Graph Neural Network

在上一节中，我们学习了如何使用“浅层编码器”表示图。这些技术为我们提供了在向量空间中表示图的强大方式，但也有其局限性。在本节中，我们将探索使用图神经网络克服限制的三种不同方法。

Limitation of "Shallow Encoders"(浅层编码器”的局限性)

- 浅编码器无法缩放，因为每个节点都有唯一的嵌入。
- 浅层编码器具有固有的传导性。它只能为单个固定图生成嵌入。
- 不考虑节点的特征。
- 不能将浅层编码器推广到具有不同损失函数的训练中。

幸运的是，图神经网络可以解决上述限制。

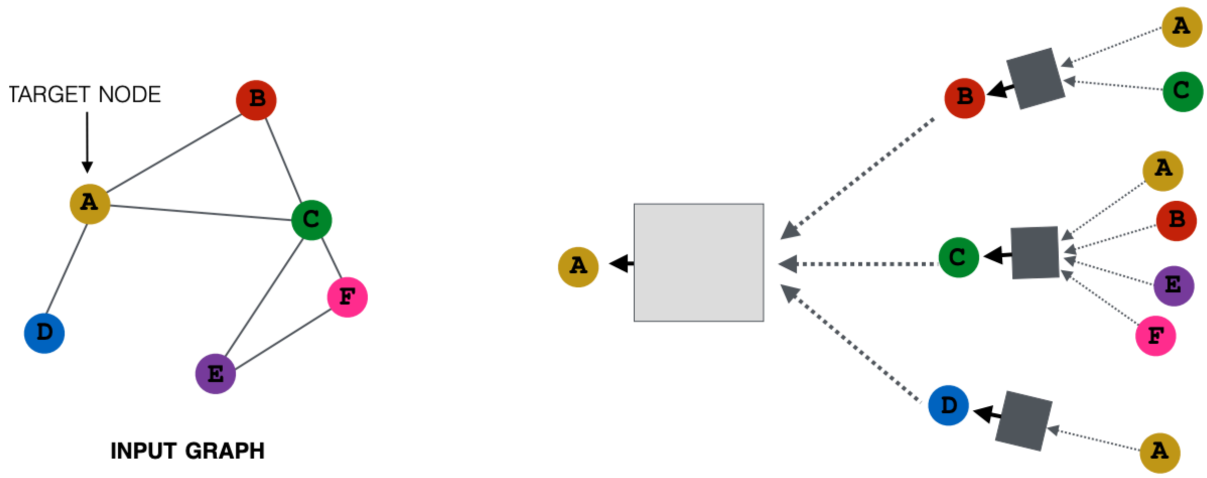
Graph convolutional Networks(GCN, 图神经网络)

传统上，神经网络是为固定大小的图设计的。例如，我们可以将图像视为网格图，或将一段文本视为线图。但是，现实世界中的大多数图具有任意大小和复杂的拓扑结构。因此，我们需要不同地定义GCN的计算图。

