中国科学技术大学
University of Science and Technology of China

《人工智能数学原理与算法》
第4章：图神经网络

4.4 图神经网络

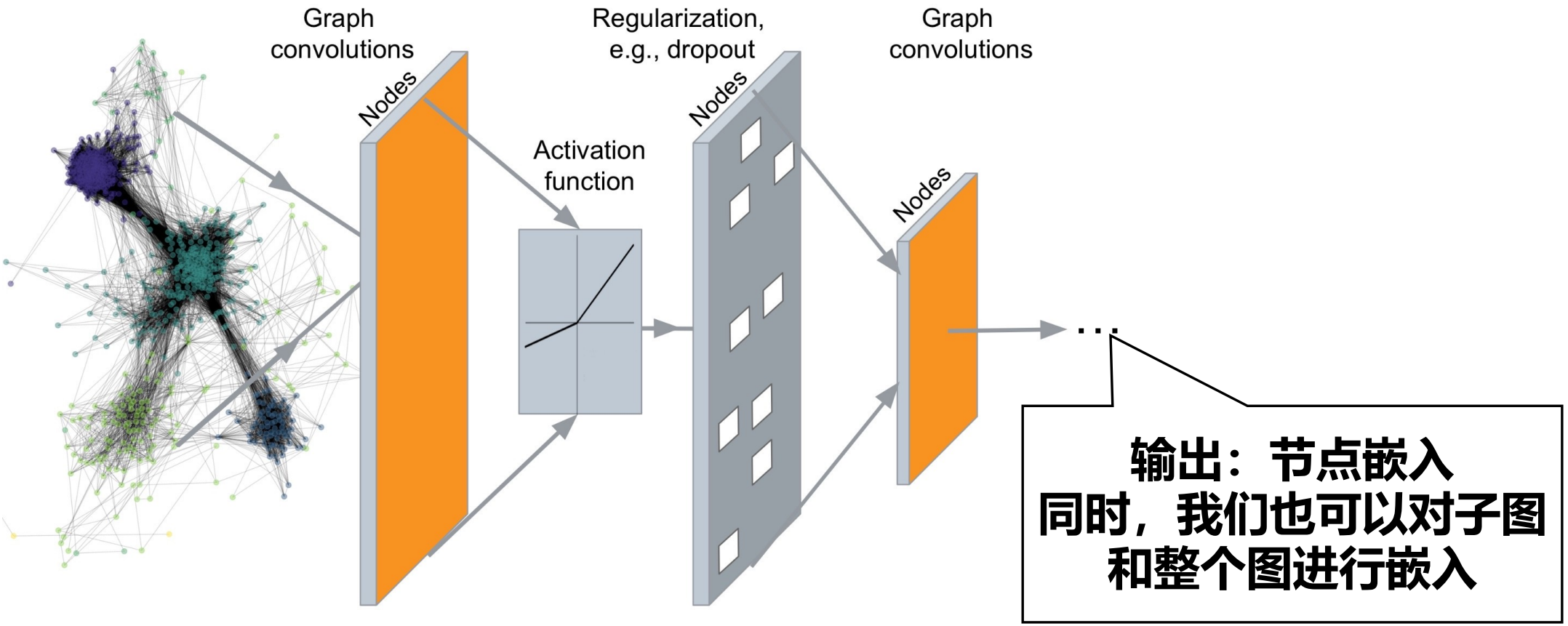
王翔

xiangwang@ustc.edu.cn

目录

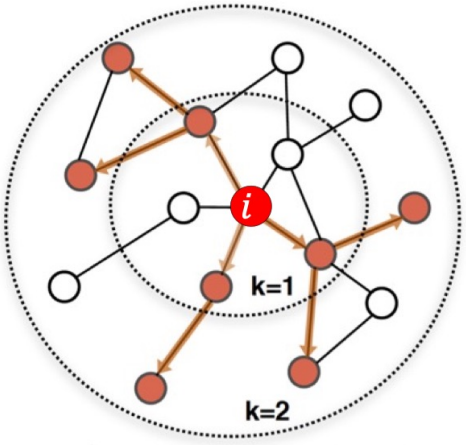
回顾：基于图神经网络的深度编码

$$\mathrm{ENC}(v) =$$ **基于图结构的多层非线性变换**

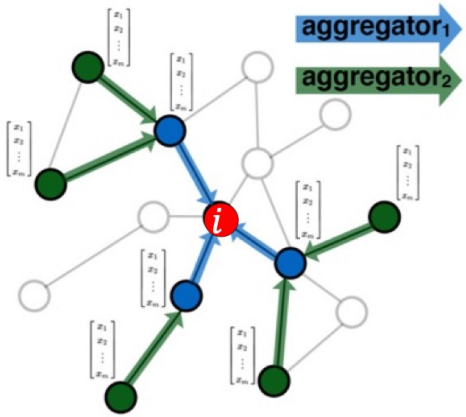


回顾：图神经网络

- **思路**：
  - 节点的邻居定义了一张计算图（computation graph）
  - 学习如何在计算图中传播与变换信息（propagate & transform information）以计算节点嵌入



确定节点的计算图

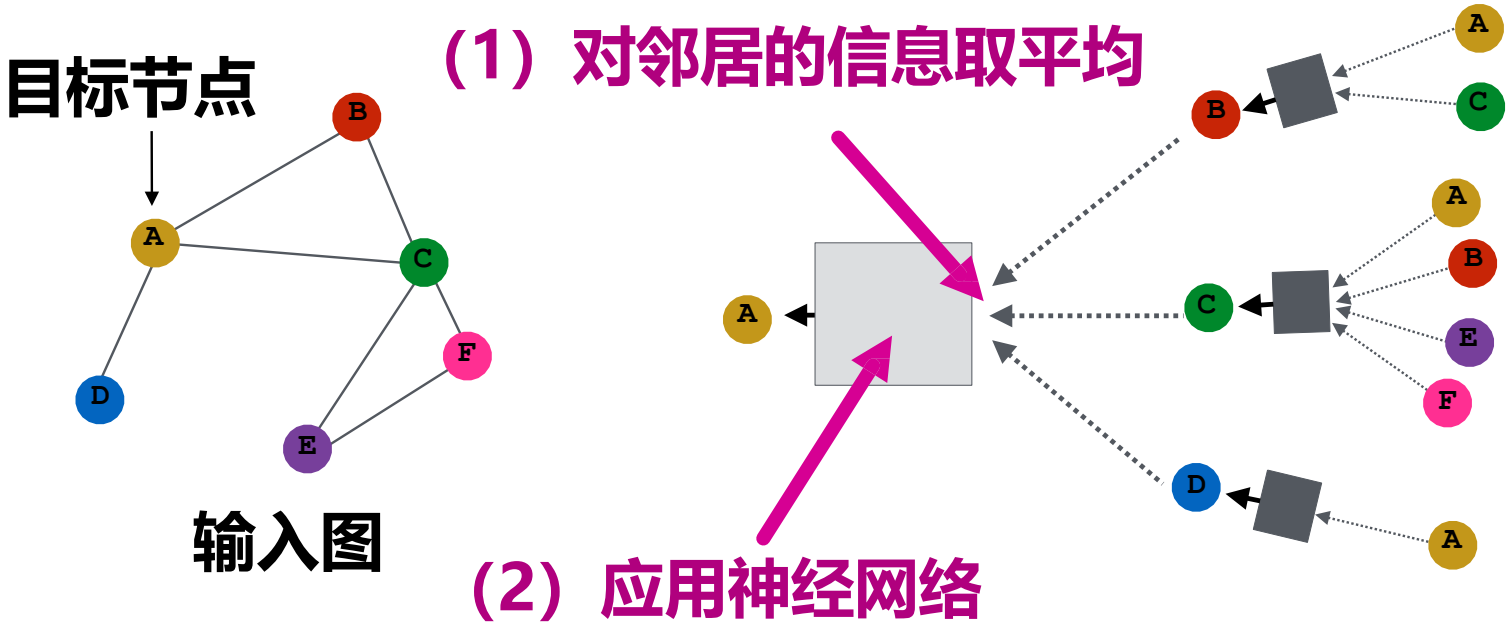


传播并变换信息

回顾：领域聚合思想

口 基本方法：

- 对邻居的信息取平均后，应用神经网络进行处理



回顾：领域聚合方法

口 基本方法：

- 对邻居的信息取平均后，应用神经网络进行处理

$h_v^o = x_v$ 初始的第0层嵌入等于节点的特征。

对邻居节点的上一层嵌入取平均

$$h_v^{(k+1)} = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)}\right), \forall k \in \{0, \ldots, K-1\}$$

