北京郵電大學
Beijing University of Posts and Telecommunications

第四章

图神经网络初步

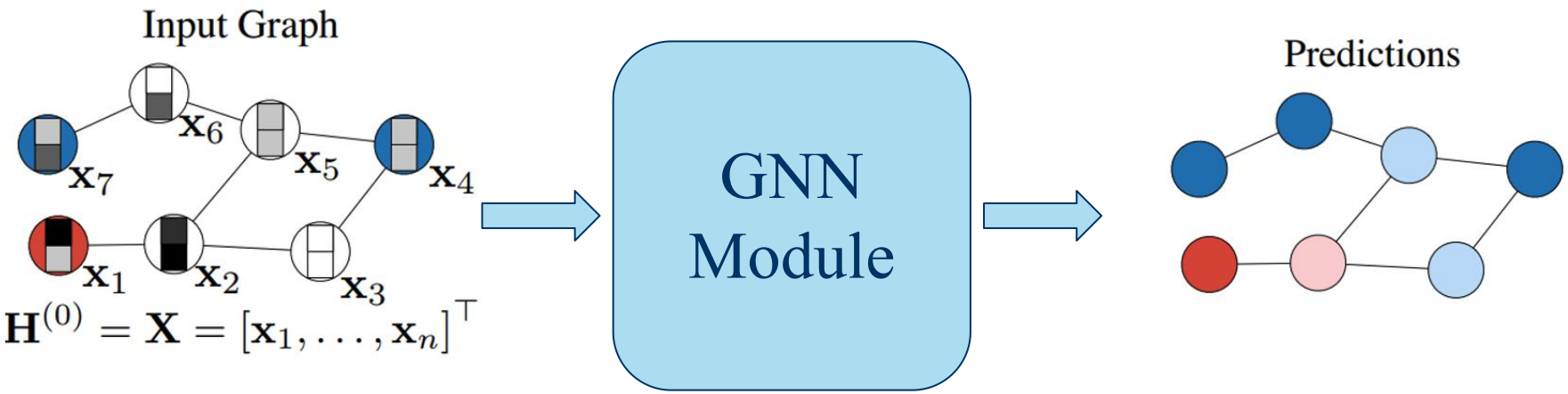
石川 教授
数据科学与服务中心 计算机学院

内容提纲

- 课程内容
  - 图神经网络基础
    - 通用图神经网络框架
    - 消息传递算子
    - 图池化算子
  - 经典图神经网络
    - 图卷积网络，图采样聚合网路，图注意力网络，图同构网络
  - 图神经网络的训练
    - 针对节点级别任务的训练
    - 针对图级别任务的训练

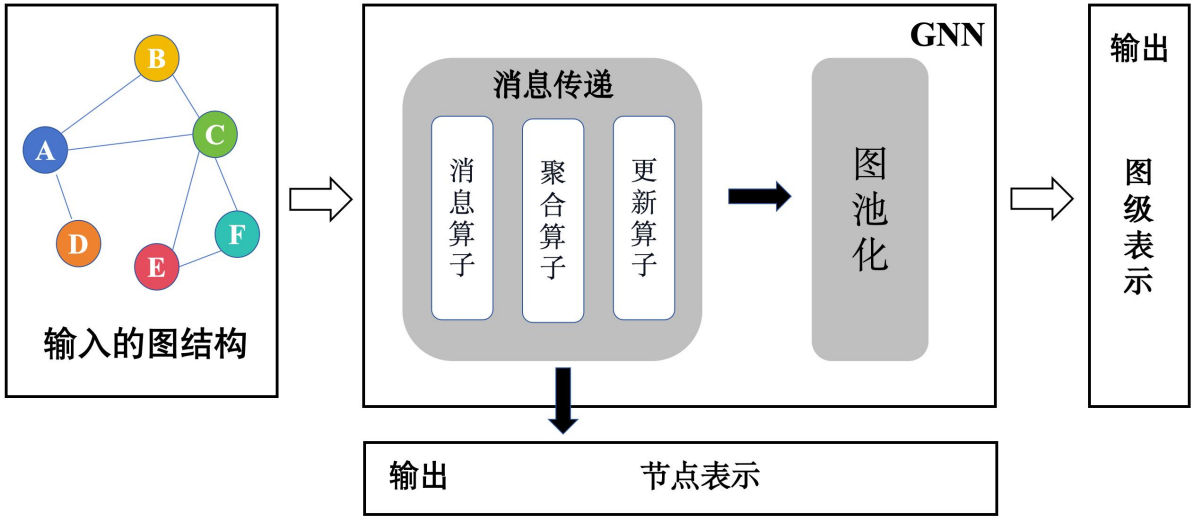
4.1 图神经网络基础

- 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。
- 输入图结构 $\boldsymbol{A}$ 和节点属性 $\boldsymbol{X}$



