Semi-Supervised Classification with Graph Convolutional Networks

用于图结构的半监督学习的可扩展方法

GCN的最原始论文，基于图卷积

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

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1}\theta_k T_k(\tilde{\Lambda})$$

其中，$\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$；$\lambda_{max}$ 是指拉普拉斯矩阵 L 的最大值。

我们知道归一化的拉普拉斯矩阵的特征值区间为 [0, 2]，进一步近似 $\lambda_{max} = 2$，所以我们有新的表达式：

$$g_\theta * x \approx \theta_0 x + \theta_1(L - I_N)x = \theta_0 x - \theta_1 \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}x$$

在实际训练过程中，我们需要规范化参数来避免过拟合，所以我们令 $\theta = \theta_0' = -\theta_1'$，从而有：

$$g_\theta \star x \approx \theta(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x$$

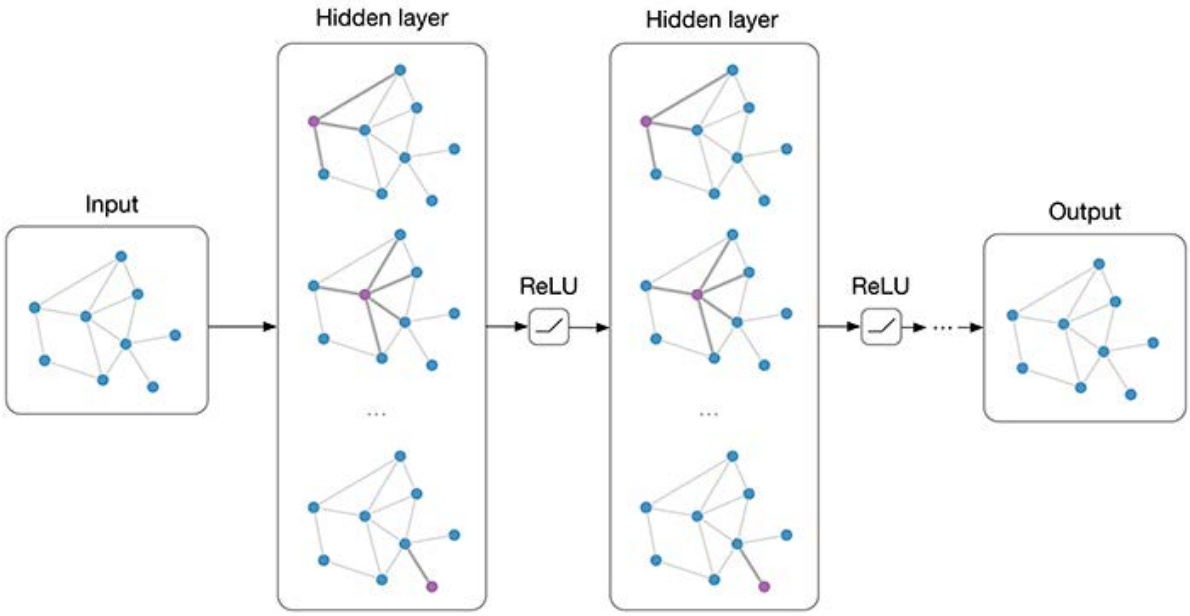
需要注意的是，$I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ 的特征值范围在 [0, 2] 之间，所以如果在很深的网络中会引起梯度爆炸的问题，所以我们需要再次对他进行一次归一化（原文也称 **renormalization trick**）：

$$I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \to D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

我们把这个公式从标量推广到矩阵，对于输入节点的向量 $X \in R^{N\times C}$，其中 N 为节点数，C 为节点的特征向量维度，我们有：

$$Z = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$$

其中，$\Theta \in R^{C\times F}$ 是滤波器的参数矩阵，$Z \in R^{N\times F}$ 是卷积后的信号矩阵，时间复杂度为 $O(|E|FC)$。节点的向量可以通过卷积操作从 C 维度 转变为 F 维度。



