第 4 章 图神经网络初步

引言

图神经网络（Graph Neural Networks, GNNs）作为近年来深度学习中的重要分支，因其在处理图结构数据中的卓越表现而备受关注。GNN 能够有效捕捉节点与节点之间的复杂关系，在社交网络、推荐系统、化学分子等众多领域展现了出色的表现。与传统方法相比，图神经网络的优势在于它能够通过图结构信息进行更精确的特征学习和关系建模，从而提升任务的准确性和泛化能力。本章将深入探讨图神经网络的基础内容，包括通用框架、核心算子以及发展历程。接着，将详细介绍核心算子的设计方法，并以图卷积网络和图采样聚合网络等经典模型为例，分析它们在消息传递算子和图池化算子上的设计差异及应用特点。最后，本章还将介绍图神经网络的训练方法，分别涉及节点级别和图级别的任务。

本章学习目标

(1) 理解图神经网络的基础概念和图结构数据的特性；
(2) 掌握通用的图神经网络框架，包括消息传递算子和图池化算子；
(3) 熟悉经典的图神经网络模型，了解它们的工作原理与应用场景；
(4) 了解如何训练图神经网络模型，针对不同任务选择合适的训练方法。

4.1 图神经网络基础

图神经网络[1,2]（Graph Neural Networks, GNNs）是一类专门处理图结构数据的深度学习模型，它们在社交网络分析、知识图谱、生物信息学等领域展现出了卓越的性能。GNNs 的核心优势在于能够有效捕捉节点间的复杂关系，并通过消息传递机制对节点特征进行聚合与更新，从而学习到节点、边甚至整个图的高质量表示。

GNNs 通常由多层构成，每一层执行特定的图卷积操作，包括节点特征的转换、消息传递和聚合等。这些操作使得 GNNs 能够捕捉到节点的局部和全局邻域信息，进而用于节点分类、链接预测和图分类等任务。GNNs 的架构设计允许其在不同的层级上捕捉图数据的拓扑结构和特征信息，这一点与传统的神经网络模型如卷积神经网络[3]（CNNs）和循环神经网络[4]（RNNs）形成鲜明对比，后者主要处理规则的数据结构，如文本和图像。

在实际应用中，GNNs 已经被证明在多个领域具有广泛的应用潜力。例如，在社交网络分析中，GNNs 可以帮助识别社区结构和信息传播模式；在药物发现中，GNNs 能够预测分子间的相互作用；在推荐系统中，GNNs 能够利用用户和物品之间的关系来提升推荐的准确性和多样性。

随着研究的深入，GNNs 的变种模型也在不断涌现，如图卷积网络、图采样聚合网络以及图注意力网络等，这些模型通过引入注意力机制、门控单元等创新点，进一步提升了 GNNs 的性能和适用性。此外，为了解决大规模图数据的计算效率问题，研究者们还提出了各种优化策略，如邻域采样、图分区等技术，使得 GNNs 能够更高效地处理大规模图结构数据。

4.1.1 通用图神经网络框架

图神经网络（Graph Neural Networks, GNNs）是一类能够有效处理图结构数据的深度学习模型，广泛应用于各种需要对图数据进行推理与预测的任务中。图数据由节点（nodes）

和边（edges）组成，其中节点代表实体，边则表示节点之间的关系。这种数据的复杂结构使得传统的神经网络难以直接处理，而 GNNs 通过其特有的框架，能够对节点、边甚至整个图进行信息的高效处理和表达。

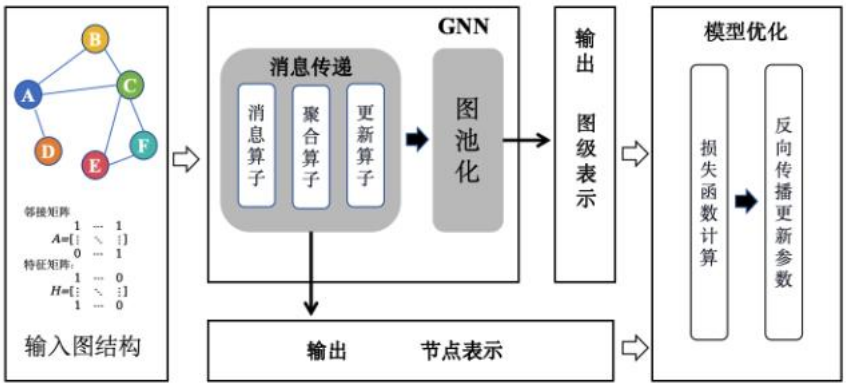

图 4-1 图神经网络框架

如图 4-1 所示，GNNs 的核心思想是通过 消息传递算子来传播和聚合信息，使得每个节点的表示不仅包含自身的特征，还隐含了图中邻居节点及其关系的综合信息。在这一过程中，信息在节点间通过边不断交换与更新，逐步形成全局视角下的节点特征表示。消息传递的多轮迭代使得 GNNs 能够捕捉图结构中的局部与全局关系，从而完成如节点分类、边预测等节点级任务。

然而，仅通过消息传递往往难以充分表达复杂图的全局结构，尤其在需要将整个图嵌入到固定维度的向量表示时，因而图池化机制成为了关键。图池化通过对图中节点或边进行降维处理，将其信息压缩为全局性的特征表示。这不仅有助于降低计算复杂度，还能够更好地捕捉图的全局特征，使得 GNNs 可以在图分类等任务中表现优越。

在得到节点级别或者图级别的表示之后，GNNs 需要计算损失函数，如交叉熵损失或均方误差，来衡量 GNNs 的预测与真实标签之间的误差。进而通过反向传播算法将误差梯度逐层传递，不断更新 GNNs 的参数，从而模型能够有效学习数据中的模式，不断提升其性能。损失函数的计算与参数的反向传播构成了模型优化的关键步骤，使得 GNNs 可以在复杂的图数据任务中实现良好的预测能力。

4.1.2 图神经网络输入与输出

1. 图神经网络的输入

图神经网络（GNN）的输入与输出定义了其与数据的交互方式。与传统的神经网络不同，GNN 需要处理复杂的图结构数据，因此，其输入通常包括以下部分：

1) 节点特征（Node Features）：

节点特征通常以矩阵形式表示，其中每一行对应一个节点的特征向量，列数为特征的维度。例如，一个具有n个节点的图，如果每个节点具有d个特征，则节点特征矩阵的维度为 $n \times d$。

2) 邻接矩阵（Adjacency Matrix）：

邻接矩阵表示图的连接关系。对于一个有 n个节点的图，邻接矩阵的维度为 d。在实际应用中，为了增强节点自身特征对表示的影响，常常在邻接矩阵中添加自环（Self-loop）。

3) 边特征（Edge Features）：

对于边拥有特征的图，可用边特征矩阵表示，其中每行对应一条边的特征向量。这在需要捕捉节点间复杂关系时尤为重要。

2. 图神经网络的输出

GNN 的输出根据任务类型不同而有所差异：

1) 节点级别任务：

节点分类任务输出每个节点的嵌入表示或类别预测结果，通常以一个 $1 \times c$的矩阵表示，其中c是类别数。

2) 边级别任务：

边预测任务输出每条边的存在概率或类别。输出的通常为这条边是否存在的概率，概率值越高则说明存在的可能性越大。

3) 图级别任务：

图分类或回归任务的输出是整个图的嵌入表示或预测结果，通常是一个固定维度的向量，用于分类或回归。

4.1.3 图神经网络核心算子

1. 消息传递算子

消息传递计算（Message Passing Computation）是图神经网络的核心机制，用于在图结构中传递信息并更新节点或图的表示。其基本思想是每个节点通过与邻居节点的信息交互来学习到图结构中的特征表示。这种过程是逐步迭代的，随着迭代次数的增加，节点能够从更广泛的邻居节点中汇集信息，从而逐步获得全局图结构的感知。

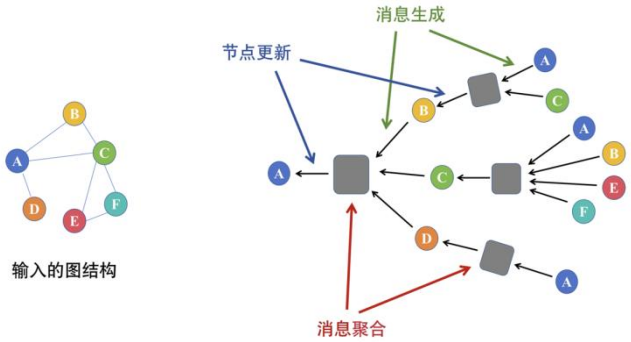

图 4-2 消息传递机制

如图 4-2 所示，在这一机制中，消息传递操作通常分为三个步骤：消息生成、消息聚合和节点更新。每个节点通过与其直接邻居的节点生成消息，随后聚合邻居的消息，并基于聚合的结果和自身的当前状态来更新节点表示。这一过程可在多轮迭代中重复，逐步扩展节点的感受野。

1) 消息算子

如图 4-3 所示，消息算子（Message Operators）是图神经网络（GNNs）中至关重要的组成部分，它负责在节点之间生成信息，使得每个节点能够从其邻居中获取有意义的数据。这一过程是图结构学习的基础，影响着后续的消息聚合和节点更新。消息算子的设计和实现直接关系到模型的性能和表达能力。

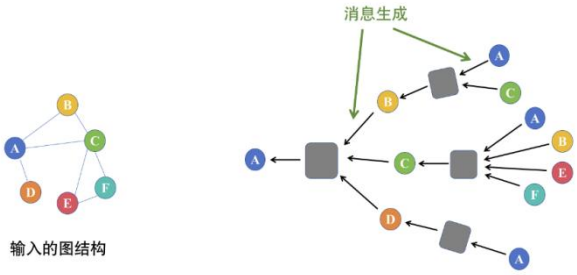

图 4-3 消息算子

定义：在给定图$G=(V,E)$的情况下，假设节点在第k轮迭代时的特征表示为$\boldsymbol{h}_j^{(k)}$，并且它的邻居节点为$N(v_i)$。消息算子生成的消息$\boldsymbol{m}_{ij}^{(k)}$，从节点$v_j$到节点$v_i$的公式可以表示为：

$$\boldsymbol{m}_{ij}^{(k)}=\phi\left(\boldsymbol{h}_j^{(k)},\boldsymbol{e}_{ij}\right),\ j\in N(v_i)$$

这里，ϕ 是定义消息生成过程的函数，e_{ij}是边特征，通常包括节点间的关系信息。

2) 聚合算子

如图 4-4 所示，聚合算子 (Aggregate Operators) 是图神经网络（GNN）中用于整合邻居节点信息的关键步骤。它决定了目标节点如何汇集来自邻居的消息，从而生成一个综合表示。在多轮迭代中，聚合算子不断扩大节点的感受野，使节点能够捕获更广泛的图结构信息。