非线性激活函数　　第k层嵌入　　神经网络总层数

$z_v = h_v^{(K)}$ 经过K层邻域聚合后的嵌入表示 ➔ 最终的节点嵌入

回顾：领域聚合方法

模型参数

- W_k：第k层用于邻域聚合的权重矩阵
- B_k：第k层用于变换节点自身嵌入的权重矩阵

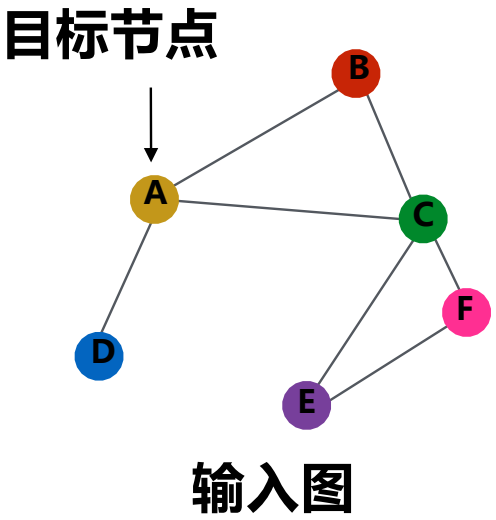
可训练的权重矩阵（即，我们需要学习的）

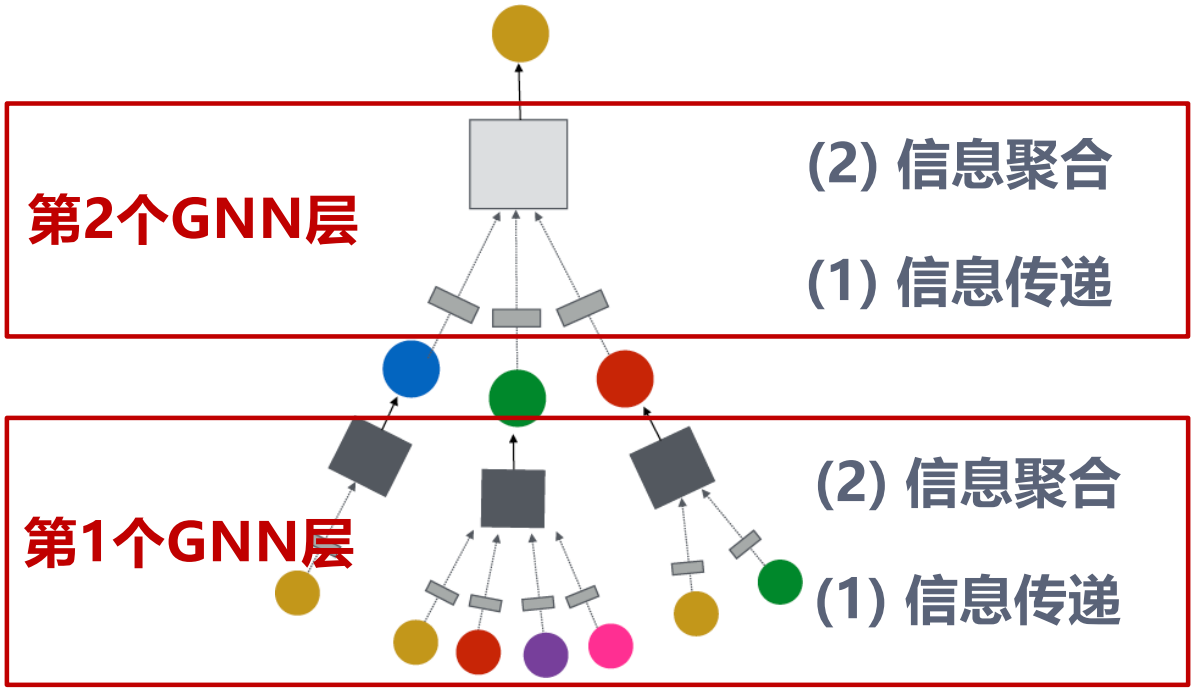
$$h_v^{(k+1)} = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{(k)}}{|N(v)|} + B_k h_v^{(k)}\right), \forall k \in \{0, \ldots, K-1\}$$

目录

通用：GNN层（A GNN Layer）

- **GNN 层 = 信息传递（Message）+ 聚合（Aggregation）**
  - 在这一视角下有不同的实现方式
  - GCN、GraphSAGE、GAT 等