4.1.1 通用图神经网络框架

- GNN通用框架由**消息传递算子**和**图池化算子**组成。

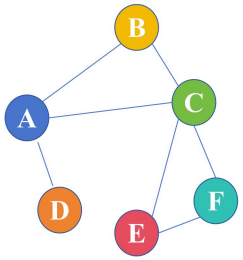


- GNN 通过**消息传递算子**来传播和聚合信息，使得每个节点的表示隐含了图中邻居节点及其关系的综合信息。

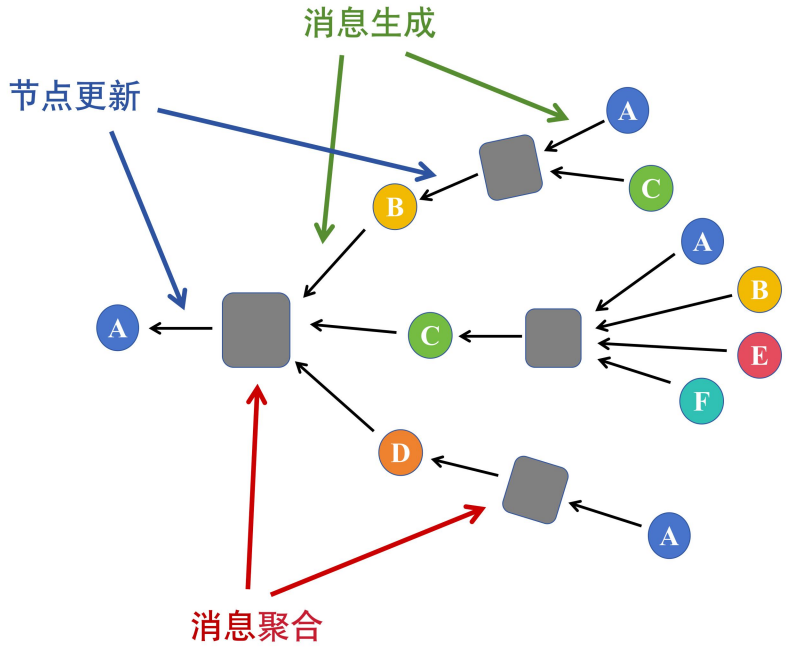
- **图池化算子**通过对图中节点或边进行降维处理，将其信息压缩为全局性的嵌入表示，从而更好地捕捉图的全局特征

4.1.2 消息传递算子

- 消息传递（Message Passing）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表示。
- 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交互生成消息，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。



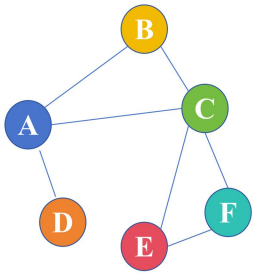
输入的图结构



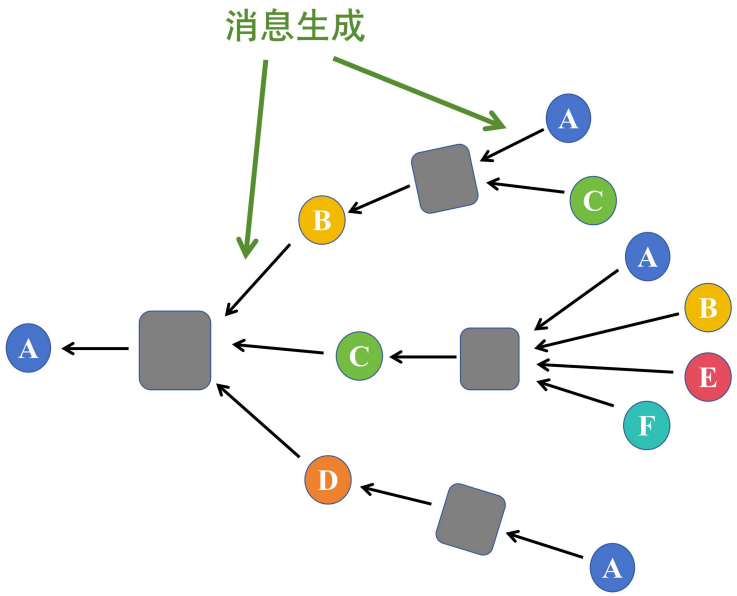
4.1.2 消息传递算子

- **消息生成** 定义：给定图$G=(V,E)$，假设节点 V_i在第 k 轮迭代时的嵌入表示为 $h_i^{(k)}$，ϕ是定义消息生成过程的函数，e_{ij}是边特征则消息算子生成的消息$m_{ij}^{(k)}$从节点v_j到节点v_i的公式可以表示为：

$$m_{ij}^{(k)} = \phi(h_j^{(k)}, e_{ij})$$



输入的图结构



4.1.2 消息传递算子

- **消息生成** 一般有两种常见类型：线性变换、非线性变化。
- 线性变换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信息映射为信息，令W表示权重矩阵，b表示偏置项，则生成的消息可以表达为：

$$m_{ij}^{(k)} = W \cdot h_j^{(k)} + b$$

- 非线性变换通过引入非线性激活函数，增强模型的表达能力，σ 为非线性激活函数(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。公式可以表示为：

$$m_{ij}^{)k(} = \sigma)W \bullet h_j^{)k(}, e_{ij}($$

4.1.2 消息传递算子

- 示例：给定图$G=(V,E)$，包含三个节点：$V=\{v_1, v_2, v_3\}$，邻接矩阵$A=\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}$，初始特征矩阵$H=\begin{bmatrix}1.0 & 0.5\\ 0.2 & 0.3\\ 0.4 & 0.7\end{bmatrix}$，权重矩阵 $W=\begin{bmatrix}0.5 & 0.5\\ 0.1 & 0.9\end{bmatrix}$，偏置向量$b=[0.1 \quad 0.2]$。
  - 消息算子 生成从节点 v_j 传递到节点 v_i 的消息：
  - 线性变换：$m_{ij}^{)k)}=W\cdot h_j^{(k)}+b$，$m_{12}=W\cdot\begin{bmatrix}0.2\\ 0.3\end{bmatrix}+\begin{bmatrix}0.1\\ 0.2\end{bmatrix}=\begin{bmatrix}0.35\\ 0.49\end{bmatrix}$
  - 非线性变换加入激活函数 σ，例如ReLU：