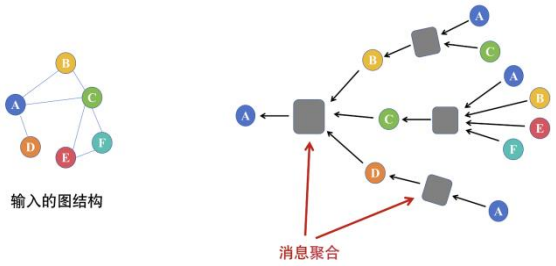

图 4-4 聚合算子

定义：假设在给定的图$G=(V,E)$中，节点 vi 的邻居集合为$N(v_i)$。第t 轮迭代中，节点 v_i 从邻居节点 $j\in N(v_i)$ 接收到的消息记为 $\boldsymbol{m}_{ij}^{(k)}$。聚合算子用于将这些消息整合为一个综合表示 $\boldsymbol{m}_i^{(k)}$。公式如下：

$$\boldsymbol{m}_i^{(k)}=\psi\left(\left\{\boldsymbol{m}_{\boldsymbol{ij}}^{(k)}|j\in N(v_i)\right\}\right)$$

其中， 是聚合函数，定义了消息的整合方式。

3）更新算子

如图 4-5 所示，更新算子(Update Operators) 负责将聚合后的消息与节点自身的嵌入表示结合，进而更新节点的状态。这一过程使得节点在消息传递过程中不断更新其表示。这一更新机制在多轮迭代中至关重要，它确保了节点嵌入的动态性和灵活性，使得模型能够有效地学习图结构中的复杂关系。

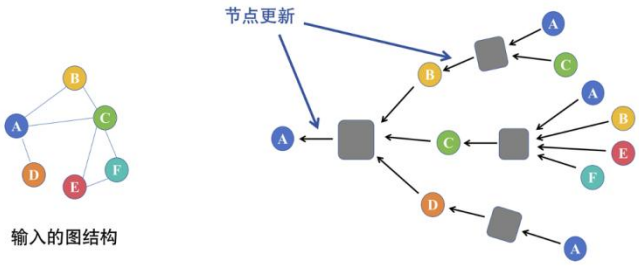

图 4-5 更新算子

定义：在每一轮迭代中，假设节点v_i在第k轮的嵌入表示为$\boldsymbol{h}_i^{(k)}$，并且通过聚合算子得到的邻居消息为$\boldsymbol{m}_i^{(k)}$。更新算子的目标是结合当前的节点表示和聚合的消息，从而生成新的节点表示 $\boldsymbol{h}_i^{(k+1)}$。这一过程可以通过以下公式表示：

$$\boldsymbol{h}_i^{(k+1)} = \gamma\left(\boldsymbol{h}_i^{(k)}, m_i^{(k)}\right)$$

其中，γ 是定义更新过程的函数。更新算子的选择直接影响节点状态的更新效果，决定了信息在图中的传播方式。

2. 图池化算子

图池化是 GNNs 中用于降低图的复杂度和维度的重要步骤。如图 4-6 所示，与传统的卷积神经网络（CNNs）中的池化操作相似，图池化的目的是从一个较大图中提取出具有代表性的子图或节点，同时保留图的关键特征。图池化操作的有效性对于图分类等图级别任务的性能具有重要影响。

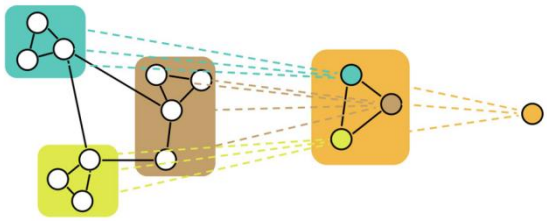

图 4- 6 图池化算子

定义：图池化操作通常是对图中的节点进行选择或聚合，以形成更小的图或缩小节点的特征表示。假设在某一时刻，图 $G = (V, E)$ 的节点集为V，图池化操作的目标是生成一个新的图$G' = (V', E')$，其中$V' \subseteq V$。图池化的操作可以表示为：

$$\boldsymbol{h}_{V'} = \text{READOUT}\ (\boldsymbol{h}_V)$$

其中，h_V表示原图中所有节点的嵌入表示，READOUT$(\cdot)$ 是定义池化过程的函数，生成的新节点表示 $h_{V'}$ 通常为原节点表示的聚合结果。

3. 模型优化

在图神经网络(GNNs) 优化的过程及训练的过程中，包含优化目标（损失函数）的计算以及反向传播两个部分。这些步骤确保模型可以高效地学习参数，并在训练数据上取得良好的泛化性能。

(1) 优化目标。在图神经网络（GNNs）中，优化目标（即损失函数），用于衡量模型预测与真实标签之间的差异，指导模型的优化方向。损失函数的选择取决于具体任务（如节点分类、链接预测、图分类等）。

(2) 反向传播。反向传播（Backpropagation）[24]是神经网络模型训练中的关键算法，它通过计算损失函数相对于网络参数的梯度，从而实现模型参数的逐层更新。反向传播背后的核心思想是链式法则（Chain Rule）的应用，它允许我们有效地计算网络层数较多时的梯度。为了保证高效训练，反向传播通常与梯度下降算法（Gradient Descent）或其变种（如 Adam 和 RMSProp）结合使用。

4.1.4 图神经网络发展

图神经网络[1,2]（Graph Neural Networks, GNNs）的出现源于对图结构数据高效建模的需求，是深度学习技术在非欧几里得数据上的重要拓展。最初，研究者通过递归方式迭代节点状态以捕捉局部结构信息，但这一方法在处理复杂图结构和大规模数据时表现有限。随后，空域方法[6,7]的提出标志着 GNNs 进入成熟阶段，其核心在于一个通用的设计框架，包括消息传递算子和图池化算子。消息传递算子主要用于节点级任务，通过聚合邻居节点的特征更新中心节点的表示；而图池化算子则应用于图级任务，通过生成全局图的低维表示完成分类等操作。这一框架统一了图神经网络的设计思路，并为后续研究和应用奠定了坚实基础。

在 GNNs 的发展历程中，谱图方法[5]作为早期代表，以图谱理论为基础，利用频域中的滤波器捕捉图的全局结构信息。这种方法依赖于拉普拉斯矩阵的特征分解，可以有效建模节点间的复杂关系。然而，其计算复杂度较高，对图结构的强假设限制了其扩展性，主要适用于规则图或小规模图场景。尽管谱方法具有一定的理论优势，其局限性促使研究者逐步转向空域方法，后者直接在图的邻域结构上进行特征聚合，避免了频域计算的高复杂度。空域方法凭借计算效率和扩展性的优势，成为图神经网络研究的主流方向。我们将在第八章系统地从理论和实践角度详细讲解谱图方法。

随着图数据结构的复杂性增加，异质图学习[10,11]成为 GNNs 研究的重要分支。异质图学习专注于处理具有多类型节点和边的复杂图结构，能够捕捉不同类型数据之间的丰富关系。这一方向通过类型感知的特征聚合机制，极大地扩展了 GNNs 的应用范围，尤其在知识图谱、生物网络和推荐系统等场景中表现出色。然而，异质图数据的高复杂性对模型的设计效率和可扩展性提出了更高要求，这也为后续自动化优化方法的出现奠定了需求基础。我们将在第七章深入解析异质图学习的理论与应用。

与此同时，随着标注数据的稀缺性限制了 GNNs 的进一步应用，图对比学习[9]为无监督图学习提供了新的突破方向。通过构建代理任务（例如最大化局部与全局表示的互信息）或利用多视图对比，图对比学习能够在减少人工标注依赖的同时，有效提升模型的泛化能力。这一方法的提出，不仅为 GNNs 拓宽了应用场景，还奠定了构建更鲁棒、更高效模型的理论基础。对于这一方向，我们将在相关章节展开讨论。

在实际应用中，随着图任务复杂度的提升，可信图学习[24,25]成为研究的重要议题。此方向聚焦于提升 GNNs 的鲁棒性和公平性，以应对图数据中的噪声、偏差以及潜在攻击。通过对抗训练、去偏正则化等方法，可信图学习显著增强了模型在不确定环境下的适应能力，同时提升了结果的可解释性。这一方向的持续发展为实现更加安全可靠的图学习系统提供了基础支持，我们将在第九章系统探讨可信图学习。

为了进一步提升效率，自动图学习[12,13]则通过神经架构搜索（NAS）和自动特征提取技术，解决了传统模型设计的复杂性问题。这种方法能够根据具体任务需求自动生成优化模型架构，不仅降低了开发门槛，还极大提升了模型的灵活性和适应性。这一领域的进展为图学习应用的规模化和多任务化提供了重要支撑。

在图神经网络表达能力的进一步探索中，研究者逐步突破了传统 GNNs 的局限，图 Transformer[14,15]通过结合图结构与 Transformer 架构的全局注意力机制，显著提升了模型在远距离节点关系建模上的能力。相较于传统 GNNs 只能局部传播信息的方式，图

Transformer 引入的全局视角使其在分子设计、知识图谱和社交网络等任务中展现了极高的建模能力，为全局建模奠定了技术基础。

基于上述方向的发展，图基础模型[16,17]的概念应运而生。这些模型通过在大规模图数据上预训练通用表示，支持跨任务迁移，成为图学习研究中的新里程碑。从分子设计到推荐系统，图基础模型展现了 GNNs 在跨领域应用中的巨大潜力，同时也为科学研究和工业应用提供了重要的技术支持。第十章将系统探讨当前图基础模型中的热点与未来趋势。最后，随着科学研究中的复杂问题不断涌现，GNNs for Science 成为一个备受关注的新

领域。通过将 GNNs 的强大建模能力与科学数据的多样性相结合，研究者在分子建模[19]、材料设计和生物信息[18]等领域取得了显著进展。这一方向展现了 GNNs 在解决复杂科学问题中的潜力，推动了科学研究与技术创新的进一步融合。我们将在后续章节深入探讨 GNNs 在科学领域中的应用及未来发展方向。

4.2 核心算子设计

在了解了通用图神经网络的框架和核心算子的基本定义之后，接下来将深入探讨这些核心算子的设计原理与实现。核心算子在图神经网络中扮演关键角色，决定了节点和边的信息传递与聚合方式，直接影响模型在捕捉图结构和节点关系上的表现。通过详细分析核心算子的构建方法和优化策略，我们将进一步揭示图神经网络如何有效地从复杂图结构中提取特征信息。

4.2.1 消息算子

1. 消息算子

1) 消息算子的类型：

在实际应用中，消息算子通常可以采用以下两种形式：

(1) 线性变换（Linear Transformation）：这是最基本的消息生成方式，通过简单的线性层将邻居节点的嵌入映射为消息。公式表示为：

$$\boldsymbol{m}_{ij}^{(k)} = \boldsymbol{W} \cdot \boldsymbol{h}_j^{(k)} + \boldsymbol{b},$$