由于知道了 GCN 的传播规则，所以我们有最终的结果：

$$Z = f(X, A) = Softmax\left(\hat{A}\ ReLU(\hat{A}XW^{(0)})W^{(1)}\right)$$

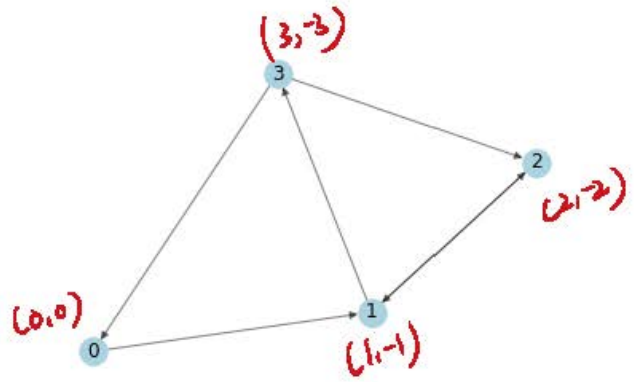
其中，$W^{(0)} \in R^{C\times H}$ 是输入层到隐藏层的权重，$W^{(1)} \in R^{H\times F}$ 是隐藏层到输出层的权重；用 Softmax 是因为这是一个节点分类任务，需要预测标签。

然后，我们用交叉熵作为代价函数：

$$L = -\sum_{l\in y_L}\sum_{f=1}^{F} Y_{lf}\ ln\ Z_{lf}$$

其中，y_L 为有标签的节点集合。

有了代价函数后，我们可以通过梯度下降来更新网络的参数。



邻接矩阵 A 和 节点的特征向量 X 为：

$$\boldsymbol{A}=\begin{bmatrix}0,1,0,0\\0,0,1,1\\0,1,0,0\\1,0,1,0\end{bmatrix}\quad \boldsymbol{X}=\begin{bmatrix}0,0\\1,-1\\2,-2\\3,-3\end{bmatrix}$$

我们有一个简单的传播规则（不考虑参数矩阵和激活函数）：

$$f(\boldsymbol{A},\boldsymbol{X})=\boldsymbol{A}\boldsymbol{H}^{0}\boldsymbol{W}^{0}=\boldsymbol{A}\boldsymbol{X}\boldsymbol{W}^{0}=\boldsymbol{A}\boldsymbol{X}=\begin{bmatrix}1,-1\\5,-5\\1,-1\\2,-2\end{bmatrix}$$

可以看到节点的特征变成了其邻居的特征之和！

但这边也就一些小问题：

1. 这种传播过程没有考虑节点自身的特征；
2. 度的大节点特征值会越来越大，度小的节点特征值会越来越小，传播过程对特征的尺度敏感。

为了解决这个问题，我们需要：

1. 加一个单位矩阵，考虑自环路；
2. 将邻接矩阵 A 与度矩阵 D 的逆相乘对特征进行归一化。

我们先看下加上单位矩阵的效果：

$$f(\boldsymbol{A},\boldsymbol{X})=(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{X}=\begin{bmatrix}1,-1\\6,-6\\3,-3\\5,-5\end{bmatrix}$$

可以看到，加上单位矩阵的计算考虑了节点的特征。

再看下邻接矩阵归一化的效果：

$$f(A,X)=D^{-1}AX=\begin{bmatrix}1,0,0,0\\0,2,0,0\\0,0,2,0\\0,0,0,1\end{bmatrix}^{-1}\begin{bmatrix}0,1,0,0\\0,0,1,1\\0,1,0,0\\1,0,1,0\end{bmatrix}\begin{bmatrix}0,0\\1,-1\\2,-2\\3,-3\end{bmatrix}=\begin{bmatrix}0,1,0,0\\0,0,0.5,0.5\\0,0.5,0,0\\0.5,0,0.5,0\end{bmatrix}\begin{bmatrix}0,0\\1,-1\\2,-2\\3,-3\end{bmatrix}=\begin{bmatrix}1,-1\\2.5,-2.5\\0.5,-0.5\\2,-2\end{bmatrix}$$

邻接矩阵被归一化到 0 到 1 之间。

我们将两个放在一起，并考虑参数矩阵 W：

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

所以我们有：

$$f(A,X)=ReLU\left(D^{-1}(A+I)XW\right)=ReLU\left(\begin{bmatrix}1,-1\\4,-4\\2,-2\\5,-5\end{bmatrix}\right)=\begin{bmatrix}1,0\\4,0\\2,0\\5,0\end{bmatrix}$$

以上便完成了 GCN 的简单仿真。

我们回过头来再来看一下网络的传播规则：

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

现在是不是更能明白为什么这么传播了？

这里解释一下归一化为什么是两边乘上矩阵的 -1/2 次方。

这是因为对称归一化的拉普拉斯矩阵其元素定义为：

$$L_{ij}=\begin{cases}1 & if\ i=j\ and\ deg(v_i)\neq 0\\ -\dfrac{1}{\sqrt{deg(v_i)deg(v_j)}} & if\ i\neq j\ and\ v_i\ is\ adjacent\ to\ v_j\\ 0 & otherwise\end{cases}$$