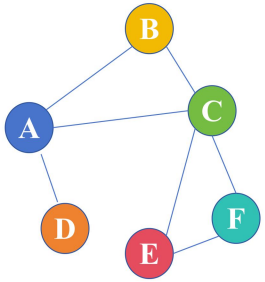
$$m_{12}=\mathrm{ReLU}(W\cdot h_2+b)$$

$$m_{12}=\mathrm{ReLU}(\begin{bmatrix}0.35\\ 0.49\end{bmatrix}=(\ \begin{bmatrix}0.35\\ 0.49\end{bmatrix}$$

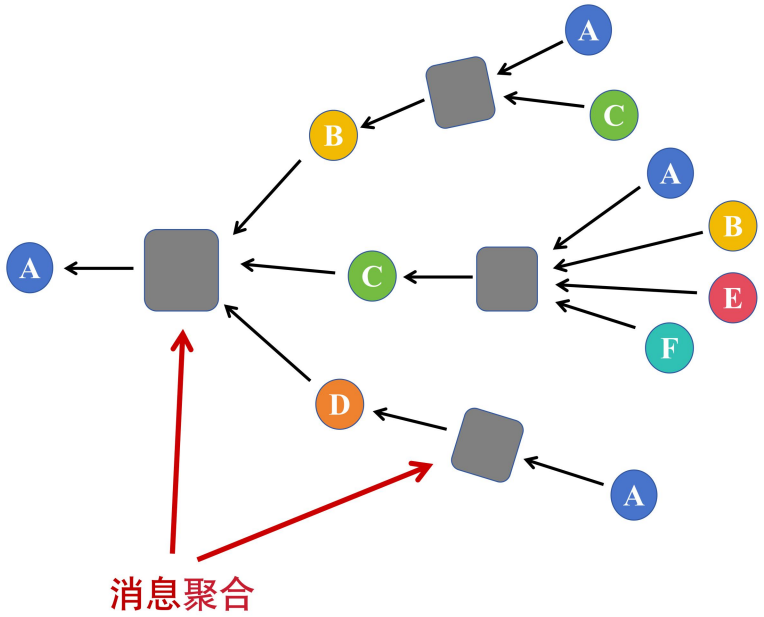
4.1.2 消息传递算子

- **消息聚合** 定义：给定图 $G = (V, E)$，假设节点 v_i 的邻居节点集合为 $N(v_i)$，聚合算子用于将节点v_i从邻居节点v_j接收到的所有消息$m_{ij}^{(k)}$进行整合，形成一个综合的消息表示$m_i^{(k)}$，令 φ 是聚合函数，该过程可以表示为：

$$m_i^{(k)} = \varphi(\{m_{ij}^{(k)} | v_j \in N(v_i)\})$$



输入的图结构



4.1.2 消息传递算子

- **聚合算子** 一般有五常见的类型，最小值聚合、最大值聚合、求和聚合、LSTM聚合、注意力机制聚合。
  - 最小值聚合（Min）：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。公式表示为：

$$\mathrm{m}_{\mathrm{i}}^{(\mathrm{k})} = \min_{v_j \in N(v_i)} m_{ij}^{(k)}$$

  - 最大值聚合（Max）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。公式表示为：

$$\mathrm{m}_{\mathrm{i}}^{(\mathrm{k})} = \max_{v_j \in N(v_i)} m_{ij}^{(k)}$$

4.1.2 消息传递算子

- 求和聚合（Sum）通过对所有邻居节点消息进行求和，生成一个综合聚合消息。这种聚合方式对于节点数目的变化不敏感，能够保持信息的完整性，公式表示为：

$$m_i^{(\mathrm{k})} = \sum_{v_i \in N(v_i)} m_{ij}^{(\mathrm{k})}$$

- LSTM 聚合使用长短期记忆网络（LSTM）来处理邻居节点的消息，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,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。

$$h_i^{(k)} = LSTM(h_i^{(k-1)}, m_{ij}^{(k)})$$

4.1.2 消息传递算子

- 注意力机制聚合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。

$$\mathrm{m_i^{)k)}} = \sum_{\mathrm{v_i \in N(v_i(}} \alpha_{ij}^{)k(} \bullet \mathrm{m_{ij}^{)k)}}$$

注意力权重 $\alpha_{ij}^{(k)}$ 通过以下公式计算的：

$$\alpha_{ij}^{)k(} = \frac{exp)LeakyReLU)\alpha^T]W \bullet h_i^{)k(} \,||\, W \bullet h_j^{)k(}(([}{\sum_{v_k \in N)v_i(} exp)LeakyReLU)\alpha^T]W \bullet h_i^{)k(} \,||\, W \bullet h_k^{)k(}(([}$$

α是可学习的向量，W是权重矩阵，$||$ 表示向量拼接操作

4.1.2 消息传递算子

- 示例：聚合算子用于整合节点从邻居收到的所有消息

假设节点 v_1 的邻居为 v_2 和 v_3，它们的消息分别为：

$$m_{12}\,]= 0.35,\ 0.49,[\ m_{13}\,]= 0.6,\ 0.82[$$

- 最小值聚合：$m_1 = \min([0.35,\ 0.49],\ [0.6,\ 0.82]) = [0.35,\ 0.49]$
- 最大值聚合：$m_1 = \max([0.35,\ 0.49],\ [0.6,\ 0.82]) = [0.6,\ 0.82]$
- 求和聚合：$m_1 =]0.35,\ 0.49]+[\ 0.6,\ 0.82[=]0.95,\ 1.31[$
- 平均聚合：$m_1 = \frac{]0.35,\ 0.49]+[0.6,\ 0.82[}{2} =]0.475,\ 0.655[$
- 注意力聚合，假设注意力权重为 $\alpha_{12} = 0.7$和$\alpha_{13} = 0.3$：