其中，$\boldsymbol{W}$是权重矩阵，$\boldsymbol{b}$ 是偏置项。这种算子适用于需要快速信息生成的情景。

(2) 非线性变换（Non-linear Transformation）：通过引入非线性激活函数，增强模型的表达能力。公式为：

$$\boldsymbol{m}_{ij}^{(k)} = \sigma\left(\boldsymbol{W} \cdot \boldsymbol{h}_j^{(k)} + \boldsymbol{b}\right),\ j \in N(v_i)。$$

这里，σ 是非线性激活函数（如 ReLU、Sigmoid 等）。这种方式允许消息算子捕捉更复杂的特征关系，从而提高模型的灵活性。

2) 案例分析：

我们以一个简单的图为例$G = (V,\ E)$，包含三个节点：$V = \{v_1, v_2, v_3\}$。

邻接矩阵$\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

图G的初始特征矩阵$\boldsymbol{H}$，每个节点的特征维度为 2：

$$\boldsymbol{H} = \begin{bmatrix} 1.0 & 0.5 \\ 0.2 & 0.3 \\ 0.4 & 0.7 \end{bmatrix}$$

权重矩阵 $\boldsymbol{W}$：

$$\boldsymbol{W} = \begin{bmatrix} 0.5 & 0.5 \\ 0.1 & 0.9 \end{bmatrix}$$

偏置向量 $\boldsymbol{b} = [0.1, 0.2]$。

消息算子生成从节点 v_j 传递到节点 v_i 的消息。常用的消息生成方法如下：

(1) 线性变换：

$$\boldsymbol{m}_{ij} = \boldsymbol{W} \cdot \boldsymbol{h}_j + \boldsymbol{b}$$

以 $\mathbf{m}_{12}$ 为例：

$$\mathbf{m}_{12} = \boldsymbol{W} \cdot \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 0.35 \\ 0.49 \end{bmatrix}$$

(2) 非线性变换：

加入激活函数 σ，例如 ReLU：

$$\boldsymbol{m}_{12} = \mathrm{ReLU}(\boldsymbol{W} \cdot \boldsymbol{h}_2 + \boldsymbol{b})$$

$$\boldsymbol{m}_{12} = \mathrm{ReLU}\left(\begin{bmatrix} 0.35 \\ 0.49 \end{bmatrix}\right) = \begin{bmatrix} 0.35 \\ 0.49 \end{bmatrix}$$

2. 聚合算子

1) 基础运算聚合

这些算子通过简单的数学操作整合邻居节点的消息，计算速度快且易于实现。

(1) 最小值聚合（Min Aggregation）： 从邻居节点的消息中选择每一维度上的最小值，用于保留邻居中较低的特征信息。

$$\boldsymbol{m}_i^{(k)} = \min_{j \in N(v_i)} \boldsymbol{m}_{ij}^{(k)}$$

(2) 最大值聚合（Max Aggregation）： 从邻居节点的消息中选择每一维度上的最大值，突出最显著的特征。

$$\boldsymbol{m}_i^{(k)} = \max_{j \in N(v_i)} \boldsymbol{m}_{ij}^{(k)}$$

(3) 求和聚合（Sum Aggregation）： 将所有邻居的消息进行逐维求和，是最常用的聚合方式之一。

$$\boldsymbol{m}_i^{(k)} = \sum_{j \in N(v_i)} \boldsymbol{m}_{ij}^{(k)}$$

这些基础聚合算子具有计算简单、参数少的优势，但在某些复杂任务中，可能难以捕捉更复杂的图结构信息。同时，基础运算算子需要聚合节点的全部邻居，这就要求模型在训练时可以访问整个图结构，包括所有的节点和他们的边。一般我们将这种学习方式称为传导式学习（Transductive Learning）。

2) 采样聚合

在大规模图或节点邻居数量极多的场景中，计算每个节点的完整邻居集合的消息会导致计算开销过高。采样聚合（Sampling Aggregation）是一种解决这一问题的高效方法，通过从邻居集合中随机选择一部分节点来近似聚合结果。这种方法既降低了计算成本，又能保持足够的上下文信息，因此在处理超大规模图时十分常见。

采样聚合主要包括以下几种类型：

(1) 固定数量采样（Fixed-size Sampling）： 从邻居集合 $N(v_i)$ 中随机选择固定数量的节点进行聚合。假设每轮迭代采样 k 个邻居，采样到的邻居记为 $N_k(v_i)$。

$$\boldsymbol{m}_i^{(k)} = \psi\left(\{\boldsymbol{m}_{ij}^{(k)} | v_j \in N_k(v_i)\}\right)$$

(2) 概率采样（Probability-based Sampling）： 为每个邻居节点分配一个概率，基于该概率进行采样。例如，使用度数归一化概率：

$$p(v_j) = \frac{1}{|N(v_j)|}$$

根据概率 $p(v_j)$ 对邻居集合进行抽样，并使用这些邻居的消息来进行聚合。

(3) 层次采样（Layer-wise Sampling）： 在每一层网络中为所有节点独立采样邻居。这种方法常用于类似 GraphSAGE 等模型中,每层的采样邻居数可以逐层缩减，以达到信息压缩的目的。GraphSAGE 是一种经典的基于采样聚合的 GNN 模型。其核心思想是在每一层通过随机采样部分邻居来更新节点嵌入:

$$\boldsymbol{h}_i^{(k+1)} = \sigma\left(W \cdot AGGREGATE\left(\{\boldsymbol{h}_j^{(k)} | v_j \in N_k(v_i)\}\right)\right)$$

其中，$N_k(v_i)$ 是从邻居集合中随机采样的邻居，$AGGREGATE(\cdot)$ 可以是求和、最大值或平均等聚合算子。

采样聚合是一种别适用于大规模图数据场景的聚合策略。相比于基础运算聚合需要将完整的图结构输入到模型当中去，采样聚合使得模型在训练阶段仅使用部分图的节点及其结构信息，而在测试阶段，模型可以繁华到未见过的新节点或者图结构，我们一般呈这种学习方式为归纳学习（Inductive Learning）。

3) LSTM 聚合

LSTM 聚合使用长短期记忆网络（LSTM）来处理邻居节点的消息，特别适用于邻居节点存在顺序关系或需要捕捉复杂依赖的情况。

$$\boldsymbol{h}_i^{(k)} = LSTM\left(\boldsymbol{h}_i^{(k-1)}, \boldsymbol{m}_{ij}^{(k)}\right)$$

在这种聚合方式中，每次迭代会将当前节点状态和新的邻居消息输入到 LSTM 单元中，更新后的节点状态能够保留更长期的历史信息。这种方法增强了模型的表达能力，但计算成本较高。

4) 注意力聚合

注意力聚合（Attention Aggregation）为每个邻居节点的消息赋予一个可学习的权重，根据消息的重要性进行加权求和。这种方法能够动态地关注关键的邻居信息。

$$\boldsymbol{m}_i^{(k)} = \sum_{v_j \in N(v_i)} \alpha_{ij}^{(k)} \cdot \boldsymbol{m}_{ij}^{(k)}$$

其中，注意力权重 $\alpha_{ij}^{(k)}$ 是通过以下公式计算的：

$$\alpha_{ij}^{(k)} = \frac{exp\left(LeakyReLU\left(\boldsymbol{a}^{\mathrm{T}}\left[\boldsymbol{W} \cdot \boldsymbol{h}_i^{(k)} \,\|\, \boldsymbol{W} \cdot \boldsymbol{h}_j^{(k)}\right]\right)\right)}{\sum_{v_k \in N(v_i)} exp\left(LeakyReLU\left(\boldsymbol{a}^{\mathrm{T}}\left[\boldsymbol{W} \cdot \boldsymbol{h}_i^{(k)} \,\|\, \boldsymbol{W} \cdot \boldsymbol{h}_k^{(k)}\right]\right)\right)}$$

这里，$\boldsymbol{a}$ 是可学习的共享参数，$\boldsymbol{W}$是权重矩阵，$\|$ 表示向量拼接操作。

5) 案例分析：

给定图$G = (V, E)$，$v_1 \in V$，v_1的邻居集合为$N(v_1) = \{v_2, v_3, v_4, v_5\}$。每个邻居的消息如

下：$\boldsymbol{m}_{12}=[0.2,0.5]$、$\boldsymbol{m}_{13}=[0.3,0.7]$、$\boldsymbol{m}_{14}=[0.4,0.1]$、$\boldsymbol{m}_{15}=[0.8,0.3]$。

(1) 采样固定数量邻居并使用求和聚合：

假定每轮采样 2 个邻居。在本轮中，采样结果为 $\{v_2,v_5\}$。

使用求和聚合：

$$\boldsymbol{m}_1^{(k)}=\boldsymbol{m}_{12}+\boldsymbol{m}_{15}=[0.2+0.8,0.5+0.3]=[1.0,0.8]$$

(2) 概率采样并使用注意力机制聚合：

每个邻居以相同概率 $1/4$ 被选中。在多次迭代中，每轮可能采样到不同的邻居，从而提高模型的鲁棒性和泛化能力。

假设本轮节点 v_1 的邻居为 v_2 和 v_3，它们的消息分别为：$\boldsymbol{m}_{12}=[0.35,0.49]$ 和 $\boldsymbol{m}_{13}=[0.6,0.82]$。

使用注意力聚合：

假设注意力权重为 $\alpha_{12}=0.7$ 和 $\alpha_{13}=0.3$：

$$\boldsymbol{m}_1=0.7\cdot[0.35,0.49]+0.3\cdot[0.6,0.82]$$

计算得到：

$$\boldsymbol{m}_1=[0.385,0.343]+[0.18,0.246]=[0.565,0.589]$$

以上，这些聚合方法展示了如何在图神经网络中整合邻居节点的消息，从而有效捕捉图结构和节点之间的关系。不同的采样方法还能相互组合，应用于更多不同的场景。采样与注意力机制的结合则更是在保持计算效率的同时，增强了模型的预测能力和鲁棒性。

3. 更新算子

1) 更新算子的类型

在实际应用中，更新算子可以采用不同的形式，以下是两种常见的更新算子：

(1) 加法更新（Additive Update）：通过简单地将当前节点的嵌入与聚合的消息相加，更新节点的表示：

$$\boldsymbol{h}_i^{(t+1)}=\boldsymbol{h}_i^{(k)}+\boldsymbol{m}_i^{(k)}$$

这种方式保持了节点自身信息和邻居信息的简单结合，易于实现。

(2) 门控更新（Gated Update）：通过引入门控机制来动态调整节点信息的更新比例。这种方法通常利用一个学习到的门控向量 $\boldsymbol{g}$，以此来控制节点嵌入的更新:

$$\boldsymbol{h}_i^{(t+1)}=(\boldsymbol{1}-\boldsymbol{g})\odot\boldsymbol{h}_i^{(k)}+\boldsymbol{g}\odot\boldsymbol{m}_i^{(k)}$$