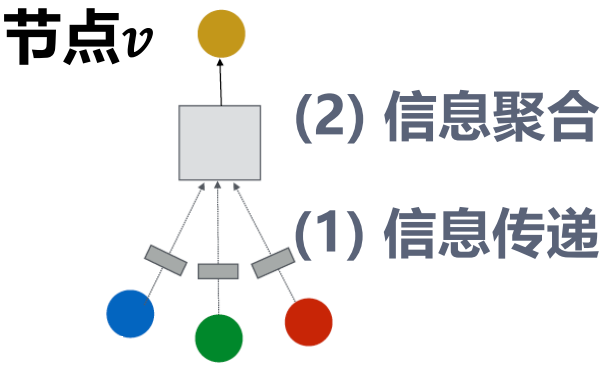


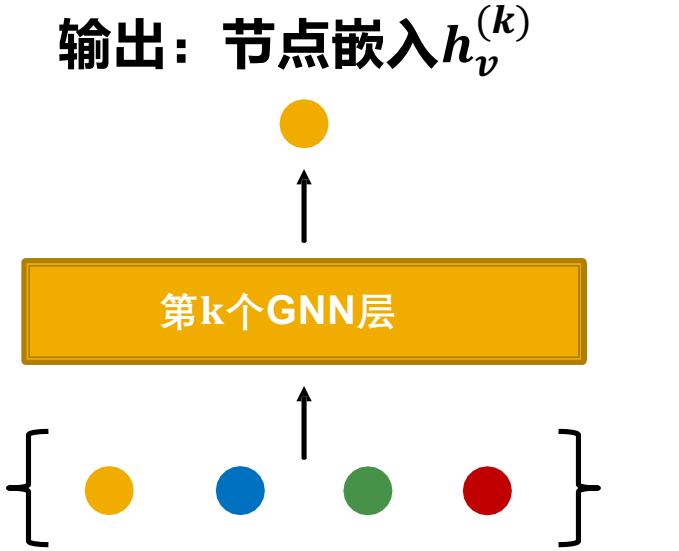


通用：GNN层（A GNN Layer）

GNN 层的基本思想

- 将一组向量压缩为一个单一向量
- 两个步骤组成的过程：
  - 信息传递（Message）
  - 信息聚合（Aggregation）





通用：信息传递与聚合

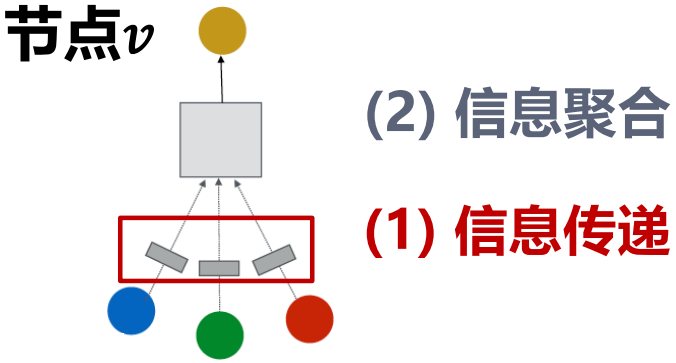
信息传递

- 直观理解：每个节点u会生成一个信息，之后会将该信息发送给其他节点：

$$\boldsymbol{m}_u^{(k)} = \mathrm{MSG}^{(k)}(\boldsymbol{h}_u^{(k-1)})$$

- 例如： $\mathrm{MSG}^{(k)}$ 由一个线性层实现， 即用权重矩阵$\boldsymbol{W}^{(k)}$乘以节点当前的嵌入

$$\boldsymbol{m}_u^{(k)} = \boldsymbol{W}^{(k)}\boldsymbol{h}_u^{(k-1)}$$



通用：信息传递与聚合

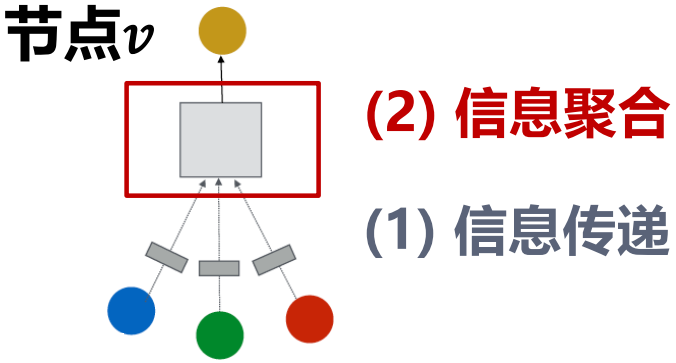
信息聚合

- 直观理解： 节点v会聚合来自其邻居节点u的消息：

$$h_v^{(k)} = \mathrm{AGG}^{(k)}(\{m_u^{(k)}, u \in N(v)\})$$

- 例如：聚合函数$\mathrm{AGG}^{(k)}$为求和$\mathrm{Sum}(\cdot)$、求平均$\mathrm{Mean}(\cdot)$或取最大值$\mathrm{Max}(\cdot)$

$$h_v^{(k)} = \mathrm{Sum}(\{m_u^{(k)}, u \in N(v)\})$$



通用：信息传递与聚合

- **潜在问题：来自节点v自身的信息可能会丢失**
  - 当前的$h_v^{(k)}$的计算并不直接依赖$h_v^{(k-1)}$

- **解决方案：在计算$h_v^{(k)}$时引入节点自身的特征/表征 $h_v^{(k-1)}$**
  - **信息传递：** 从节点v自身生成消息，通常与邻居信息产生方式不同

$$m_u^{(k)} = W^{(k)} h_u^{(k-1)}, \qquad m_v^{(k)} = B^{(k)} h_v^{(k-1)}$$

  - 信息聚合：在从邻居聚合后，还可以聚合节点v自身的消息，例如通过拼接（CONCAT）或求和

首先邻居聚合　　再聚合自身

$$h_v^{(k)} = \text{CONCAT}\left(\text{AGG}\left(\left\{m_u^{(k)}, u \in N(v)\right\}\right), m_v^{(k)}\right)$$

通用：信息传递与聚合

整体流程总结：

- 信息传递：每个节点（自身+邻居）计算消息

$$\boldsymbol{m_u^{(k)}} = \mathrm{MSG}^{(k)}\left(\boldsymbol{h_u^{(k-1)}}\right), \boldsymbol{u} \in \{\boldsymbol{N(v)} \cup \boldsymbol{v}\}$$

- 信息聚合：从所有节点（自身+邻居）聚合消息

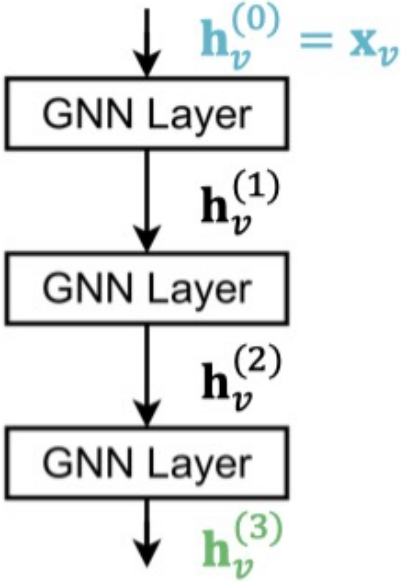
$$\boldsymbol{h_v^{(k)}} = \mathrm{AGG}^{(k)}(\left\{\boldsymbol{m_u^{(k)}}, \boldsymbol{u} \in \boldsymbol{N(v)}\right\}, \boldsymbol{m_v^{(k)}})$$

- **非线性激活： 增强模型表达能力**
  - 通常表示为$\sigma(\cdot)$，例如ReLU(·), Sigmoid(·), …
  - 可以加在消息计算或聚合结果之后

堆叠多个GNN层

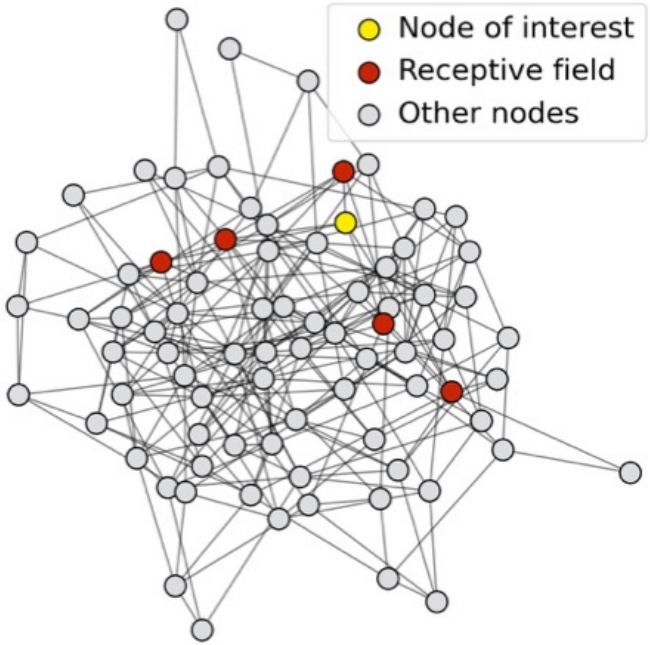
如何构建图神经网络：

- **标准方法**：顺序堆叠多个GNN 层
- **输入**： 初始的原始节点特征x_v
- **输出**： 经过K个GNN层后得到的节点嵌入表示$h_v^{(K)}$