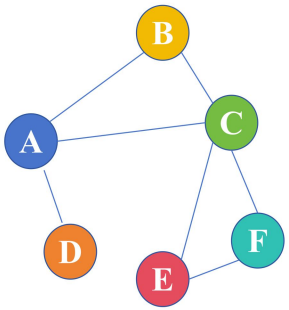
$$m_1 = 0.7 \bullet [0.35,\ 0.49] + 0.3 \bullet [0.6,\ 0.82] = [0.95,\ 1.31]$$

$$m_1\,]= 0.385,\ 0.343]+[\ 0.18,\ 0.246=[\ ]0.565,\ 0.589[$$

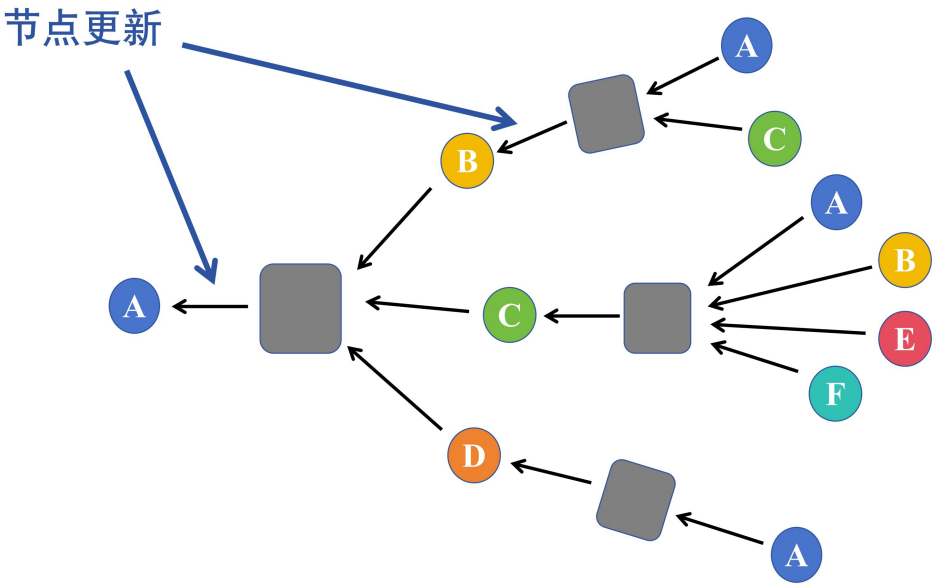
4.1.2 消息传递算子

- **更新算子** 定义：假设节点 v_i 在第 k轮的嵌入表示为 $h_i^{(k)}$，通过聚合算子得到的邻居消息为 $m_i^{(k)}$。更新算子会结合当前的节点表示和聚合的消息，生成新的节点表示$h_i^{(k+1)}$ 。这一过程可以通过以下公式表示：

$$h_i^{(k+1)} = h_i^{(k)} + m_i^{(k)}$$



输入的图结构



4.1.2 消息传递算子

- **更新算子** 一般有两种常见的类型：加法更新和门控更新。
  - 加法更新，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。公式表示为：

$$h_i^{(k+1)} = h_i^{(k)} + m_i^{(k)}$$

  - 门控更新，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 g，以此来控制节点嵌入的更新。

$$g = \sigma(W_g \cdot h_i^{(k)} + b_g)$$

$$h_i^{(k+1)} = (1 - g) \cdot h_i^{(k)} + g \cdot m_i^{(k)}$$

4.1.2 消息传递算子

- 示例：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$h_1 = [1.0, 0.5]$，聚合消息为$m_1 = [0.95, 1.31]$

  - 加法更新：

$$h_1' = h_1 + m_1 = [1.0, 0.5] + [0.95, 1.31] = [1.95, 1.81]$$

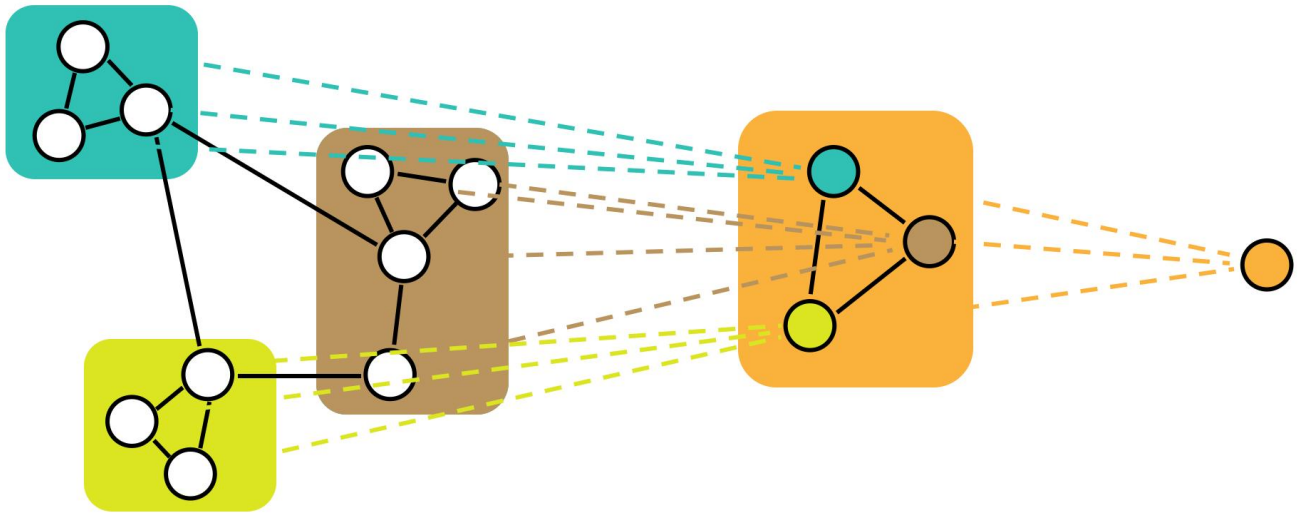
  - 门控更新假设门控向量 $g = [0.5, 0.5]$：

$$h_1' = (1 - g) \cdot h_1 + g \cdot m_1$$

$$h_1' = [0.5, 0.25] + [0.475, 0.655] = [0.975, 0.905]$$

4.1.3 图池化算子

- **图池化** 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、节点分类和图生成等下游任务的性能具有重要影响。