其中 $\boldsymbol{g}=\sigma\left(\boldsymbol{W}_g\cdot\boldsymbol{h}_i^{(k)}+\boldsymbol{b}_g\right)$ 是通过$Sigmoid$激活函数计算得出的门控向量。此更新方式允许模型自适应地选择保留多少旧信息以及引入多少新信息，增强了模型的灵活性。

2) 激活函数

激活函数的非线性使得神经网络几乎可以任意逼近任何非线性函数。如果不使用激活函数，无论神经网络有多少层，其每一层的输出都是上一层输入的线性组合，这样的神经网络表达能力十分有限。激活函数的选择多种多样，一个基本的要求是连续可导，可以允许在少数点上不可导。在图神经网络中，由于网络结构往往较深，信息在节点间的传递过程中会不断叠加非线性变换，因此激活函数不仅影响模型的表达能力，还会对梯度的传递产生显著影响。如果激活函数选择不当，可能会导致梯度在反向传播时逐层缩小或放大，进而引发梯度消失或梯度爆炸问题。接下来，我们详细分析常见的非激活函数及其在 GNN 中的适用性，并介绍针对这些问题的常见优化策略。

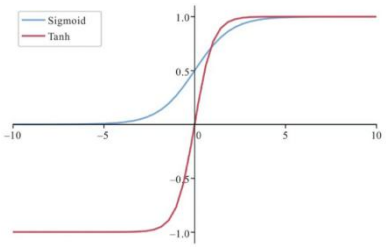

图 4-7 Sigmoid 函数和 Tanh 函数

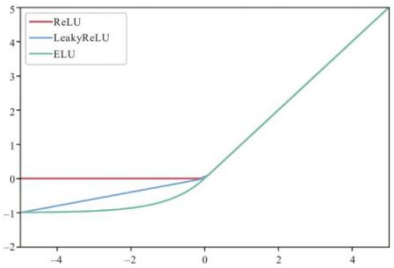

图 4-8 ReLU、LeakyReLU、ELU 函数

常见的非激活函数有：

(1) S 型激活函数

Sigmoid ：

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

如图 4-7 所示，$Sigmoid$[20]函数将任意大小的输入都压缩到 0 到 1 之间，输入的值越大，压缩后越趋近于 1；输入值越小，压缩后越趋近于 0。当$x \in [-1, 1]$时，可以近似看作线性函数，并且当$x = 0$ 时，函数值为 0.5。

它在神经网络中常常用作二分类器最后一层的激活函数，可将任意实数值转换为概率；另一个应用场景是由于它的值域为(0,1)，故可以作为一个类似于开关的调节器，用于对其他信息进行调节。

Tanh：

$$\sigma(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

相比较于$Sigmoid$，$Tanh$[21]的值域范围更大一些，为$(-1, 1)$。在需要中心化数据时（如图节点特征的变换）可能有一定效果，但通常更倾向于 ReLU 变体。

(2) ReLU 及其变种

ReLU：

$$\sigma(x) = max(0, x)$$

如图 4-8 所示，线性整流函数[22]（ Rectified Linear Unit，ReLU ）的正负值处理方式截然不同：当输入为负时，输出直接被置为零，而正值输入则保持不变，这一特性被称为单侧抑制。这种特性在隐藏层中带来了输出的稀疏性，有助于模型的计算效率。同时，由于正输入时 ReLU 的输出与输入一致，其梯度为 1，这在一定程度上缓解了梯度消失的问题。然而，单侧抑制也可能导致某些神经元“死亡”——当某个神经元的输出始终为负时，反向传播中的梯度将始终为零，使得该神经元无法得到有效更新。

$\boldsymbol{LeakyReLU}$：

$$\sigma(x) = \max(\lambda x, x), \quad \lambda > 0$$

通过引入一个小斜率λ来允许负值通过，从而减少神经元死亡现象。

$\boldsymbol{ELU}$：

$$\mathrm{ELU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha(\mathrm{e}^x) - 1 & \text{if } x < 0 \end{cases}$$

不同于$LeakyReLU$在输入为负时，进行线性压缩，指数线性单元（Exponential Linear Unit，ELU）在输入为负时，进行非线性变换。其中$\alpha > 0$，α是一个超参数，控制着输入为负时的饱和区。它具有调节激活值的均值 为 0 的功能，可以加速神经网络的收敛。

3) 案例分析

更新算子将聚合后的消息与节点的当前特征结合，生成新的特征。

假设节点 v_1 当前特征为 $\boldsymbol{h}_1 = [1.0,0.5]$，聚合消息为 $\boldsymbol{m}_1 = [0.95,1.31]$。

加法更新：$\boldsymbol{h}_1' = \boldsymbol{h}_1 + \boldsymbol{m}_1 = [1.0,0.5] + [0.95,1.31] = [1.95,1.81]$

门控更新：假设门控向量 $\boldsymbol{g} = [0.5,0.5]$：

$$\boldsymbol{h}_1' = (\boldsymbol{1} - \boldsymbol{g}) \odot \boldsymbol{h}_1 + \boldsymbol{g} \odot \boldsymbol{m}_1$$

$$\boldsymbol{h}_1' = [0.5,0.25] + [0.475,0.655] = [0.975,0.905]$$

激活函数：$\boldsymbol{h}_1' = ReLU(\boldsymbol{h}_1')$

4.2.2 图池化算子

在图神经网络中，图池化算子是一种关键操作，用于在不同层次上整合图结构及其节点特征，从而生成紧凑的图表示。根据池化范围的不同，图池化算子可以分为全局池化和层次池化。全局池化直接对整张图的节点特征进行聚合，适合生成固定大小的图表示；而层次池化则通过逐步筛选和简化图结构，在捕捉局部细节的同时保留全局信息。

首先，我们将介绍最常用的全局池化方法及其具体实现方式。

1. 全局池化

这种方法通过对整个图的节点嵌入进行聚合来生成一个单一的全局特征向量。常见的聚合操作包括求和、平均和最大值等。公式可以表示为：

$$\boldsymbol{h}_g = READOUT(\boldsymbol{h}_V) = \sum_{v \in V} \boldsymbol{h}_v$$

这种方法对输入图的规模不敏感，能够直接生成固定维度的输出。

2. 层次池化

在层次池化中，节点通过一系列的选择和聚合操作，逐步形成更小的图。例如，可以选择一部分节点进行聚合，并根据其重要性进行重新连接，形成一个新的图结构。具体过程可以用以下公式表示：

$$\boldsymbol{h}_{V'} = select(\boldsymbol{h}_V, A)$$

其中，A 是节点选择的策略（如注意力机制、边权重等），而 select 函数用于选择重要节点。

3. 案例分析

考虑一个简单的图结构，其中节点及其嵌入表示如下：节点 v_1 的嵌入表示为 $\boldsymbol{h}_1 = [0.5,0.3]$；节点 v_2 的嵌入表示为 $\boldsymbol{h}_2 = [0.2,0.8]$；节点 v_3 的嵌入表示为 $\boldsymbol{h}_3 = [0.6,0.1]$。

在全局池化中，我们可以使用求和操作来生成一个全局特征向量：

$$\boldsymbol{h}_g = \boldsymbol{h}_1 + \boldsymbol{h}_2 + \boldsymbol{h}_3 = [0.5,0.3] + [0.2,0.8] + [0.6,0.1] = [1.3,1.2]$$

这里，生成的全局特征向量 $\boldsymbol{h}_g = [1.3, 1.2]$ 是整个图的综合表示。

通过这个案例，我们可以清楚地看到图池化操作如何通过聚合节点信息，生成更加紧凑且具代表性的特征表示。全局池化通过直接汇总所有节点信息，生成固定大小的输出；而层次池化则通过选择重要节点形成新的子图，增强了模型的表达能力。

4.2.3 模型优化

模型优化是图神经网络训练中的核心步骤，通过优化模型参数，最大化模型在特定任务上的表现。不同任务的需求决定了模型优化的目标和方法，而损失函数则是这一过程的关键工具。损失函数不仅衡量模型预测与真实目标的偏差，也为模型参数更新提供了方向和依据。

1. 优化目标

1) 节点级别任务

(1) 节点分类任务的目标是预测未标记节点的类别，因此通常被视为监督学习问题。由于 节点的标签种类通常较多，节点分类任务也属于典型的多分类问题。对于多分类的监督任务，我们通常选择**交叉熵损失函数**[23]：

$$Loss = -\sum_{v \in V_l} \sum_{c=1}^{C} y_v^c \, log(\hat{y}_v^c)$$

其中y_v^c 表示节点 v 的真实标签，采用 $one-hot$ 编码的形式。如果节点属于类别 c ，则 $y_v^c = 1$，否则 $y_v^c = 0$。$\hat{y}_v^c$ 表示模型对节点v属于类别 c 的预测概率。C表示类别的总数。V_l 是带标签的节点集合。

该损失度量了模型预测的概率分布与真实分布之间的差异。信息量越大，预测越不确定，因此模型需通过优化交叉熵最小化这种不确定性。

(2) 链接预测任务的目标是判断两节点之间是否存在边，我们通常将其归为有监督的二分 类任务。对于监督二分类任务，我们可以使用**二元交叉熵损失函数**：

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

其中，N 是总共的样本数量（即节点对的数量）。y_i 是第 i 个节点对的真实标签，1 表示存在边，0 表示不存在边。具体而言，$y_i \log(\hat{y}_i)$：这项专门处理正样本。当 $y_i = 1$ 时，这项成为有效项，如果 $\hat{y}_i$接近 1，损失变小；如果 $\hat{y}_i$接近 0，损失变大。$(1 - y_i)\log(1 - \hat{y}_i)$：这项专门处理负样本。当$y_i = 0$ 时，这项成为有效项，如果 $\hat{y}_i$ 接近 0，损失变小；如果 $\hat{y}_i$ 接近 1，损失变大。

2) 图级别任务

图级别任务的目标是为整个图分配一个标签，我们通常将其归为有监督的分类任务或回归任务。对于分类任务，我们可以使用**交叉熵损失函数**，其形式为：

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} [y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$

其中，N 是总共的图数量。y_i 是第 i 个图的真实标签，1 表示图属于某个类别，0 表示图不属于该类别。

对于回归任务，我们通常使用**均方误差（Mean Squared Error, MSE）损失函数**，其形式为：

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

其中，y_i是第 i 个图的真实值，$\hat{y}_i$ 是模型预测的值。损失函数衡量的是预测值与真实值之间的距离。当模型的预测值与真实值越接近时，损失越小。

2. 反向传播

反向传播的一般过程：

1) 链式法则计算梯度

根据损失函数 $\mathcal{L}$，对每一层的参数 $\boldsymbol{\theta}$ 计算梯度：

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}^{(L)}} \cdot \frac{\partial \boldsymbol{h}^{(L)}}{\partial \boldsymbol{\theta}}$$