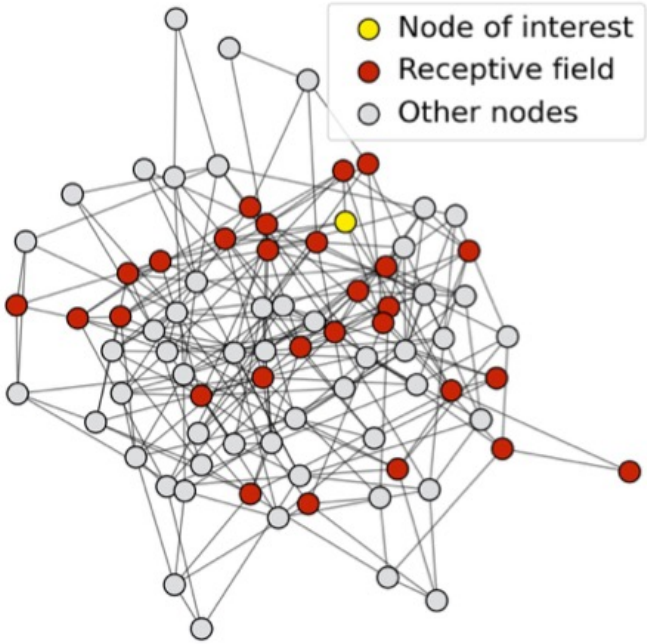


堆叠多个GNN层

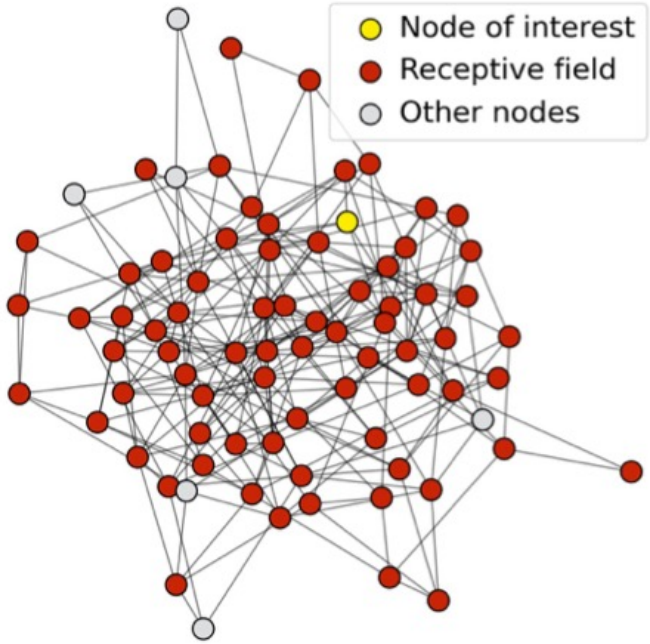
- **感受野： 指决定某个目标节点表示的所有节点的集合**
  - 在一个K层的GNN中，每个节点的感受野为其K跳邻居的集合



一层GNN的感受野



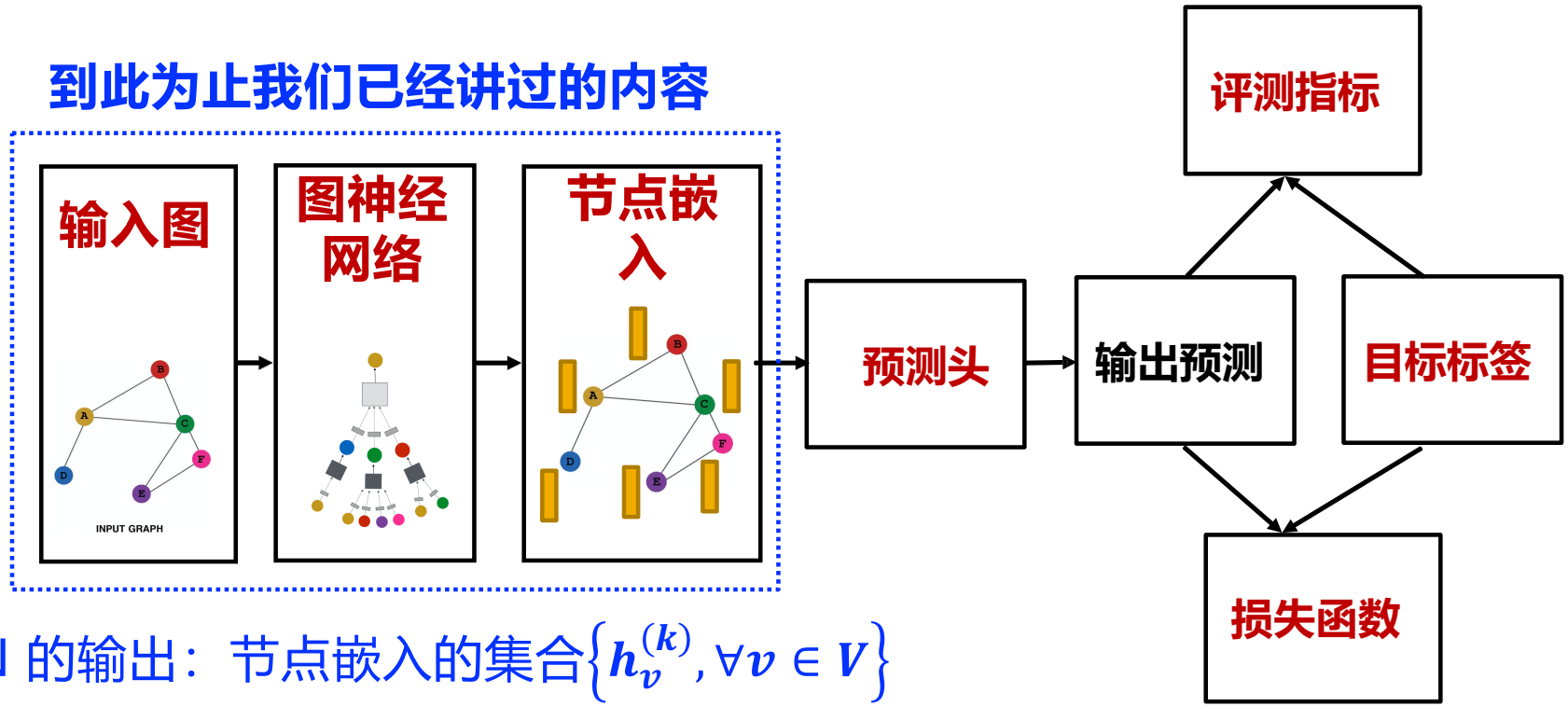
二层GNN的感受野



三层GNN的感受野

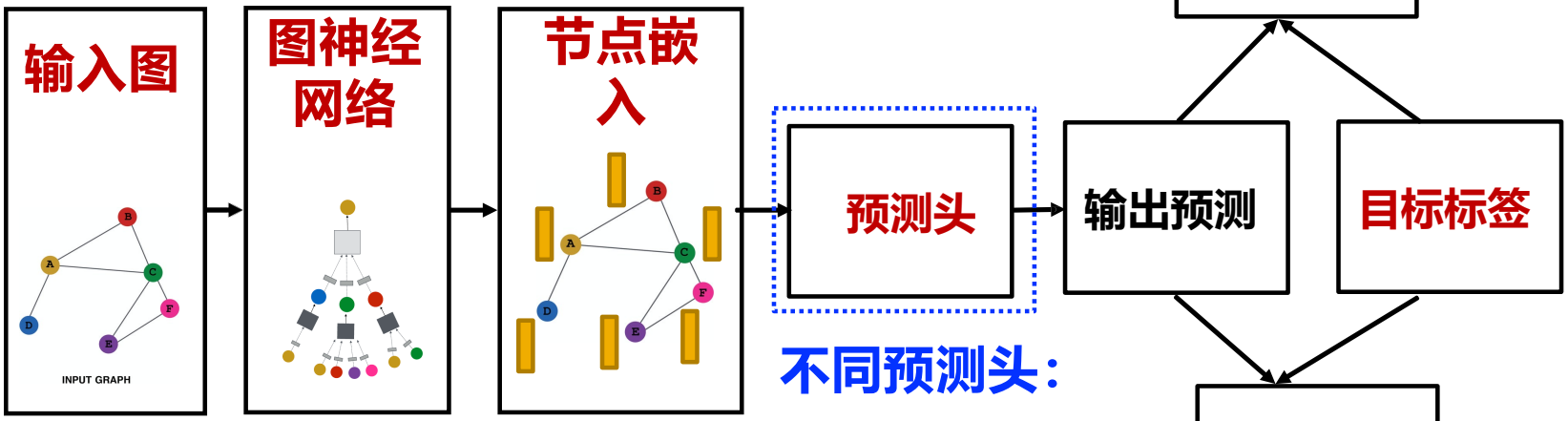
目录

图神经网络：训练流程



GNN 的输出：节点嵌入的集合$\left\{\boldsymbol{h}_v^{(k)}, \forall v \in \boldsymbol{V}\right\}$