图池化示意图

4.1.3 图池化算子

- 图池化 一般有两种常见的策略：全局池化和层次池化
  - 全局池化（Global Pooling）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。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。公式可以表示为：

$$\mathrm{h_g} = \mathrm{pool}(\mathrm{h}_v) = \sum_{v \in V} \mathrm{h}_v$$

  - 层次池化（Hierarchical Pooling）中，节点通过一系列的选择和聚合操作，逐步形成更小的图。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。用公式表示为：

$$h_{v'} = \mathrm{select}(\mathrm{h}_v, \mathrm{A})$$

4.1.3 图池化算子

- 示例：一个简单的图结构，其中节点及其嵌入表示为：

  节点 v_1 的嵌入表示为 $h_1 = [0.5, 0.3]$，节点 v_2 的嵌入表示为 $h_2]= 0.2, 0.8[$，节点 v_3 的嵌入表示为 $h_3]= 0.6, 0.1[$

  - 在全局池化中，我们使用求和来生成一个全局特征向量：

    $$h_g = h_1 + h_2 + h_3]= 1.3, 1.2[$$

  - 在层次池化中，我们通过某种选择策略决定仅选择节点 v_1 和 v_2 进行层次池化。我们通过平均池化来生成新的表示：

    $$h_{v'} = \text{average}(h_1, h_2 = (\frac{1}{2}])0.5, 0.3] + [0.2, 0.8([$$

    $$]= 0.35, 0.55[$$

4.3 经典图神经网络

- 经典的图神经网络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。
- 接下来我们从消息、聚合、更新的角度介绍以下经典图神经网络
  - 图卷积网络（GCN）
  - 图采样聚合网络（GraphSAGE）
  - 图注意力网络（GAT）
  - 图同构网络（GIN）

4.2.1 图卷积网络（GCN）

- **半监督学习：**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，**图卷积网络（GCN）**的引入为了解决上述问题。

4.2.1 图卷积网络（GCN）

- GCN 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
- GCN的核心公式：

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

$\tilde{A}$：图的邻接矩阵加单位矩阵。

$\tilde{D}$：节点的度矩阵。

$H^{(l)}$：第 1 层的节点特征。

$W^{(l)}$：第 1 层的权重矩阵。

σ ：激活函数。

4.2.1 图卷积网络（GCN）

- 消息算子
  - 激活矩阵 $H^{(l)}$表示第l层的节点，$H^{(0)} = X$，GCN的输入是初始节点特征矩阵。
  - 对于每一层的输出$H^{(l)}$维度为$N \times D$， N为节点特征，D为特征维度。
  - 每层节点特征通过邻居信息的聚合与更新生成$H^{(l+1)}$，新的特征矩阵输入下一层进行进一步聚合。

4.2.1 图卷积网络（GCN）

- 聚合算子
  - 聚合算子负责将邻居间的信息进行聚合。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：

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

邻接矩阵（A）：表示节点间的连接关系

对角矩阵（I）：仅在对角线上有值的矩阵

在原始邻接矩阵A上添加单位矩阵I，即为每个节点添加一个自环，确保节点在更新时参考邻居与自身的初始特征，提升节点表示的表达能力。

4.2.1 图卷积网络（GCN）

- 聚合算子

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

  - 标准化邻接矩阵（$\tilde{A}$）： $\tilde{A}$ 中节点度数差异较大，直接聚合会导致高节点度对结果的影响过大
  - 度矩阵（$\tilde{D}$）： $\tilde{D}_{ii} = \sum_j A_{ij}$，表示每个节点的度数
  - 行列归一化：通过左、右乘 $\tilde{D}^{-\frac{1}{2}}$，实现行归一化和列归一化，均衡信息传播，防止由于度数差异导致的梯度消失或爆炸问题

4.2.1 图卷积网络（GCN）

- 更新算子

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$$

  - 使用权重矩阵 $W^{(l)}$ 和激活函数 σ 进行非线性转换，逐层丰富节点特征，捕捉更多局部图结构和节点信息
  - 聚合特征矩阵并执行线性变换与非线性激活，聚合邻居信息生成新特征

4.2.2 图采样聚合网络（GraphSAGE）

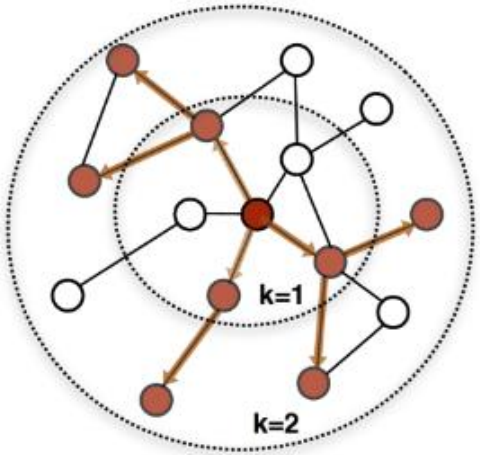
- 传统的图节点嵌入方法大多数是传导式（Transductive）的，训练和推理依赖于全图结构，只能处理训练期间看到的节点，新节点需重新训练。传导式的局限性导致这类神经网络在大规模图上效率较低。
- GraphSAGE 为解决上述问题，采用了归纳式（Inductive）的设计，通过采样和聚合邻居节点的特征信息来学习节点的嵌入，使得模型可以对未见过的节点进行嵌入。
- GraphSAGE不为每个节点单独训练嵌入向量，而是训练一组聚合器函数（Aggregator），通过从节点的局部邻域采样和聚合特征信息聚合生成嵌入