其中，$\boldsymbol{h}^{(L)}$ 表示第 L 层的输出。

2) 梯度传播过程

从输出层向输入层逐层反向传播误差：

$$\boldsymbol{\delta}^{(l)} = \left(\boldsymbol{W}^{(l+1)}\right)^{\mathrm{T}} \boldsymbol{\delta}^{(l+1)} \circ \sigma'\left(\boldsymbol{h}^{(l)}\right)$$

其中，$\boldsymbol{\delta}^{(l)}$ 是第 l 层的误差，$\circ$ 表示 Hadamard 乘积，σ' 是激活函数的导数。

3) 参数更新

使用梯度下降或其变种（如 Adam、RMSProp）更新参数：

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$$

其中，η 是学习率。

4) 案例分析

通过上述步骤，我们从输入特征开始，经过两层 GNN 的消息传递、聚合与更新，最终计算出输出预测值和损失。这个简单的案例展示了 GNN 的核心工作流程及其在实际任务中的应用。

任务：给定一个小型社交网络中的用户-好友关系图，采用两层消息传递网络（Message Passing Network, MPN）预测每个用户的兴趣标签。考虑一个无向图 $G = (V, E)$，其中节点集 $V = \{0,1,2\}$ 表示 3 个用户，边集 $E = \{(0,1),(1,2)\}$ 表示好友关系。节点特征矩阵 $\boldsymbol{X} \in \mathbf{R}^{3\times 2}$ 包含每个用户的特征向量：$\boldsymbol{X} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix}$。目标是预测用户的二分类兴趣标签（如是否感兴趣：0 或 1）。

(1) 第一层消息传递与更新

邻接矩阵 $\boldsymbol{A}$：$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$;消息传递公式：$\boldsymbol{H}^{(1)} = \boldsymbol{\sigma}\left(\boldsymbol{A} \cdot \boldsymbol{X} \cdot \boldsymbol{W}^{(0)}\right)$

其中，$\boldsymbol{W}^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 是第一层权重矩阵，激活函数 σ 为 ReLU：$\sigma(x) = max(0, x)$。

计算 $\boldsymbol{A} \cdot \left(\boldsymbol{X} \cdot \boldsymbol{W}^{(0)}\right)$ 得到第一层输出：$\boldsymbol{H}^{(1)} = \begin{bmatrix} 2 & 3 \\ 4 & 3 \\ 2 & 3 \end{bmatrix}$

(2) 第二层消息传递与更新

重复前面的步骤，使用第二层权重矩阵 $\boldsymbol{W}^{(1)} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ 进行计算：计算 $\boldsymbol{A} \cdot \boldsymbol{H}^{(1)} \cdot \boldsymbol{W}^{(1)}$ 得到第二层输出：$\boldsymbol{H}^{(2)} = \begin{bmatrix} 0.5 & 1 \\ 0 & 0.5 \end{bmatrix}$

(3) 输出层与预测

假设输出层权重 $\boldsymbol{W}^{out} = [1 \quad -1]$：$\hat{y}_0 = \sigma(-0.5) \approx 0.377, \quad \hat{y}_1 = \sigma(-0.5) \approx 0.377$

(4) 损失计算

我们使用二元交叉熵损失函数来衡量模型的预测输出与真实标签之间的误差。对于标签 y_i 和预测输出 $\hat{y}_i$，单个样本的损失表示为：

$$\mathcal{L}_i = -\left(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)\right)$$

假设真实标签为 $y_0 = 1$ 和 $y_1 = 0$，则总损失为：

$$\mathcal{L} = -\frac{1}{2}\left(y_0 log(\hat{y}_0) + (1 - y_0) log(1 - \hat{y}_0) + y_1 log(\hat{y}_1) + (1 - y_1) log(1 - \hat{y}_1)\right)$$

代入具体值进行计算：

$$\mathcal{L} = -\frac{1}{2}\left(1 \cdot log(0.377) + 0 \cdot log(0.623) + 0 \cdot log(0.377) + 1 \cdot log(0.623)\right)$$

$$= -\frac{1}{2}(-0.975 + - 0.475) = 0.725$$

(5) 反向传播与参数更新

我们通过反向传播（Backpropagation）计算每层权重的梯度，并通过梯度下降法更新权重。假设学习率 $\eta = 0.1$，则更新公式为：

$$\boldsymbol{W}^{out} \leftarrow \boldsymbol{W}^{out} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{\text{out}}}$$

类似地，计算隐藏层的梯度，并更新第一层和第二层的权重。每层权重更新都会逐层反向传播误差，以提高模型的预测能力。

4.3 经典图神经网络

接下来我们将介绍一些经典的图神经网络模型，它们在处理图结构数据时，展现了强大的能力，它们能够通过消息传递机制有效地捕捉节点和边之间的关系。我们将从消息、聚合与更新三个角度对图卷积网络（GCN）、图采样聚合网络（GraphSAGE）、图注意力网络（GAT）和图同构网络（GIN）进行剖析。

4.3.1 图卷积神经网络

在大多数图神经网络的应用场景下，只有少数节点是具有标签信息的，而大多数的节点是未标记的，这也被称为图结构上的**半监督学习**任务。传统的图半监督学习方法假设直接相连的节点将会共享相同的标签，但是考虑到图中的边不一定总是表示节点的相似性，边还能包含额外的信息，因此上述假设会限制模型的表现。为了克服这个问题，图卷积网络（GCN）[5]通过在每一层中聚合来自邻居和自身的特征，不断更新，使得 GCN 能够有效的处理图结构数据。下面是 GCN 的核心公式：

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

1.输入部分

在图卷积网络（GCN）中，输入通常由以下三个部分组成：

(1) 邻接矩阵$\widetilde{\boldsymbol{A}}$：这是图的结构表示，用来描述节点之间的连接关系。对于无向图，$\widetilde{\boldsymbol{A}}$ 是对称的。通常为了防止信息丢失，会将自环（self-loop）添加到邻接矩阵中，形成带有自连接的邻接矩阵。

(2) 节点特征矩阵$\boldsymbol{H}^{(0)}$：每个节点都具有特定的特征，这些特征存储在节点特征矩阵中。矩阵的每一行对应一个节点的特征向量，列数为节点特征的维度。例如，如果每个节点有 5 个特征，且图有 10 个节点，特征矩阵的维度为 10×5。

(3) 度矩阵 $\widetilde{\boldsymbol{D}}$：度矩阵是一个对角矩阵，表示每个节点的度数，即与每个节点相连的边的数量。矩阵的对角线元素 $\widetilde{D}_{ii}$ 表示第 i 个节点的度数。

2.消息算子

在 GCN 中，激活矩阵$\boldsymbol{H}^{(l)}$表示第 l 层节点的特征表示，矩阵的维度为$\mathbf{R}^{\mathrm{N}\times\mathrm{D}}$，其中 N 是图中节点的数量， D 是特征的维度。换个角度理解，每一行代表一个节点的特征，每一列代表特定的特征维度。对于最初的输入数据即$\boldsymbol{H}^{(0)}$代表着节点的初始特征矩阵，其也通常被记为 $\boldsymbol{X}$ 。随着网络的层数增加，节点的特征通过邻居信息的不断聚合和更新，生成新的特征矩阵即$\boldsymbol{H}^{(l+1)}$，投入下一层的进一步聚合。

3.聚合算子

聚合算子的作用是将邻居间的消息进行聚合，在 GCN 中，聚合过程是通过标准化的邻接矩阵实现的。邻接矩阵的标准化具体公式如下：

$$\widehat{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$$

其中 $\boldsymbol{A}$ 代表着邻接矩阵，$\boldsymbol{I}$ 是一个对角矩阵。 $\widetilde{\boldsymbol{A}} = \boldsymbol{A}+\boldsymbol{I}$ 表示在原始的邻接矩阵 $\boldsymbol{A}$ 上添加单位矩阵 $\boldsymbol{I}$，这相当于为图中的每个节点添加一个自环。通过这种方式，GCN 中的消息传递不仅从节点的所有邻居处聚合信息，还能够包含节点自身的特征信息。这一操作确保每个节点在更新自身表示时，能够同时参考邻居的特征和自身的初始特征，从而增强节点表示的表达能力。

对于$\widetilde{\boldsymbol{A}}$，在 GCN 中还需要进行标准化，这是因为$\widetilde{\boldsymbol{A}}$中可能存在度数差异较大的情况，一些节点与许多其他节点相连，而另一些节点的连接较少，直接利用$\widetilde{\boldsymbol{A}}$进行聚合，可能会导致高度数节点对结果产生过大的影响。因此我们需要使用度矩阵$\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$进行标准化。具体的处理方式是$\widehat{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\boldsymbol{D}^{-\frac{1}{2}}$，GCN 分别左乘和右乘了$\boldsymbol{D}^{-\frac{1}{2}}$，实现了行归一化和列归一化，使得不同节点之间的信息传播更加均衡，避免在深层网络中由于度数差异引发的梯度消失或爆炸问题。

4.更新算子

在 GCN 中，更新是通过权重矩阵$\boldsymbol{W}^{(l)}$和非线性激活函数σ实现的。首先，聚合得到的特征矩阵与权重矩阵进行线性变换，然后通过激活函数引入非线性，从而为模型带来更多的表达能力。具体过程为：

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right)$$

可以看到 GCN 利用聚合后的邻居信息生成新的节点特征，并由于非线性的激活函数，节点的特征表示逐渐丰富，逐渐捕捉更多的局部图结构信息和节点自身的特征信息。

5.输出与优化

在图卷积网络中，损失函数通常根据特定任务设定，例如节点分类、回归、链接预测等

任务。通常，损失函数 $\mathcal{L}$ 表示模型预测结果 $\boldsymbol{H}^{(L)}$ 和目标结果 Y 之间的差异，形式可以为：

$$\mathcal{L}\left(\boldsymbol{H}^{(L)}, \boldsymbol{Y}\right)$$

其中 $\boldsymbol{H}^{(L)}$ 是网络最后一层的输出，$\boldsymbol{Y}$ 是目标标签或期望的输出。

计算完损失函数之后，通过反向传播算法计算损失相对于每层可训练参数（如权重矩阵 $\boldsymbol{W}^{(l)}$）的梯度。 在 GCN 中，主要的可训练参数是每一层的权重矩阵 $\boldsymbol{W}^{(l)}$。可以通过链式法则计算损失函数对权重的梯度：$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(l)}}$，并利用优化算法（如梯度下降、Adam 等）对模型参数进行更新。对于每一层的权重矩阵 $W^{(l)}$，更新公式为：

$$\boldsymbol{W}^{(l)} \leftarrow \boldsymbol{W}^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(l)}}$$