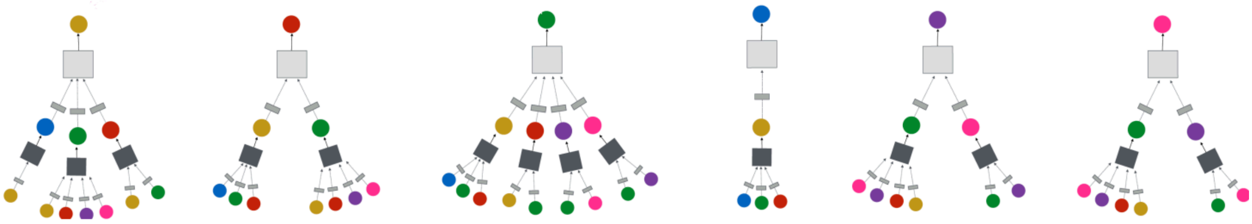
假设给定图 $G = (V, A, X)$：

- V 是顶点集合
- A 是邻接矩阵
- $X \in \mathbb{R}^{m \times |V|}$ 是节点的特征矩阵

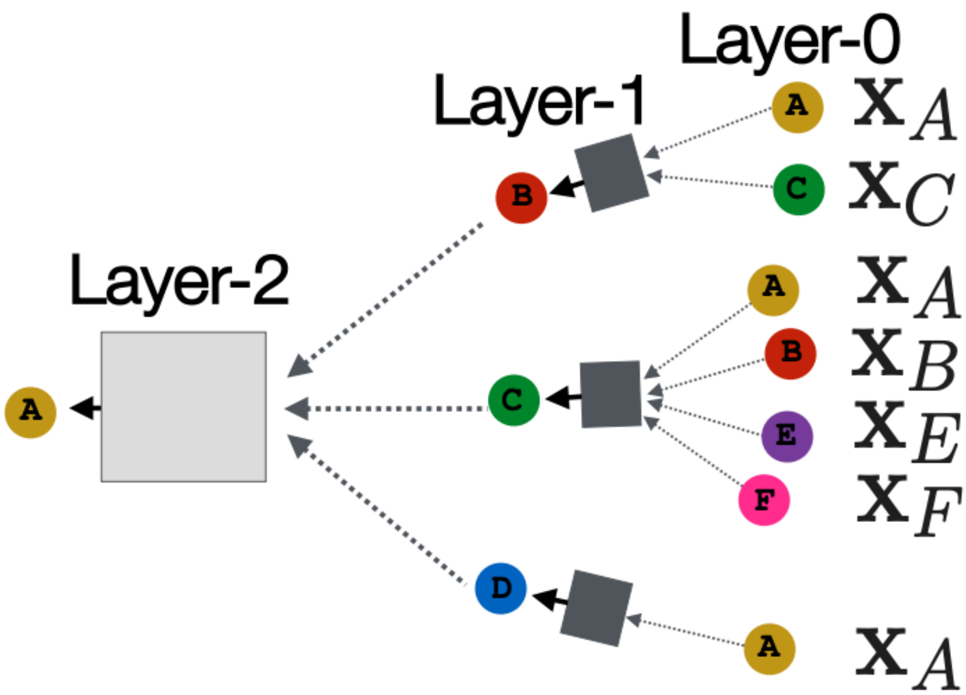
计算图和广义卷积



假设示例图(上图左图)为图 G 。我们的目标是定义在图 G 上的GCN计算图。计算图应同时保持图 G 的结构和合并节点的相邻要素。例如，节点的嵌入向量 A 应该包括它的邻居 $\{B, C, D\}$ 并且和 $\{B, C, D\}$ 的顺序无关。一种方法是简单地取 $\{B, C, D\}$ 的平均值。通常，聚合函数(上图右图中的方框)必须是阶不变的(最大值，平均值等)。上图具有两层计算图 G 如下所示:



这里，每个节点都基于其邻居定义一个计算图。特别的，节点 A 的计算图结构如下所示:（第0层是输入层，输入为节点特征 X_i):



Deep Encoders(深度编码器)

有了以上想法，这是节点 v 使用平均聚合函数的每一层的数学表达式：

- 在第0层: $h_v^0 = x_v$, 表示节点特征
- 在第k层:
  $h_v^k = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}\right), \forall k \in \{1, \ldots, K\}$

h_v^{k-1} 是节点 v 从上一层开始的嵌入。$|N(v)|$ 是节点 v 的邻居数。$\sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$ 的目的是聚合节点 v 上一层的所有邻居特征。σ 是引入非线性的激活函数(例如ReLU)。W_k 和 B_k 是可训练的参数。

- 输出层: $z_v = h_v^K$ 是K层嵌入后的最后的嵌入层。

等效地，以上计算可以以写成整个图矩阵乘法的形式:

$$H^{l+1} = \sigma\left(H^l W_0^l + \tilde{A} H^l W_1^l\right) \text{ such that } \tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Training the Model

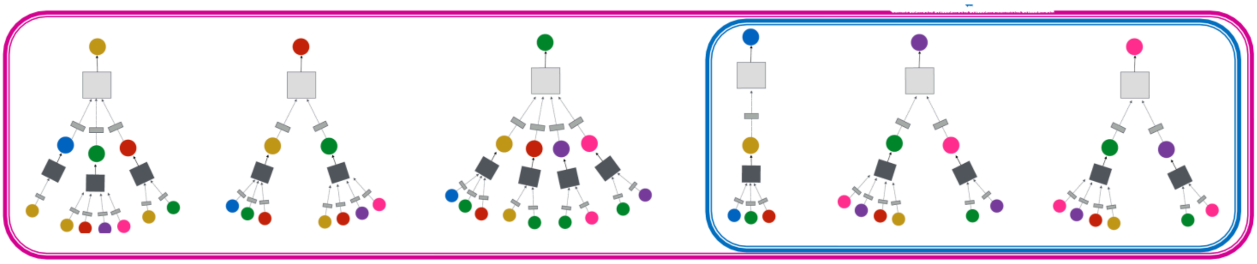
我们可以为这些嵌入提供给任何损失函数，并进行随机梯度下降训练参数。例如，对于二进制分类任务，我们可以将损失函数定义为:

$$L = \sum_{v \in V} y_v \log\left(\sigma\left(z_v^T \theta\right)\right) + (1 - y_v) \log\left(1 - \sigma\left(z_v^T \theta\right)\right)$$