图神经网络：训练流程



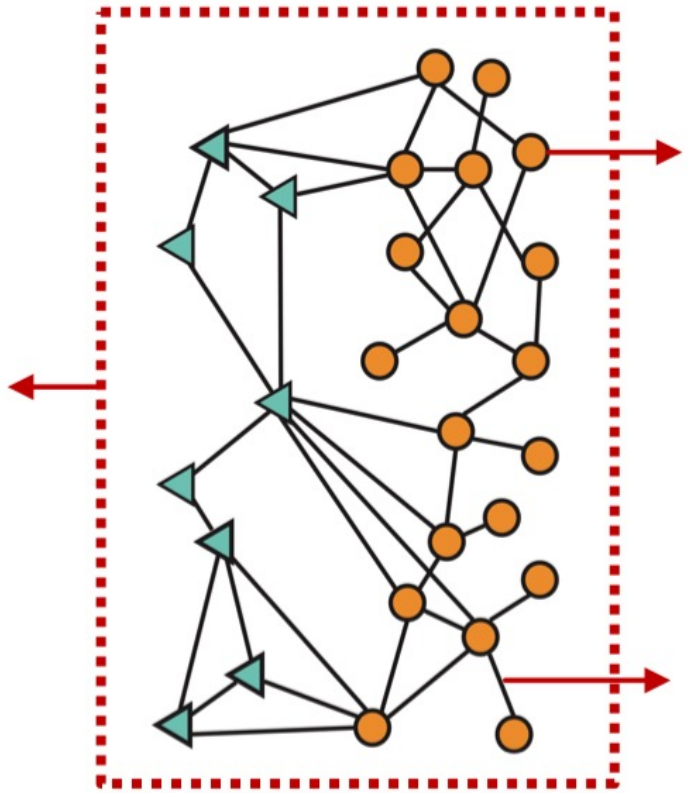
不同预测头：

- 节点级任务
- 边级任务
- 图级任务

图神经网络：预测流程

口 核心思想：不同的任务层级需要不同的预测头

- 在一个K层的GNN中，每个节点的感受野为其K跳邻居的集合



口 节点级预测

口 图级预测

口 边级预测

预测头（Prediction Head）：节点级

节点级预测：我们可以直接使用节点表示进行预测！

- 经过 GNN 计算后，我们得到了维度为d的节点嵌入：

$$\left\{\boldsymbol{h}_v^{(k)}, \forall \boldsymbol{v} \in \boldsymbol{V}\right\}$$

- 假设我们要进行C分类或C维回归预测：
  - 分类任务： 在C个类别中进行分类
  - 回归任务： 对C个目标值进行回归
- 预测过程如下：

$$\hat{\boldsymbol{y}}_v = \boldsymbol{Head}_{\boldsymbol{node}}\left(\boldsymbol{h}_v^{(K)}\right) = \boldsymbol{Linear}\left(\boldsymbol{h}_v^{(K)}\right) = \boldsymbol{W}^{(H)}\boldsymbol{h}_v^{(K)}$$

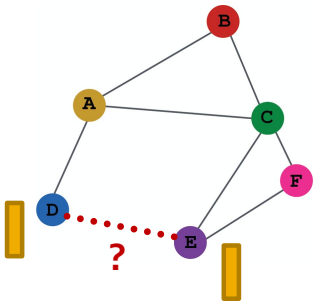
  - $\boldsymbol{W}^{(H)} \in \boldsymbol{R}^{\boldsymbol{C}\times\boldsymbol{d}}$将节点嵌入$\boldsymbol{h}_v^{(K)} \in \boldsymbol{R}^{\boldsymbol{d}}$映射为预测结果$\hat{\boldsymbol{y}}_v \in \boldsymbol{R}^{\boldsymbol{C}}$以计算损失函数

预测头（Prediction Head）：边级

边级预测：使用一对节点表示进行预测

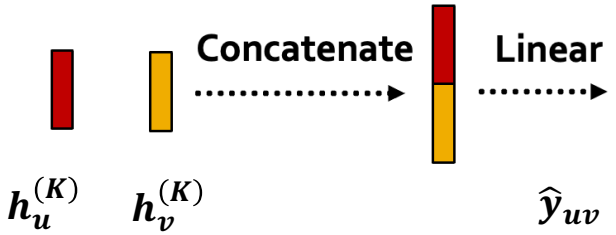
- 假设我们要进行C类预测任务
- 预测过程如下：

$$\hat{y}_{uv} = Head_{edge}\left(h_u^{(K)}, h_v^{(K)}\right)$$



- 当C为任意大小时，实现方案：

$$\hat{y}_{uv} = Linear(Concat\left(h_u^{(K)}, h_v^{(K)}\right))$$



- 当$C = 1$，实现方案：

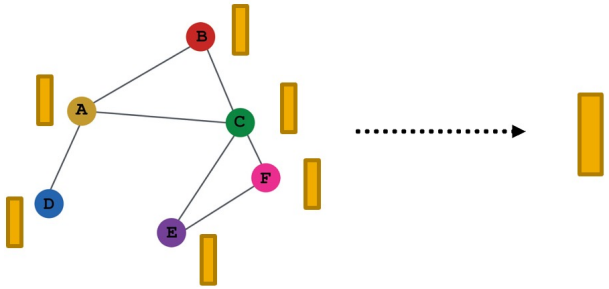
$$\hat{y}_{uv} = (h_u^{(K)})^T h_v^{(K)}$$

预测头（Prediction Head）：图级

图级预测：使用整张图中所有节点的表示进行预测

- 假设我们要进行C类预测任务
- 预测过程如下：

$$\hat{\boldsymbol{y}}_{\boldsymbol{G}} = \boldsymbol{Head}_{\boldsymbol{graph}}\left(\left\{\boldsymbol{h}_{\boldsymbol{v}}^{(k)}, \forall \boldsymbol{v} \in \boldsymbol{V}\right\}\right)$$