4.2.2 图采样聚合网络（GraphSAGE）

- 传导式学习（Transductive Learning）
  - 传导学习是指模型在训练时直接处理目标图上的所有节点，即训练集和测试集的节点都在同一个图中。
  - 预测时需要访问整个图，因此如果新增节点，模型必须重新训练或更新。
- 归纳式学习（Inductive Learning）
  - 归纳学习则是模型通过训练集中的节点学习一个能够泛化的表示，以便在完全看不到新节点的情况下对新节点进行预测。
  - GraphSAGE 是归纳学习的典型例子，它通过采样邻居节点来生成节点的表示，并不依赖于全图结构，因此可以处理新节点的预测任务。

4.2.2 图采样聚合网络（GraphSAGE）

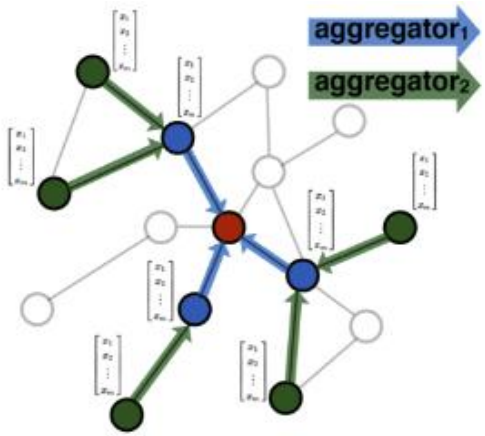
- 消息算子
  - 采样方法：对k层每个节点v，从其邻居$N(v)$中等概率采样固定数量的节点
  - 反向采样顺序：从“目标”节点开始采样，先采样目标节点的直接邻居（第k层），逐层向前找到邻居的邻居（第$k-1$层）。



1. Sample neighborhood

4.2.2 图采样聚合网络（GraphSAGE）

- 聚合算子
  - GraphSAGE 的聚合过程是一个逐层聚合邻居嵌入信息的过程，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。
  - 正向聚合顺序：从最远的k阶邻居开始，逐层向目标节点聚合嵌入信息。



2. Aggregate feature information from neighbors

4.2.2 图采样聚合网络（GraphSAGE）

- 常用聚合器函数：
  - Mean aggregator:

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

  - Max pooling aggregator:

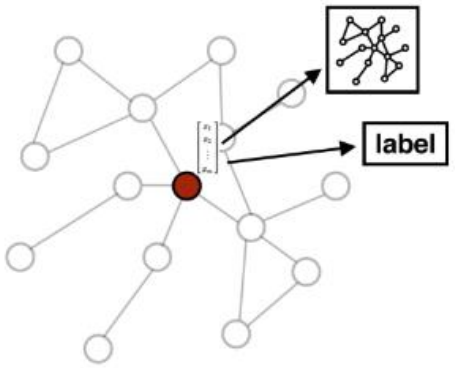
$$h_{N(v)}^{k} = \max\left(\left\{\sigma\left(W_{\mathrm{pool}} \cdot h_{u}^{k-1} + b_{\mathrm{pool}}\right), \forall u \in N(v)\right\}\right)$$

  - LSTM aggregator:

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

4.2.2 图采样聚合网络（GraphSAGE）

- 更新算子 $h_v^k \leftarrow \sigma\left(W^k \cdot \mathrm{CONCAT}(h_v^{k-1}, h_{N(v)}^k)\right)$
  - 特征拼接：将节点 v 的上一层嵌入 h_v^{k-1} 与邻居聚合表示 $h_{N(v)}^k$ 拼接，保留节点自身特征并融合邻域结构信息
  - 线性变换与激活：：拼接后的特征向量通过权重矩阵 w^k 进行线性变换，经过激活函数 σ 引入非线性，提升模型表达能力



3. Predict graph context and label using aggregated information

4.2.2 图采样聚合网络（GraphSAGE）

- GraphSAGE的核心算法流程：

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

4.2.3 图注意力网络（GAT）

- 传统图神经网络方法面临的挑战主要包括：
  - 如何处理节点邻域大小不固定的问题
  - 如何应对节点度的多样性
  - 如何通过有效采样确保信息不丢失
  - 如何捕捉全局图结构或远距离节点的关系
- 图注意力网络（GAT）通过使用图注意力机制技术解决图卷积网络（GCN）的一些缺点。它允许(隐式地)为邻接结点集中的不同结点分配不同的权重，并且不需要任何昂贵的矩阵操作，比如反演，也不依赖于需要预先知道图形结构。

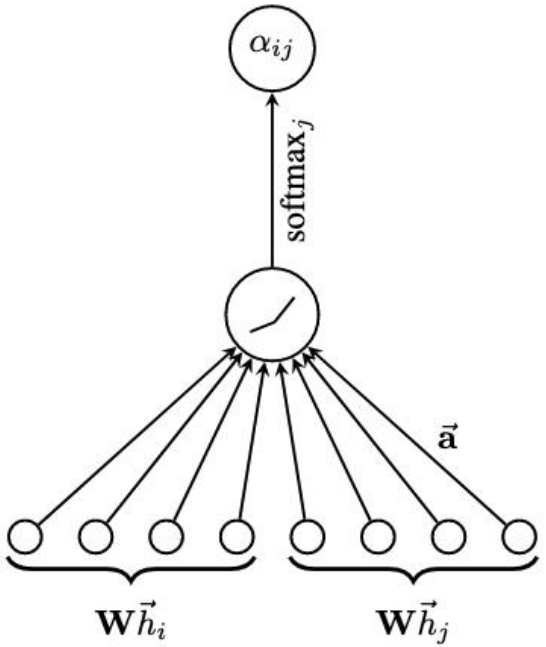
4.2.3 图注意力网络（GAT）

- 消息算子
  - 输入：节点特征集合$h = \{h_1, h_2, \ldots\ldots, h_N\}$，共有N个特征，对于每个$h_i$，特征维度为$F$
  - 输出：输出是一组新的特征维度$h' = \{h_1', h_2', \ldots\ldots, h_N'\}$，特征的数目不变，但是特征的维度被修改为$F'$