其中，η 是学习率，用于控制更新的步长。

4.3.2 图采样聚合网络

传统的图节点嵌入方法（如图卷积网络，GCN）大多数是**传导式**(Transductive)的，即训练和推理都依赖于整个图的全图结构，即所有的节点和邻居在训练时必须是已知的。换句话说，模型只能处理训练期间看到的节点，一旦有新的节点加入，模型必须重新进行训练。传导式的局限性导致这类神经网络在大规模图上效率较低。图采样聚合网络（GraphSAGE）[6]为了解决上述问题，采用了**归纳式**（Inductive)的设计，通过采样和聚合邻居节点的特征信息来学习节点的嵌入，使得模型可以对未见过的节点进行嵌入。GraphSAGE 不为每个节点单独训练嵌入向量，而是训练一组**聚合函数**（Aggregator)，通过从节点的局部邻域采样和聚合特征信息聚合生成嵌入。

1.输入部分

对于 GraphSAGE 网络，输入主要由四部分组成：

1) 图 $G(V, E)$

其中 V 是节点集合，E 是边集合，表示图的结构。

2) 节点特征$\{\boldsymbol{x}_v\}$

每个节点 v 对应的输入特征向量 $\boldsymbol{x}_v$，这些特征可以是任何相关属性（例如文本、图像或其他特征），这些输入特征是算法最初用来生成节点嵌入的基础。

3) 深度K

用于控制在多远的邻居中聚合特征。通常，K 表示搜索深度或模型中层数，即模型将从 K 跳邻居中获取和聚合特征。

4) 聚合函数 $\{AGGREGATE_k\}$

每层都有一个可微的聚合器，用于聚合节点邻居的信息，例如均值聚合、池化或 LSTM 聚合。

2.消息算子

消息算子的核心作用是对节点的特征进行处理，以便适应后续的聚合过程。GraphSAGE 在消息传递过程中，为了保证特征维度的一致性和增强模型的表达能力，通常需要对节点的特征进行维度调整。

具体而言，对于每一层 k，每个节点 $v \in V$ 的特征表示从上一层的嵌入 $\boldsymbol{h}_v^{(k-1)} \in \mathbf{R}^{d_{k-1}}$ 转换为当前层的目标维度 d_k。这一调整通过一个可训练的线性变换实现，公式为：

$$\boldsymbol{h}_v^{(k)} \rightarrow \boldsymbol{h}_v^{(k-1)} \boldsymbol{W}_k,$$

其中：$\boldsymbol{h}_v^{(k-1)} \in \mathbf{R}^{d_{k-1}}$ 是节点 v 在第 $k-1$ 层的特征表示；$\boldsymbol{W}_k \in \mathbf{R}^{d_{k-1} \times d_k}$ 是第 k 层的权重矩阵，用于调整特征的维度；$\boldsymbol{h}_v^{(k)} \in \mathbf{R}^{d_k}$ 是调整维度后的特征表示。

3.聚合算子

如图 4-9 所示，为了处理大规模图数据，GraphSAGE 通过采样邻居节点来降低计算复杂度。对于每一层 k，GraphSAGE 节点 v 从其邻居节点 $N(v)$ 中采样一个固定数量 S_k 的节点进行信息聚合，这里的采样是等概率的。定义采样集合为：

$$N_k(v) = Sample(N(v), S_k),$$

其中 $N(v)$ 是节点 v 的所有邻居，$\text{Sample}(N(v), S_k)$ 表示从 $N(v)$ 中随机采样 S_k 个邻居节点。

在实际实现中，GraphSAGE 从“目标”节点开始进行反向采样： 1. 首先从目标节点找到其直接邻居（第 K 层，深度为 1 的节点），记为 $N_K(v)$； 2. 然后递归地对这些邻居节点找到它们的邻居（第 $K-1$ 层），记为 $N_{K-1}(v)$； 3. 按此过程逐层采样直至第 1 层。

对于每一层 k，仅对采样集合 $N_k(v)$ 的节点进行聚合，从而显著减少计算量。这种反向采样方式确保了在每一层中只计算需要的信息，从而使得每批次的时间和空间复杂度固定为：

$$O\left(\prod_{i=1}^{K} S_i\right),$$

其中 S_i 表示每一层的采样大小，K 是网络的深度。

这种方法有效减少了节点度数偏差对计算复杂度的影响，同时通过限制每层的采样大小 S_k，大幅提升了训练速度。实践中，当 $K = 2$ 且 $S_1 \cdot S_2 \leq 500$ 时，GraphSAGE 能够取得较优的性能。

而 GraphSAGE 的聚合过程本质上是一个逐层聚合邻居嵌入信息的过程，与采样过程的顺序相反。它从最远的 k 阶邻居的嵌入信息开始，逐层向目标节点聚合，直到最终生成目标节点的嵌入。这种分层聚合的设计使得每个节点的嵌入不仅包含其直接邻居的信息，还能反映出其多阶邻居的结构和特征信息。聚合过程中的聚合函数（Aggregator)有多种，可依据不同的任务类型进行选择。

1) Mean aggregator

$$\boldsymbol{h}_{N(v)}^{k} = MEAN\left(\left\{\boldsymbol{h}_{u}^{k-1}, \forall u \in N(v)\right\}\right)$$

2) Max pooling aggregator

$$\boldsymbol{h}_{N(v)}^{k} = \max\left(\left\{\sigma\left(\boldsymbol{W}_{\text{pool}} \cdot \boldsymbol{h}_{u}^{k-1} + \boldsymbol{b}_{\text{pool}}\right), \forall u \in N(v)\right\}\right)$$

3) LSTM aggregator

$$\boldsymbol{h}_{N(v)}^{k} = LSTM\left(\left\{\boldsymbol{h}_{u}^{k-1}, \forall u \in N(v)\right\}\right)$$

通过上述聚合器，$\boldsymbol{h}_{\boldsymbol{N(v)}}^{\boldsymbol{k}}$中保留来自 k 阶邻居的聚合信息。此外，对于任意一个新节点，我们可以方便的使用其邻居信息与聚合器得到新节点的嵌入表示，而无须重新训练整个模型。

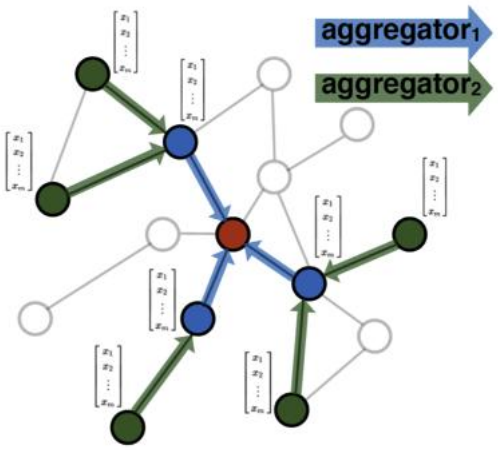

图 4-9 图采样聚合网络聚合示意图

4.更新算子

在更新算子部分，图采样网络将节点 v 的上一层嵌入表示 $\boldsymbol{h}_v^{k-1}$ 与其邻居节点的聚合表示 $\boldsymbol{h}_{N(v)}^k$ 进行拼接。通过这一拼接操作，节点不仅保留了自己上一层的特征信息，还有效地融合了来自邻居节点的聚合信息。这一步使得每个节点可以在更新时同时考虑自身特性和其局部邻域的结构信息。

拼接后的特征向量接着通过权重矩阵 $\boldsymbol{W}^k$进行线性变换，再经过非线性激活函数 σ 的处理。这个过程不仅调整了特征向量的维度，还引入了非线性因素，增强了模型对复杂数据的表达能力。这种更新方式使节点的嵌入表示能够更好地适应下游任务（如节点分类、聚类等）的需求，提升模型的表现力。

$$\boldsymbol{h}_v^k \leftarrow \sigma\left(\boldsymbol{W}^k \cdot CONCAT\left(\boldsymbol{h}_v^{k-1}, \boldsymbol{h}_{N(v)}^k\right)\right)$$

5.输出与优化

GraphSAGE 的损失函数用于度量模型生成的嵌入向量与目标结果之间的差距。设模型输出的嵌入表示为 $Z = \{z_v | v \in V\}$，其中每个 z_v 表示图中节点 v 的嵌入。假设我们有一组目标标签 $Y = \{y_v | v \in V\}$，其中 y_v 是节点 v 的真实标签，损失函数 $\mathcal{L}(Z, Y)$ 表示了嵌入结果 Z 与目标标签 Y 之间的误差。

损失函数 $\mathcal{L}$ 通常定义为交叉熵损失或均方误差损失，以衡量模型在分类或回归任务中的表现。例如，在节点分类任务中，$\mathcal{L}$可以定义为交叉熵损失：

$$\mathcal{L}(Z, Y) = -\sum_{v \in V}\sum_{c=1}^{C} y_v^{(c)} \, log\left(\hat{y}_v^{(c)}\right)$$

公式中的 Z 代表模型生成的嵌入集，其中每个 z_v 表示图中节点 v 的嵌入；Y 是目标标签集，其中每个 $\mathrm{y_v}$ 表示节点 v 的真实类别标签。V 是图中的节点集合，C 表示可能的类别总数，$y_v^{(c)}$ 是指示变量，当节点 v 属于类别 c 时 $y_v^{(c)} = 1$，否则为 0，而 $\hat{y}_v^{(c)}$ 是模型预测节点 v 属于类别 c 的概率。该公式对所有节点和类别的预测概率与真实标签进行交叉熵损失计算，通过求和来衡量模型预测的类别概率与真实标签之间的偏差。

通过计算损失函数后，接下来使用反向传播算法计算每一层中各个参数的梯度。通过这个过程，模型可以知道如何调整每一层的参数来减少损失函数的值。

4.3.3 图注意力网络

图注意网络（Graph Attention Network）[7]通过在图神经网络中引入自注意力机制，自适应地为节点邻居分配不同的权重，以此增强节点表示的学习。与传统的图神经网络依赖于预定义的邻居聚合策略不同，GAT 能够自动识别并放大对目标节点更重要的邻居信息，从而更灵活地处理复杂图结构中的信息异质性。此外，GAT 采用多头注意力机制来增强模型的鲁棒性，使得其在多个图学习任务中取得了显著的性能提升，特别是在节点分类和链路预测等任务中展现了较高的准确率和泛化能力。

1. 输入部分

GAT 的输入是一组节点特征，记为$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}, \vec{h}_i \in \mathbf{R}^F$，其中 N 是节点的个数，F 是每个节点的特征维度。

2. 消息算子

为了将输入特征转换为高维特征（增维）以获得足够的表达能力，每个节点将经过一个共享的线性变换，该模块的参数用权重矩阵$\mathbf{W} \in \mathbf{R}^{F' \times F}$来表示，对某个节点 i 做线性变换可表示为$\mathbf{W}\vec{h}_i \in \mathbf{R}^{F'}$。