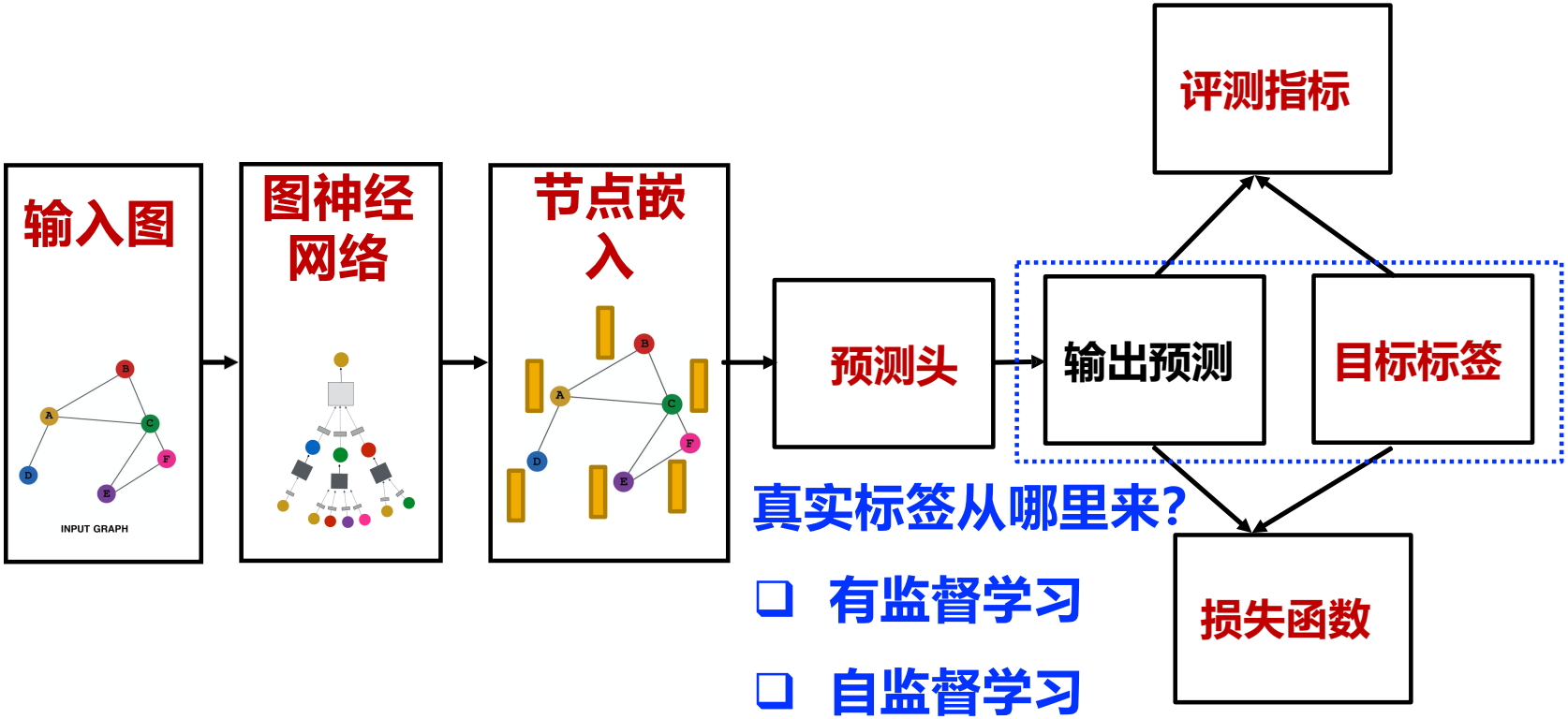
- $\boldsymbol{Head}_{\boldsymbol{graph}}(\cdot)$与 GNN 层中的 $\boldsymbol{AGG}(\cdot)$操作类似！

$$\hat{\boldsymbol{y}}_{\boldsymbol{G}} = \boldsymbol{AGG}\left(\left\{\boldsymbol{h}_{\boldsymbol{v}}^{(k)}, \forall \boldsymbol{v} \in \boldsymbol{V}\right\}\right)$$

- 聚合函数$\mathrm{AGG}^{(k)}$为求和Sum(·)、求平均Mean(·)、最大值Max(·)或注意力机制Att(·)：

$$\hat{\boldsymbol{y}}_{\boldsymbol{G}} = \mathrm{Sum}\left(\left\{\boldsymbol{h}_{\boldsymbol{v}}^{(k)}, \forall \boldsymbol{v} \in \boldsymbol{V}\right\}\right)$$

图神经网络：预测流程


输入图
INPUT GRAPH
图神经网络
节点嵌入
预测头
输出预测
评测指标
目标标签
损失函数
真实标签从哪里来？
有监督学习
自监督学习

有监督 vs 自监督

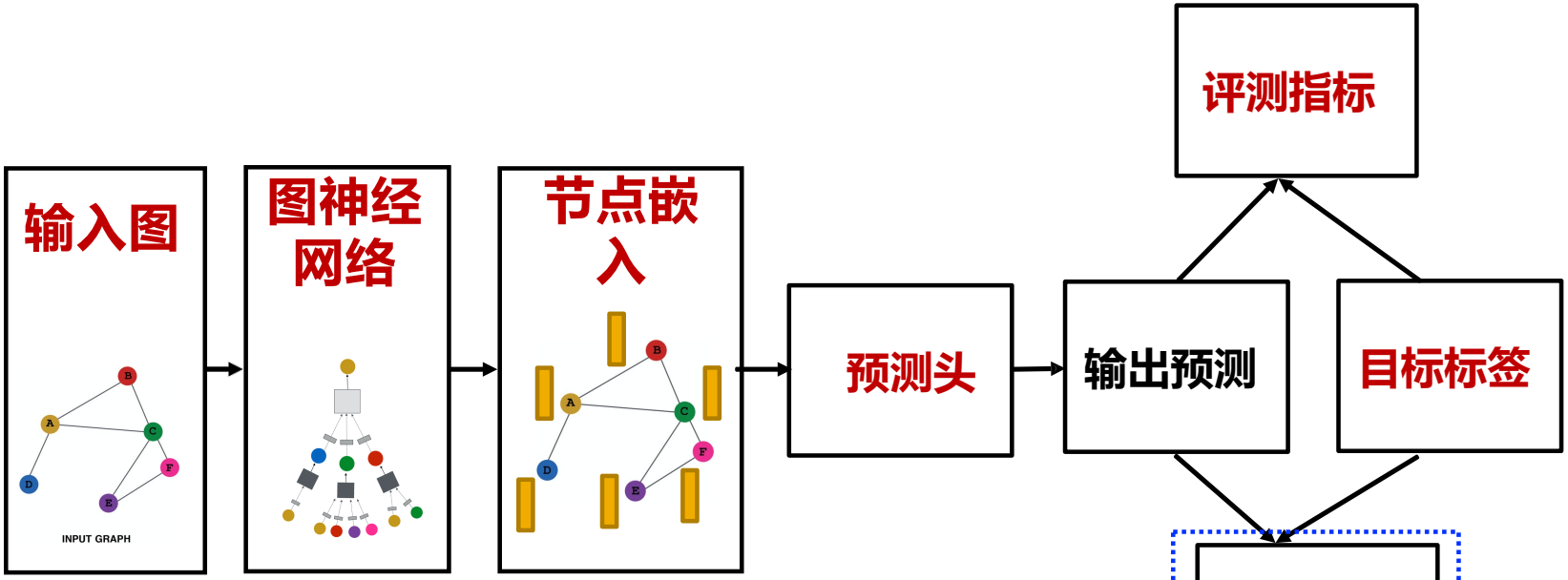
图上的有监督学习

- **标签来自外部数据源**
  - 节点标签y_v：在引文网络中，表示某个节点所属的学科领域
  - 边标签y_{uv}：在交易网络中，表示一条边是否为欺诈交易
  - 图标签y_G：在分子图中，表示每个图的药物相似性

图上的自监督学习

- **信号来自图自身**
  - 节点标签y_v：节点统计信息，例如聚类系数、PageRank 等
  - 边标签y_{uv}： 隐藏两个节点之间的边，预测它们之间是否应该存在连接
  - 图标签y_G： 图的统计信息，例如预测两张图是否同构