我们仿真模拟的是用加权求和取平均的方式来聚合，而作者采用的是拉普拉斯变换。我这边做一个化简大家可能个就会明白了：

可以看到，加上单位矩阵的计算考虑了节点的特征。

再看下邻接矩阵归一化的效果：

$$f(A,X)=D^{-1}AX=\begin{bmatrix}1,0,0,0\\0,2,0,0\\0,0,2,0\\0,0,0,1\end{bmatrix}^{-1}\begin{bmatrix}0,1,0,0\\0,0,1,1\\0,1,0,0\\1,0,1,0\end{bmatrix}\begin{bmatrix}0,0\\1,-1\\2,-2\\3,-3\end{bmatrix}=\begin{bmatrix}0,1,0,0\\0,0,0.5,0.5\\0,0.5,0,0\\0.5,0,0.5,0\end{bmatrix}\begin{bmatrix}0,0\\1,-1\\2,-2\\3,-3\end{bmatrix}=\begin{bmatrix}1,-1\\2.5,-2.5\\0.5,-0.5\\2,-2\end{bmatrix}$$

邻接矩阵被归一化到 0 到 1 之间。

我们将两个放在一起，并考虑参数矩阵 W：

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

所以我们有：

$$f(A,X)=ReLU\left(D^{-1}(A+I)XW\right)=ReLU\left(\begin{bmatrix}1,-1\\4,-4\\2,-2\\5,-5\end{bmatrix}\right)=\begin{bmatrix}1,0\\4,0\\2,0\\5,0\end{bmatrix}$$

以上便完成了 GCN 的简单仿真。

我们回过头来再来看一下网络的传播规则：

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

现在是不是更能明白为什么这么传播了？

这里解释一下归一化为什么是两边乘上矩阵的 -1/2 次方。

这是因为对称归一化的拉普拉斯矩阵其元素定义为：

$$L_{ij}=\begin{cases}1 & if\ i=j\ and\ deg(v_i)\neq 0\\ -\dfrac{1}{\sqrt{deg(v_i)deg(v_j)}} & if\ i\neq j\ and\ v_i\ is\ adjacent\ to\ v_j\\ 0 & otherwise\end{cases}$$

我们仿真模拟的是用加权求和取平均的方式来聚合，而作者采用的是拉普拉斯变换。我这边做一个化简大家可能个就会明白了：

Graph neural networks. In recent years, graph neural networks (GNNs) [1, 2, 3] have emerged as a promising approach for analyzing graph-structured data. They follow an iterative neighborhood aggregation (or message passing) scheme to capture the structural information within nodes' neighborhood. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote an undirected graph, with $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times N}$ as the feature matrix where $x_n = \boldsymbol{X}[n,:]^T$ is the N-dimensional attribute vector of the node $v_n \in \mathcal{V}$. Considering a K-layer GNN $f(\cdot)$, the propagation of the kth layer is represented as:

$$a_n^{(k)} = \text{AGGREGATION}^{(k)}(\{h_{n'}^{(k-1)} : n' \in \mathcal{N}(n)\}), \boldsymbol{h}_n^{(k)} = \text{COMBINE}^{(k)}(\boldsymbol{h}_n^{(k-1)}, \boldsymbol{a}_n^{(k)}), \quad (1)$$

where $\boldsymbol{h}_n^{(k)}$ is the embedding of the vertex v_n at the kth layer with $\boldsymbol{h}_n^{(0)} = x_n$, $\mathcal{N}(n)$ is a set of vertices adjacent to v_n, and $\text{AGGREGATION}^{(k)}(\cdot)$ and $\text{COMBINE}^{(k)}(\cdot)$ are component functions of the GNN layer. After the K-layer propagation, the output embedding for $\mathcal{G}$ is summarized on layer embeddings through the READOUT function. Then a multi-layer perceptron (MLP) is adopted for the graph-level downstream task (classification or regression):

$$f(\mathcal{G}) = \text{READOUT}(\{\boldsymbol{h}_n^{(k)} : v_n \in \mathcal{V}, k \in K\}), \boldsymbol{z}_{\mathcal{G}} = \text{MLP}(f(\mathcal{G})). \quad (2)$$