3. 聚合算子

如图 4-10 所示，在每两个节点 i、j 之间我们将用自注意力机制来计算节点特征之间的重要性。自注意力集机制：$\mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$，节点 i 和 j 之间的自注意力系数为：

$$e_{ij} = a(W\vec{h}_i, W\vec{h}_j)$$

该系数表示了节点 i 的特征对节点 j 的重要性。注意力机制α()是一个单层前馈神经网络，用一个权重向量来表示：$\vec{a} \in \mathbb{R}^{2F'}$，它把拼接后的长度为 $2F'$的高维特征映射到一个实数上，作为注意力系数。 很明显，对于一个节点 i 的所有邻居节点 j，他们与 i 之间的注意力系数之和应该为 1，因此 GAT 使用 softmax()函数对所有对于节点 i 的注意力系数进行归一化：

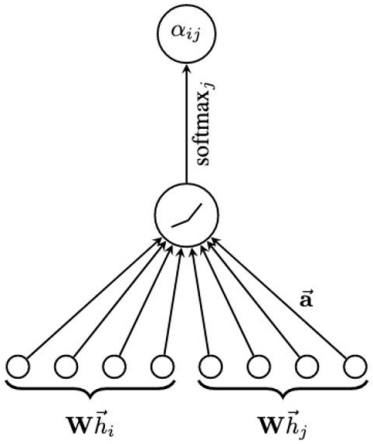

图 4-10 注意力机制示意图

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp\,(e_{ik})}$$

整理上述公式，并加入 leakyReLU 非线性激活函数，我们就得到了两节点间的注意力

系数：

$$\alpha_{ij} = \frac{exp\left(LeakyReLU\left(\vec{\boldsymbol{a}}^{\mathrm{T}}\left[\boldsymbol{W}\vec{\boldsymbol{h}}_i \parallel \boldsymbol{W}\vec{\boldsymbol{h}}_j\right]\right)\right)}{\sum_{k\in\mathcal{N}_i} exp\left(LeakyReLU\left(\vec{\boldsymbol{a}}^{\mathrm{T}}\left[\boldsymbol{W}\vec{\boldsymbol{h}}_i \parallel \boldsymbol{W}\vec{\boldsymbol{h}}_k\right]\right)\right)}$$

4. 更新算子

GAT 在得到归一化的注意力系数后，可以通过对邻接节点特征的线性组合，并经过一个非线性激活函数得到节点的新的输出：

$$\vec{\boldsymbol{h}}_{\boldsymbol{i}}' = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\boldsymbol{W}\vec{\boldsymbol{h}}_j\right)$$

GAT 为了稳定自注意力的学习过程，还使用了多头注意力机制。具体地，使K 个独立注意力机制根据上式进行变换，得到更新的节点输出特征，然后将K 个特征拼接，得到如下输出特征：

$$\vec{\boldsymbol{h}}_{\boldsymbol{i}}' = \parallel_{k=1}^{K}\sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{k}\mathbf{W}^{k}\vec{\boldsymbol{h}}_j\right)$$

如果我们在最后一层使用注意力机制，就需要将输出取均值，并使用 softmax（）或 sigmoid()函数，具体如下：

$$\vec{\boldsymbol{h}}_{\boldsymbol{i}}' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i} \alpha_{ij}^{k}\mathbf{W}^{k}\vec{\boldsymbol{h}}_j\right)$$

—

5. 输出与优化

GAT 的输出是一组新的节点特征，记为$\boldsymbol{h}' = \{\vec{h}_1', \vec{h}_2', ..., \vec{h}_N'\}, \vec{h}_i' \in \mathbb{R}^{F'}$，其中每个节点 $\vec{h}_i' \in \mathbb{R}^{F'}$ 是经过聚合和多头注意力计算得到的新的高维节点嵌入。

随后，我们可以采用与先前 GCN、GraphSAGE 相似的方式来定义和应用损失函数，用以计算模型输出与目标标签之间的误差。具体而言，假设我们有一个目标标签集 $Y = \{y_v | v \in V\}$，其中 y_v 表示图中节点 v 的真实类别标签。损失函数通常可以选择交叉熵损失或均方误差损失等形式，以衡量模型输出嵌入与真实标签之间的偏差。例如，对于节点分类任务，我们可以定义交叉熵损失函数：

$$\mathcal{L}(H', Y) = -\sum_{v\in V}\sum_{c=1}^{C} y_v^{(c)} log\left(\hat{y}_v^{(c)}\right)$$

其中 $\hat{y}_v^{(c)}$ 是模型预测节点 v 属于类别 c 的概率，$y_v^{(c)}$ 是节点 v 的真实标签（使用指示变量表示：当 v 属于类别 c 时，$y_v^{(c)} = 1$；否则 $y_v^{(c)} = 0$）。通过最小化该损失函数，我们可以有效地度量模型输出与真实标签的差距。

在计算损失函数后，我们可以利用链式法则对模型的参数进行反向传播更新，以最小化损失函数。GAT 的训练过程会逐步调整注意力机制的权重和节点特征的投影矩阵，从而使模型学习到符合输入图结构和特征分布的节点嵌入表示。通过这种方式，GAT 能够在保留图结构信息的基础上生成更具辨识度的节点特征，用于下游任务如节点分类或链接预测等。

4.3.4 图同构网络

在介绍图同构网络(Graph Isomorphism Network)[8]前，我们首先需要介绍**Weisfeiler-Lehman(WL)算法**：对于任意的节点$v_i \in \mathcal{G}$

(1) 获取节点$\mathbf{V}_i$的所有的邻居节点 j 的特征$\boldsymbol{h}_{\mathbf{v}_j}$。

(2) 更新该节点的特征$\boldsymbol{h}_i \leftarrow hash\left\{\sum_j \quad \boldsymbol{h}_{\boldsymbol{v}_j}\right\}$，其中 hash 是一个单射函数。

(3) 重复以上步骤 K 次。

事实上，**Weisfeiler-Lehman 算法**在大多数图上会得到一个独一无二的特征集合，这意味着图上的每一个节点都有着独一无二的角色定位）。因此，对于大多数非规则的图结构，得到的特征可以作为图是否同构的判别依据。

由此我们可以总结 WL 算法，如 4-11 所示：

$$\boldsymbol{a}_v^k = f\left(\left\{\boldsymbol{h}_u^{k-1}: u \in N(v)\right\}\right)$$
$$\boldsymbol{h}_v^k = Hash\left(\boldsymbol{h}_v^{k-1}, \boldsymbol{a}_v^k\right)$$

对于经典图神经网络，我们以可以概括为：

$$\boldsymbol{a}_v^{(k)} = AGGREGATE^{(k)}\left(\left\{\boldsymbol{h}_u^{(k-1)}: u \in \mathcal{N}(v)\right\}\right), \quad \boldsymbol{h}_v^{(k)} = COMBINE^{(k)}\left(h_v^{(k-1)}, \boldsymbol{a}_v^{(k)}\right)$$

可以发现 GNN 和 WL 公式，实际上是非常相似的：当两跳聚合的时候，都是先由其二阶邻居信息聚合得到一阶邻居节点信息，再由一阶邻居信息聚合得到自己，但是 WL 测试是由哈希函数聚合，GNN 是由池化操作来聚合。

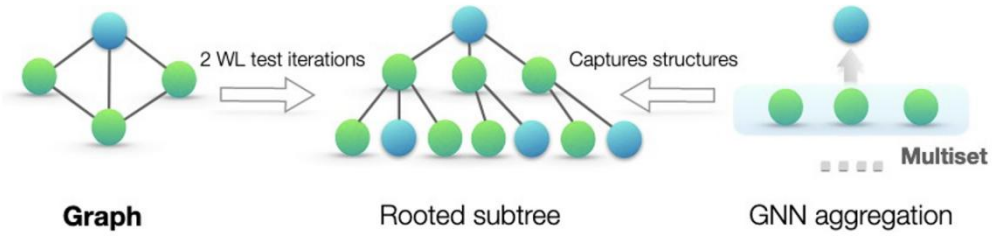

图 4- 11 Weisfeiler-Lehman(WL）算法示意图

我们假设节点的输入特征来自一个可数的集合。对于有限的图，任何固定模型的深层节点特征向量也来自于这个可数集合。为了简化表示，我们可以为每个特征向量分配一个唯一的标签（如 a, b, c 等）。这样，邻居节点的特征向量将构成一个多重集，其中某些特征可能出现多次，因为不同节点可能具有相同的特征向量。 我们定义多重集是一种允许元素重复出现的集合的广义概念。更正式地，多重集是一个二元组 X = (S, m)，其中 S 是由不同元素构成的基础集合，$m : S \rightarrow N_{\geq 1}$。

由此我们可以发现 GNN 的分类上限是 WL 算法，因为一个 GNN 的能力强弱取决于能否将两个多重集映射到不同的不同的表示。最强大的 GNN 聚合模式将会是单射的，GIN 就是一个单射的表达能力理论上最强的图神经网络。

1.输入部分

GIN 的输入主要包括以下要素：

(1) 节点特征 $\boldsymbol{h}_v^{(0)}$ ：每个节点 v 的初始特征表示，通常是一个维度为 F 的向量。这些特征可以是节点本身的属性或预先设定的初始特征，如节点的度或类别信息。

(2) 图结构 G = (V, E) ：这是图的基本结构，其中 V 是节点集合， E 是边集合。每个节点通过与其邻居的连接来进行信息传播和聚合。

2.消息算子

GIN 的输入是一组节点特征，记为$\boldsymbol{h} = \{\vec{h}_1, \vec{h}_2, ..., \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$，其中 N 是节点的个数，F 是每个节点的特征维度。

3.聚合算子

如图 4-12 所示，就像前面所解释的，表达能力越强大的 GNN 方法可以根据图拓扑结构的不同生成嵌入，因此我们的聚合算子应当是个单射函数。我们对比常见的三种聚合方式：sum()、mean()、max()：

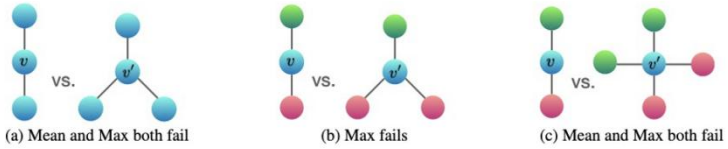

图 4-12 聚合算子示意图

如图 4-13 所示，观察图中内容我们可以发现：sum()学习全部的标签及数量，可以学习精确的结构信息，mean()更多的是学习标签的比例，偏向于学习分布信息，而 max()则是学习具有最大代表性的元素信息。因此我们可以得出结论：对于不同的多重集，sum()是单射的，而 mean()和 max()从本质上来说不是单射的。在 GIN 中我们会采用 SUM()作为聚合函数。