图神经网络：预测流程



我们如何计算最终的损失？

- **分类损失**
- **回归损失**

有监督 vs 自监督

- **任务设定：我们有N个数据点**
  - 每个数据点可以是一个节点 / 边 / 图
  - 节点级：预测值$\hat{y}_v^{(i)}$，目标标签为$y_v^{(i)}$
  - 边级：预测值$\hat{y}_{uv}^{(i)}$，目标标签为$y_{uv}^{(i)}$
  - 图级：预测值$\hat{y}_G^{(i)}$，目标标签为$y_G^{(i)}$

- **我们统一用预测值$\hat{y}^{(i)}$、标签 $y^{(i)}$表示任意层级的预测任务**

分类损失 vs 回归损失

- 分类：标签是离散类别
  - 常用：交叉熵损失（Cross Entropy, CE）

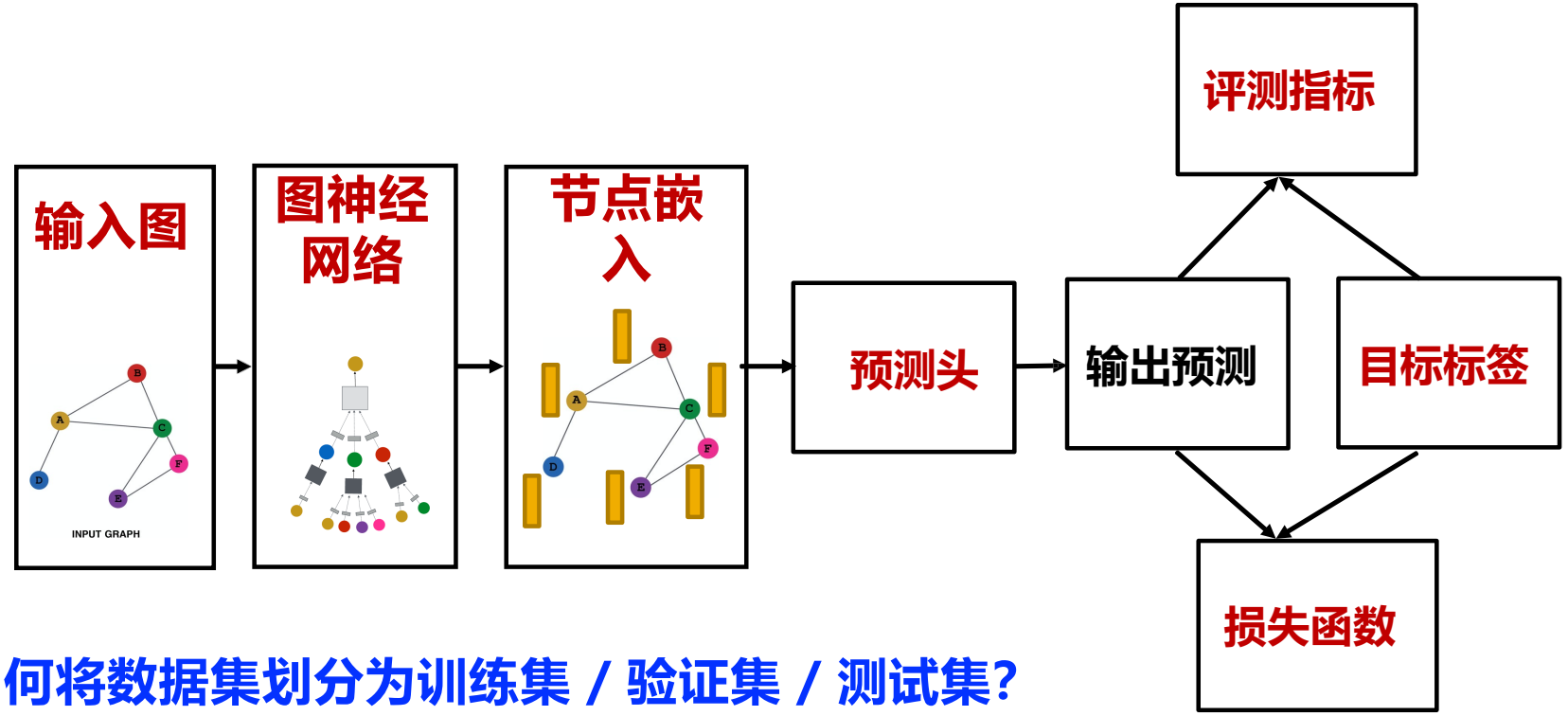
$$Loss = \sum_{i=1}^{N} CE(y^{(i)}, \hat{y}^{(i)})$$

- 回归：标签是连续值
  - 常用：最小二乘误差（Mean Squared Error, MSE）

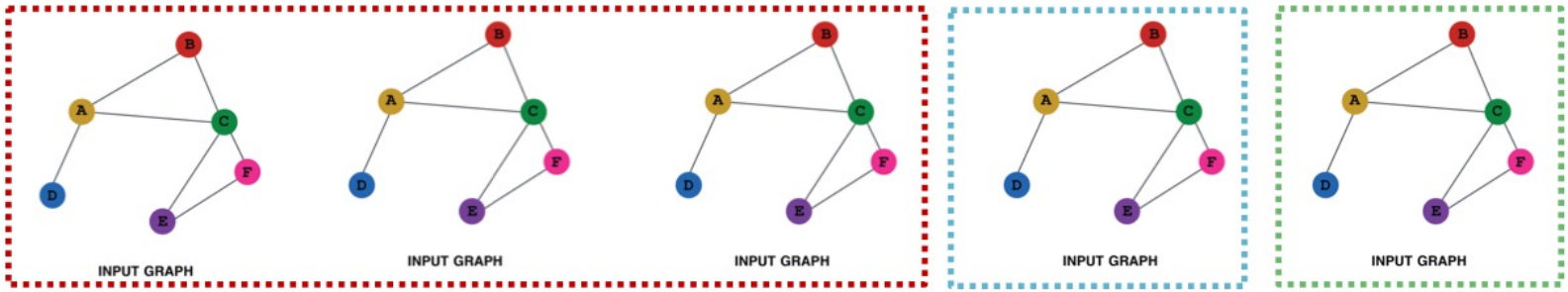
$$Loss = \sum_{i=1}^{N} MSE(y^{(i)}, \hat{y}^{(i)})$$

目录

分类损失 vs 回归损失



我们如何将数据集划分为训练集 / 验证集 / 测试集?



数据划分：固定 vs 随机划分

- **数据集：**
  - **训练集（Training set）**：用于优化 GNN 参数$\Theta_{GNN} = \{(\boldsymbol{W}_k, \boldsymbol{B}_k)\}_{k=1}^{K}$和预测头参数$\Theta_{Head}$
  - **验证集（Validation set）**：用于模型调试和超参数选择（如，学习率等）
  - **测试集（Test set）**：保留至最后用于报告最终性能

- **固定划分**：数据集只划分一次
  - 潜在问题：有时无法保证测试集在训练过程中完全不被使用
- **随机划分**：将数据集随机划分为训练 / 验证 / 测试集
  - 报告结果时会在多个随机种子上取平均性能

