | - |
| edges | 输入边数据 | - |
| labels | 输入标签数据，可以是边标签，也可以是点标签 | - |
| processArgs | 处理参数，如邻居跳数、每跳邻居数、出入边、采样策略等 | - |
| Output | success | 边、点样本 | - |