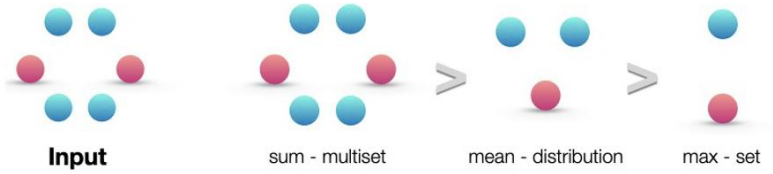

图 4-13 聚合算子示意图 2

对于全体多重集$\mathcal{X}$汇聚算子为：

$$h(c, X) = (1 + \epsilon) \cdot f(c) + \sum_{x \in X} f(x), where\ c \in X\ and\ X \subseteq \mathcal{X}$$

4.更新算子

我们使用 MLP(多层感知机）作为我们的更新算子，因为其强大的函数近似能力有效学习复杂的特征转换和聚合操作。MLP 能够表示函数的组合，并提供足够的非线性能力来捕捉图结构中的复杂关系。GIN 的更新算子表示如下，其中ϵ 是一个可学习参数或一个固定的标量：

$$\boldsymbol{h}_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot \boldsymbol{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \boldsymbol{h}_u^{(k-1)}\right)$$

5.输出与优化

在 GIN 模型中，图级别的任务（如图分类）要求从图中所有节点的特征生成整个图的表示。为此，GIN 使用图级别的汇总（readout）函数，该函数将每个节点的嵌入聚合成整个

图的嵌入。

GIN 采用的图级汇总操作为：

$$\boldsymbol{h}_G = CONCAT\left(READOUT\left(\left\{\boldsymbol{h}_v^{(k)} | v \in G\right\}\right) | k = 0,1,\ldots,K\right)$$

其中 $h_v^{(k)}$ 是第 k 层的节点嵌入，READOUT 函数是汇总节点特征的方法，可以是如求和等不依赖节点顺序的操作。这种设计能够确保 GIN 在不同深度的迭代中捕获图的结构信息，使得最终图的表示包含了不同深度的局部和全局信息。

以图分类任务为例，模型输出的图级嵌入 $\boldsymbol{h_G}$ 会通过一个 MLP 层生成分类结果，与真实标签比较，使用交叉熵计算损失。其公式为：

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \, log(\hat{y}_i)$$

其中 y_i 是真实标签， $\hat{y}_i$ 是 GIN 模型预测的类别概率。 随后通过优化算法（如 Adam 或 SGD）更新模型的参数。通过这一流程，GIN 可以在图分类任务中逐步优化其权重，使得模型能够更好地捕捉图的结构信息并提高预测的准确性。

4.4 小结

本章概述了图神经网络的基本原理及其在处理图结构数据中的独特优势。通过介绍通用框架、核心算子及其设计方法，深入解析了图神经网络的发展脉络。以图卷积网络和图采样聚合网络等经典模型为例，分析了不同模型在消息传递算子和图池化算子上的设计差异及应用特点。同时，本章还探讨了图神经网络在节点级别和图级别任务中的训练方法，为读者全面理解和应用图神经网络奠定了基础。

引言部分讨论了传统图结构学习方法的局限性，突出图神经网络在特征提取和关系建模方面的优势，并对其核心概念及广泛的应用场景进行了定义和阐释。此外，还简要概述了本章节的主要内容。第一节聚焦于图神经网络的基础知识，系统介绍了通用框架、核心算子的构成以及图神经网络的发展历程。第二节详细探讨了图神经网络核心算子的设计方法，不同算子的选择使得图神经网络能够灵活适配于图结构数据的多种任务需求。第三节深入分析了经典图神经网络的设计与优化，包括图卷积网络（GCN）、图采样网络（GraphSAGE）、图注意力网络（GAT）以及图同构网络（GIN），并剖析了各自的特点和应用场景。

拓展阅读

(1) Gasteiger J, Bojchevski A, Günnemann S. Predict then propagate: Graph neural networks meet personalized pagerank[J]. ICLR 2019

(2) Wang X, Ji H, Shi C, et al. Heterogeneous graph attention network[C]//The world wide web conference. 2019: 2022-2032.

(3) Wang X, Zhu M, Bo D, et al. Am-gcn: Adaptive multi-channel graph convolutional networks[C]//Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery data mining. 2020: 1243-1253.

(4) Bo D, Shi C, Wang L, et al. Specformer: Spectral graph neural networks meet transformers[J]. ICLR 2023

(5) Zhang M, Sun M, Wang P, et al. GraphTranslator: Aligning Graph Model to Large Language Model for Open-ended Tasks[C]//Proceedings of the ACM on Web Conference 2024. 2024: 1003- 1014.

习题

(1) 以下哪种图池化方法适用于生成整个图的固定维度特征向量，并不依赖于图的规模？

A. 全局池化（Global Pooling）

B. 层次池化（Hierarchical Pooling）

C. 局部池化（Local Pooling）

D. 自适应池化（Adaptive Pooling）

(2) 以下哪种聚合方法能够保证图同构网络（GIN）具有最强的表达能力？

A. 平均聚合（Mean）

B. 最大池化聚合（Max Pooling）

C. 求和聚合（Sum）

D. 最小池化聚合（Min Pooling）

(3) 图采样聚合网络（GraphSAGE）主要采用哪种策略来提升其在未见节点上的嵌入能力？

A. 全局聚合

B. 层次化聚合

C. 采样聚合

D. 注意力聚合

(4) 图神经网络（GNN）如何通过消息传递机制来捕捉节点的局部和全局邻域信息？请解释消息传递机制的过程，并简述其在 GNN 中的作用。

(5) 图池化在图神经网络中的作用是什么？有哪些常见的图池化方法？

(6) 图注意力网络（GAT）是如何利用注意力机制解决节点度数差异的问题的？说明图注意力网络中如何通过注意力机制来分配不同邻居节点的重要性，并讨论其对节点信息聚合的影响。

(7) 请解释图卷积网络（GCN）在处理图结构数据时如何更新节点特征，并说明其核心公式。

(8) 请简述 Weisfeiler-Lehman 算法与图同构网络（GIN）之间的联系，并说明为什么 GIN 可以用来判断图是否同构。

(9) 假设有一个图，其中三个节点的嵌入表示分别为：

节点 v_1 的嵌入表示为 $\mathbf{h}_1 = [0.4,0.6]$

节点 v_2 的嵌入表示为 $\mathbf{h}_2 = [0.1,0.9]$

节点 v_3 的嵌入表示为 $\mathbf{h}_3 = [0.7,0.2]$

请你使用全局池化中的求和操作，计算并得出整个图的全局特征向量。

(10) 在图卷积网络（GCN）中，考虑一个包含四个节点的图 A, B, C, D，节点的初始特征分别为 $\mathbf{h}_A = [1,0], \mathbf{h}_B = [0,1], \mathbf{h}_C = [1,1], \mathbf{h}_D = [0,0]$。图的邻接矩阵 A 为：

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

① 使用带有自环的标准化邻接矩阵来对节点特征进行一次更新。请计算出每个节点的新特征。

② 使用以下权重矩阵进行更新：

$$W = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

计算出更新后的每个节点的新特征。

参考资料

[1] Zhou J, Cui G, Hu S, et al. Graph neural networks: A review of methods and applications[J]. AI open, 2020, 1: 57-81.

[2] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks[J]. IEEE transactions on neural networks and learning systems, 2020, 32(1): 4-24.

[3] Li Z, Liu F, Yang W, et al. A survey of convolutional neural networks: analysis, applications, and prospects[J]. IEEE transactions on neural networks and learning systems, 2021, 33(12): 6999-7019.

[4] Yu Y, Si X, Hu C, et al. A review of recurrent neural networks: LSTM cells and network architectures[J]. Neural computation, 2019, 31(7): 1235-1270.

[5] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907, 2016.

[6] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.

[7] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks[J]. arXiv preprint arXiv:1710.10903, 2017.

[8] Xu K, Hu W, Leskovec J, et al. How powerful are graph neural networks?[J]. arXiv preprint arXiv:1810.00826, 2018.

[9] Veličković P, Fedus W, Hamilton W L, et al. Deep graph infomax[J]. arXiv preprint arXiv:1809.10341, 2018.

[10] Wang X, Ji H, Shi C, et al. Heterogeneous graph attention network[C]//The world wide web conference. 2019: 2022-2032.

[11] Zhang C, Song D, Huang C, et al. Heterogeneous graph neural network[C]//Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019: 793-803.

[12] Gao Y, Yang H, Zhang P, et al. Graph neural architecture search[C]//International joint conference on artificial intelligence. International Joint Conference on Artificial Intelligence, 2021.

[13] Hafidi H, Ghogho M, Ciblat P, et al. Graphcl: Contrastive self-supervised learning of graph representations. arXiv 2020[J]. arXiv preprint arXiv:2007.08025, 2020.

[14] Ying C, Cai T, Luo S, et al. Do transformers really perform badly for graph representation?[J]. Advances in neural information processing systems, 2021, 34: 28877-28888.

[15] Rampášek L, Galkin M, Dwivedi V P, et al. Recipe for a general, powerful, scalable graph transformer[J]. Advances in Neural Information Processing Systems, 2022, 35: 14501-14515.

[16] Hu Z, Dong Y, Wang K, et al. Gpt-gnn: Generative pre-training of graph neural networks[C]//Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020: 1857-1867.

[17] Hou Z, Liu X, Cen Y, et al. Graphmae: Self-supervised masked graph autoencoders[C]//Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022: 594-604.

[18] Jing B, Eismann S, Suriana P, et al. Learning from protein structure with geometric vector perceptrons[J]. ICLR 2021

[19] Brandstetter J, Hesselink R, van der Pol E, et al. Geometric and physical quantities improve e (3) equivariant message passing[J]. ICLR 2022

[20] Han J, Moraga C. The influence of the sigmoid function parameters on the speed of backpropagation learning[C]//International workshop on artificial neural networks. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995: 195-201.

[21] Fan E. Extended tanh-function method and its applications to nonlinear equations[J]. Physics Letters A, 2000, 277(4-5): 212-218.

[22] Agarap A F. Deep learning using rectified linear units (relu)[J]. arXiv preprint arXiv:1803.08375, 2018.

[23] De Boer P T, Kroese D P, Mannor S, et al. A tutorial on the cross-entropy method[J]. Annals of operations research, 2005, 134: 19-67.

[24] Hecht-Nielsen R. Theory of the backpropagation neural network[M]//Neural networks for perception. Academic Press, 1992: 65-93.

[25] Ying Z, Bourgeois D, You J, et al. Gnnexplainer: Generating explanations for graph neural networks[J]. Advances in neural information processing systems, 2019, 32.

[26] Zhang M, Wang X, Zhu M, et al. Robust heterogeneous graph neural networks against adversarial attacks[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2022, 36(4): 4363-4370.