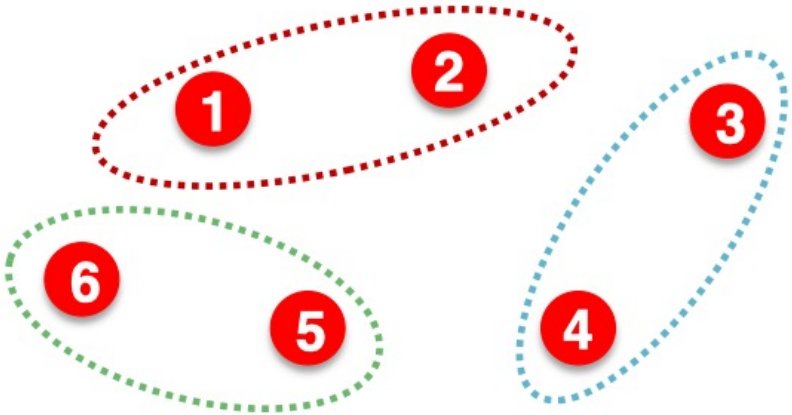
图数据划分的特殊性

- **假设我们要划分一个图像数据集**
  - 图像分类任务中：每个数据点都是一张图像
  - 在这种情况下，数据点之间是独立的
    - 例如：图像 5 的内容不会影响我们对图像 1 的预测

训练集

验证集

测试集



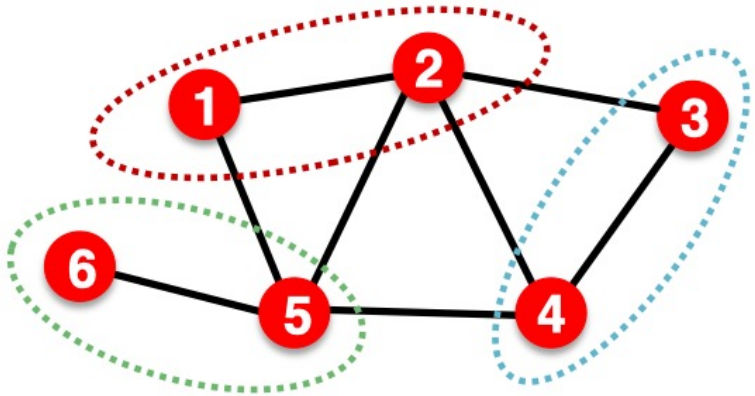
图数据划分的特殊性

- **划分图结构数据集则不同！**
  - 节点分类任务： 每个数据点是一个节点
  - 在这种情况下，数据点之间不是独立的
    - 例如：节点 5 会影响我们对节点 1 的预测
    - 因为它会参与消息传递，从而影响节点 1 的嵌入

训练集

验证集

测试集



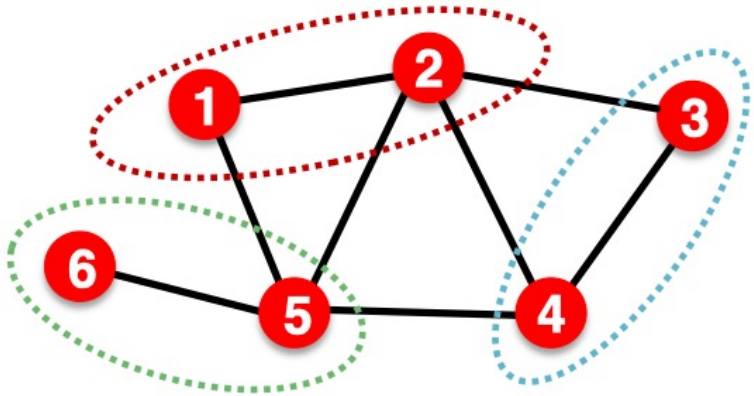
传导式设定

- **输入图在训练集、验证集和测试集中都是可见的**
- **我们只对节点标签进行划分**
  - 训练阶段：使用**整个图**计算节点表示，但只利用节点1&2的标签进行训练
  - 验证阶段：同样使用**整个图**计算表示，然后在节点3&4上评估模型性能

训练集

验证集

测试集



归纳式设定

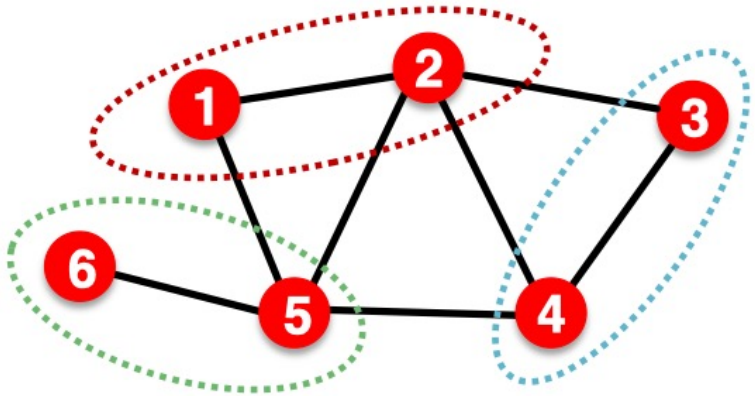
打断不同划分之间的边，从而获得多个相互独立的子图

- 现在我们有 3 个相互独立的图，节点 5 将不再影响对节点 1 的预测
- 训练阶段：使用**包含节点 1 和 2 的子图**计算表示，并用它们的标签进行训练
- 验证阶段：使用**包含节点 3 和 4 的子图**计算表示，并在它们的标签上评估模型性能

训练集

验证集

测试集



传导式 vs 归纳式设定

传导式设定： 训练集 / 验证集 / 测试集位于同一个图上

- 数据集由一个图组成
- 在所有数据划分中，整个图都是可见的，只对标签进行划分
- 仅适用于节点和边预测任务

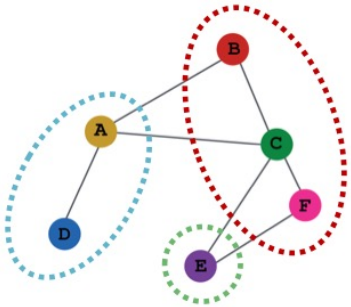
归纳式设定： 训练集 / 验证集 / 测试集位于不同的图上

- 数据集由多个图组成
- 每个划分只能访问其自身包含的图
- 一个成功的模型应当能够泛化到未见过的图
- 适用于节点、边、图级别的任务

举例：节点分类

传导式节点分类

- 所有划分都可以观察到整个图，但只能看到各自划分中节点的标签



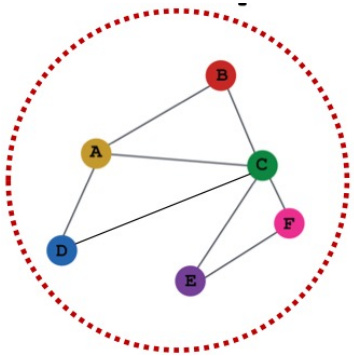
训练集

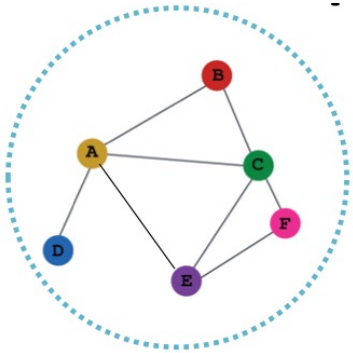
验证集

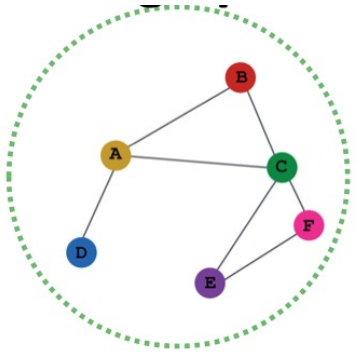
测试集

归纳式节点分类

- 假设我们有一个包含 3 个图的数据集
- 每个划分包含一个相互独立的图







训练集

验证集

测试集

举例：图分类

- **只有归纳式设定在图分类任务中是明确定义的**
  - 因为我们必须在未见过的图上进行测试
  - 假设我们有一个包含 5 个图的数据集，每个划分将包含相互独立的图