4.2.3 图注意力网络（GAT）

- 聚合算子
  - 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：$R^{F'} \times R^{F'} \rightarrow R$，节点$i$和$j$之间的注意力系数为：

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



4.2.3 图注意力网络（GAT）

- 聚合算子

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

  - 注意力机制参数$a(\cdot)$是一个单层前馈神经网络，用一个权重向量来表示：a $\in$ 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。

  - 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，并加入$leakyRelu()$非线性激活函数，我们就得到了两节点的注意力系数：

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

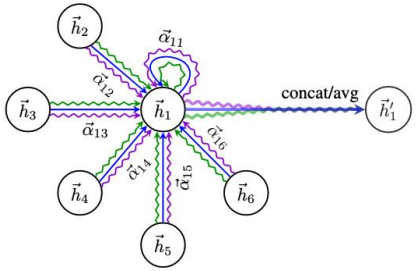
4.2.3 图注意力网络（GAT）

- 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后就得到节点的新的输出：

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

  - GAT 为了稳定 self-attention 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：

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



4.2.3 图注意力网络（GAT）

- 更新算子
  - 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：

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

4.2.4 图同构网络（GIN）

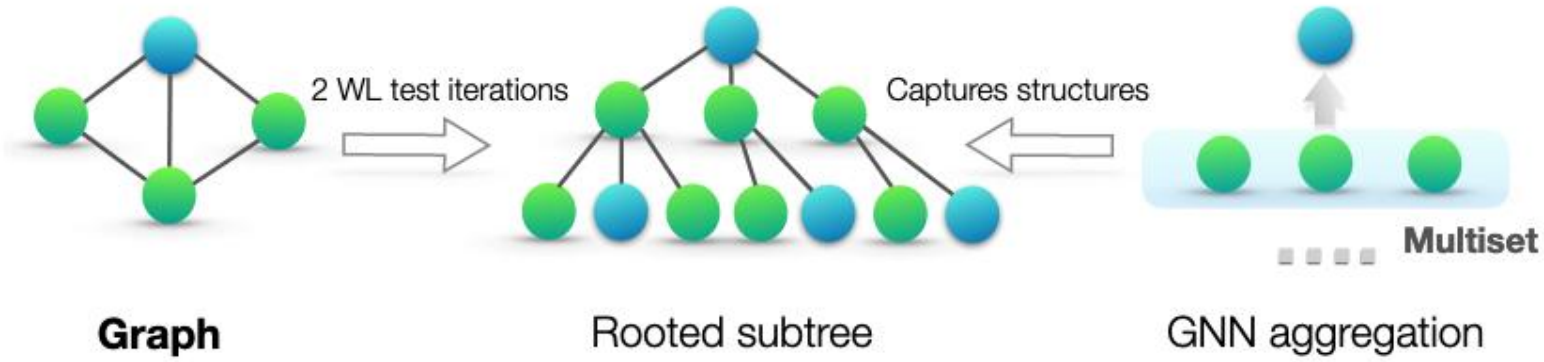
- Weisfeiler-Lehman(WL) 算法：

  在介绍 GIN 前，我们首先需要介绍 Weisfeiler-Lehman(WL) 算法：对于任意的节点$v_i \in G$

  - 获取节点 v_i 的所有的邻居节点 j 的特征 h_{v_j}
  - 更新该节点的特征 $h\ i \leftarrow \text{hash}\ \{\sum_j h_{v_j}\}$，其中 hash 是一个单射函数
  - 重复以上步骤 K 次

4.2.4 图同构网络（GIN）

- 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）。
- 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。



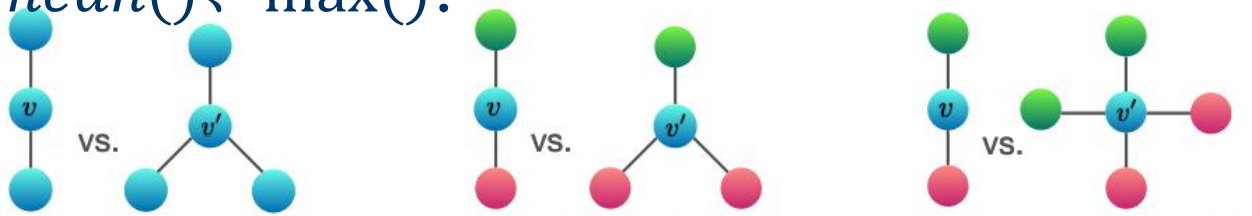
4.2.4 图同构网络（GIN）

- 消息算子：

GIN 的输入是一组节点特征，记为 $h = \{h_1, h_2, \ldots, h_N\}$, $h_i \in R^F$，其中 N 是节点的个数，F 是每个节点的特征维度。

- 聚合算子：

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不同的嵌入，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：$sum()$、$mean()$、$\max()$：