$y_v \in \{0, 1\}$ 是节点类标签。z_v 是编码器的输出。θ 是分类权重。σ 可以是 sigmoid 函数。$\sigma(z_v^T \theta)$ 表示节点 v 的预测概率。因此，如果标签为正 $(y_v = 1)$，则损失函数方程将计算前半部分，否则，损失函数方程将计算后半部分。
我们还可以通过以下方式以无监督的方式训练模型：随机游走，图形分解，节点接近等。

Inductive Capability(归纳能力)

GCN可以应用在图中看不见的节点。例如，如果使用节点 A, B, C 训练模型，由于参数在所有节点之间共享，新添加的节点 D, E, F 因此也可以进行评估。



GraphSAGE

到目前为止，我们已经探索了一种简单的邻域聚合方法，但是我们还可以将聚合方法概括为以下形式：

$$h_v^K = \sigma\left(\left[W_k AGG\left(\left\{h_u^{k-1}, \forall u \in N(v)\right\}\right), B_k h_v^{k-1}\right]\right)$$

对于节点 v，我们可以应用不同的汇总方法(AGG)与将其邻居和节点 v 本身的特征相连接。

下面是一些常用的聚合函数：

- 平均值：取其邻居的加权平均值。
  $AGG = \sum_{u \in N_v} \frac{h_u^{k-1}}{|N(v)|}$
- 池化：转换邻居向量并应用对称向量函数(γ 可以是按元素的均值或最大值)。
  $AGG = \gamma(\{Qh_u^{k-1}, \forall u \in N(v)\})$
- LSTM：使用LSTM应用于重组后的邻居。
  $AGG = LSTM(\{h_u^{k-1}, \forall u \in \pi(N(v))\})$

Graph Attention Networks(图注意力网络)

如果某些相邻节点携带的信息比其他节点更重要怎么办？在这种情况下，我们希望通过使用注意力技巧将不同的权重分配给不同的相邻节点。

假设 α_{vu} 是节点 u 向节点 v 传递的信息的加权因子(重要性)。 根据上面的平均聚合函数，我们定义了 $\alpha = \frac{1}{|N(v)|}$。但是，我们也可以基于图的结构特性显式定义 α。

Attention Mechanism(注意力机制)

设 α_{uv} 为计算注意力机制 a 的副产物，它根据节点对 u, v 的消息计算注意力系数 e_{vu} ：

$$e_{vu} = a(W_k h_u^{k-1}, W_k h_v^{k-1})$$

e_{vu} 表示了节点 u 向节点 v 传递的信息的重要性，然后，我们使用 softmax 函数归一化系数以比较不同邻居之间的重要性：

$$\alpha = \frac{\exp(e_{vu})}{\sum_{k \in N(v)} \exp(e_{vk})}$$

因此有：

$$h_v^k = \sigma(\sum_{u \in N(v)} \alpha_{vu} W_k h_u^{k-1})$$

该方法与的选择的 a 无关，并且可以与 W_k 一起训练参数。

参考

以下是有用的参考资料列表：

教程和概述：

- Relational inductive biases and graph networks (Battaglia et al., 2018)
- Representation learning on graphs: Methods and applications (Hamilton et al., 2017)

基于注意力的邻居节点聚合：

- Graph attention networks (Hoshen, 2017; Velickovic et al., 2018; Liu et al., 2018)

整个图嵌入：

- Graph neural nets with edge embeddings (Battaglia et al., 2016; Gilmer et. al., 2017)
- Embedding entire graphs (Duvenaud et al., 2015; Dai et al., 2016; Li et al., 2018) and graph pooling (Ying et al., 2018, Zhang et al., 2018)
- Graph generation and relational inference (You et al., 2018; Kipf et al., 2018)
- How powerful are graph neural networks(Xu et al., 2017)

节点嵌入：

- Varying neighborhood: Jumping knowledge networks Xu et al., 2018), GeniePath (Liu et al., 2018
- Position-aware GNN (You et al. 2019)

图神经网络的谱方法：

- Spectral graph CNN & ChebNet [Bruna et al., 2015; Defferrard et al., 2016)
- Geometric deep learning (Bronstein et al., 2017; Monti et al., 2017)

其他GNN方法

- Pre-training Graph Neural Networks (Hu et al., 2019)
- GNNExplainer: Generating Explanations for Graph Neural Networks (Ying et al., 2019)