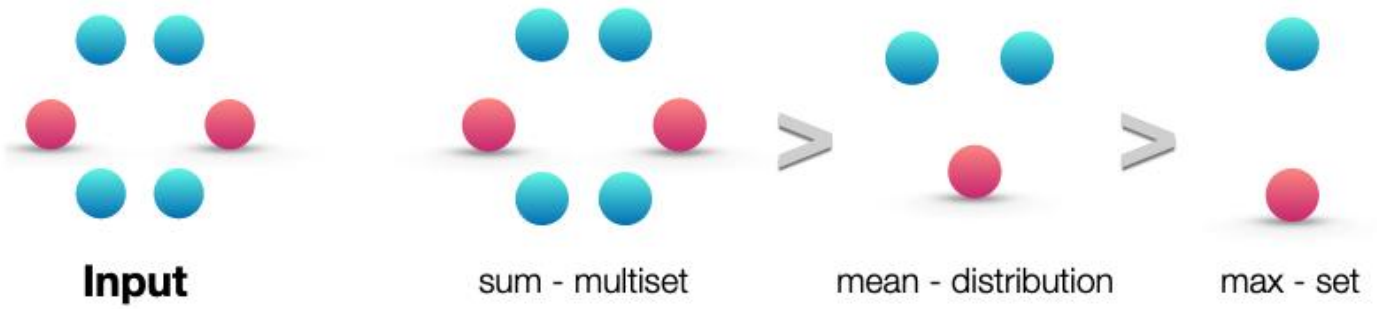


(a) Mean and Max both fail　(b) Max fails　(c) Mean and Max both fail

4.2.4 图同构网络（GIN）

- 聚合算子：
  - $sum()$：学习全部的标签及数量，学习精确的结构信息
  - $mean()$：学习标签的比例，偏向于学习分布信息
  - max()：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。



4.2.4 图同构网络（GIN）

- 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。
- 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：

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right)$$

4.4 图神经网络训练

- 在图数据上的训练任务主要分为两类
  - 节点级别的任务：其中整个数据集通常表示为一个图，图中的节点代表各个数据样本。
  - 图级别的任务：这类任务以图为中心，数据集由多个图组成，每个数据样本都是一个独立的图。

4.4 图神经网络训练

- 节点级别任务的训练流程：
  - 输入：提供图的节点特征与结构信息
  - 消息传递与聚合：通过各层级逐步聚合邻居特征
  - 监督训练：利用已标记节点的标签，计算预测输出与真实标签的损失（如交叉熵损失）
  - 参数更新：反向传播优化网络参数，循环至模型收敛
- 任务类型：

  节点分类、节点排序、链路预测等

4.3.1 节点分类

- 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：

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

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。

4.3.1 节点训练

- 为了理解交叉熵函数，我们需要从信息量讲起，信息论奠基人香农（Shannon）认为“信息是用来消除随机不确定性的东西”。也就是说衡量信息量大小就看这个信息消除不确定性的程度。
  - *“太阳从东方升起了”*：这条信息没有减少不确定性。因为太阳肯定从东面升起。这是句废话，信息量为 0。
  - *“六月飞雪”*：这条信息就比较有价值，根据历史统计信息来看，六月份鲜有下雪记录，可知该语句信息量较大。

4.3.1 节点训练

- 由上述例子我们可以发现信息量的大小和事件发生的概率应该成反比。
- 此外信息量的定义需要满足一个条件：多个事件同时发生的概率是多个事件概率相乘，总信息量却应该是多个事件信息量相加。
- 由此引出信息量的定义：

$$I(x) = -\log(P(x))$$

4.3.1 节点分类

- 如果说信息量度量的是一个具体事件发生所带来的信息，那么信息熵反映了一个系统中随机变量的“平均信息量”，即所有可能发生事件所带来的信息量的期望。由此我们得到信息熵的公式：

$$H(X) = -\sum_{i=1}^{n} p(x_i)\log(p(x_i))$$

4.3.1 节点分类

- **相对熵（relative entropy）**，又被称为 Kullback-Leibler 散度（KL 散度）或信息散度（information divergence），是两个概率分布（probability distribution）间差异的非对称性度量。在信息理论中，相对熵等价于两个概率分布的信息（Shannon entropy）的差值。

- 假设两个概率分布对应为 p(x),q(x)，如何表示这两个分布的差异，我们可以用相对熵表示。
  - p(x) 分布的信息熵：$H_{pp}(X) = -\sum_{i=1}^{n} p(x_i)\log(p(x_i))$
  - q(x) 分布的信息熵：$H_{pq}(X) = -\sum_{i=1}^{n} p(x_i)\log(q(x_i))$

4.3.1 节点分类

- 相对熵为：H_{pq}（X）- H_{pp}（X），由此我们整理相对熵公式得到：

$$D_{KL}(p||q) = \sum_{i=1}^{n} p(x_i)\log(p(x_i)) - \sum_{i=1}^{n} p(x_i)\log(q(x_i)) = \sum_{i=1}^{n} p(x_i)\log\left(\frac{p(x_i)}{q(x_i)}\right)$$

- 由于我们进行有监督训练，样本标签已经确定，相当于真实的概率的分布 P(x) 已经得知，因此 $H\,p\quad p$（X）是固定值, 这样我们就得到了交叉熵公式：

$$H(p,q) = -\sum_{i=1}^{n} p(x_i)\log(q(x_i))$$

4.3.2 链接预测

- 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：

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

4.3.3 图级别任务训练

- 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：

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

  - 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：

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

章节总结

- 在本章主要介绍了图神经网络初步。
  - 第一节，介绍了通用图神经网络框架以及消息传递算子和图池化算子。
  - 第二节介绍了经典图神经网络及其框架，理解不同的网络如何在图数据上进行编码与特征整合。
  - 第三节介绍了图神经网络在节点级别任务与图级别人物上的训练方法。
  - 通过对比不同的网络结构和任务，可以更好的掌握图神经网络的实际应用方法。