Inductive Representation Learning on Large Graphs

GraphSAGE

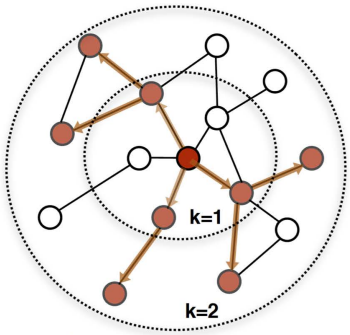
通过采样和聚合节点的本地邻域要素来生成嵌入

比GCN能够适应网络的变化，不用重新训练网络参数

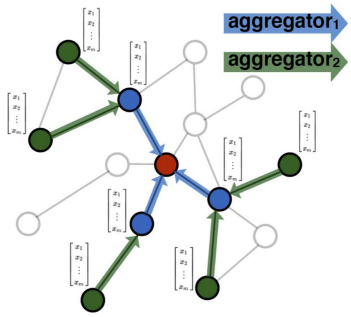
GraphSAGE是利用一组聚合函数进行学习，这些聚合函数可以从节点的邻域中学习到节点的特征信息，即使是新节点也可以通过其邻域信息进行学习

利用节点的属性（文本属性）来生成节点嵌入

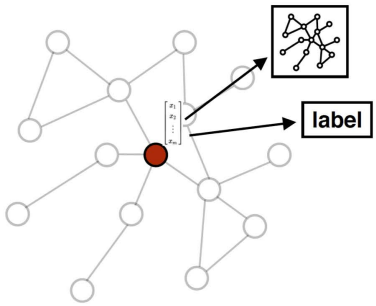
识别节点邻域的结构属性同时显示本地角色及其全局位置



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

1. 首先对节点的一阶和二阶邻居节点进行采样
2. 然后根据聚合函数聚合邻居节点的特征
3. 最后得到节点的Embedding向量

聚合邻居信息，然后不断迭代

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
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

第一行是我们要计算的节点的特征输入

第二行是第一个for循环遍历深度，可以理解为网络的层数

第三行是第二个for循环是遍历图中的所有节点

第四行是从上一层神经网络中利用聚合函数聚合当前节点的新特征

第五行是将当前节点的特征和邻居特征拼接并经过一个全连接网络得到当前节点的新特征

第七行是归一化

第八行是通过k层GCN后进行输出

用k-1层的节点邻居信息和自身信息更新k层节点的信息

为了使用SGD，通过mini-batch的方法来正向和反向传播

Algorithm 2: GraphSAGE minibatch forward propagation algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$;
input features $\{\mathbf{x}_v, \forall v \in \mathcal{B}\}$;
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 sampling functions, $\mathcal{N}_k : v \rightarrow 2^{\mathcal{V}}, \forall k \in \{1, ..., K\}$

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

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

Loss

学习节点的 Embedding 向量是一个非监督学习，我们希望节点与较近的节点的 Embedding 向量相似，而与较远的节点不相似，其损失函数为：

$$J(z_u) = -log(\sigma(z_u^T z_v)) - Q \cdot E_{v_n \sim P_n(v)} log(\sigma(-z_u^T z_{v_n}))$$

其中，节点 v 是节点 u 在定长随机游走算法中邻居节点；P_n 是负采样分布，Q 定义了负采样的个数。

聚合器

对于一个无序网络来说，理想的聚合器是对称的，即：不考虑节点顺序。同时也需要保证 Embedding 向量具有较好的效果。这里作者提出了三种不同的聚合器：

1. **Mean aggregator**

均值聚合器，将邻居节点和当前节点的 Embedding 向量取均值，这与我们先前介绍的 GCN 有很多相似的地方（都是基于邻居节点进行聚合）。

1. **LSTM aggregator**

LSTM 聚合器，因为 LSTM 具有强大的表达能力，但是 LSTM 是非对称的，为了解决这个问题，我们将只需将 LSTM 应用于节点邻居的随机排列，即可使 LSTM 适应无序集合。

1. **Pooling aggregator**

池化聚合器，这是作者最终使用的聚合器。这种聚合方式可以使得节点的每个邻居的 Embedding 向量都可以独立的通过全连接的神经网络，通过这样的转换后最大池化操作可以聚合整个邻居集合。

GraRep: Learning Graph Representations with Global Structural Information

使用全局结构信息学习图形表示

学习低维向量来表示图形中出现的顶点，将图形的全局结构信息集成到学习过程中

图形中的每个顶点都由一个低维向量表示，在该向量中可以准确捕获图形所传达的有意义的语句，关系和结构信息

具有不同K值（K阶邻居）的不同顶点之间的K步关系与图相关的有用的全局结构信息。

定义了不同的损失函数--> 捕获不同的k 阶本地关系信息(即不同的k)

优化每个模型,通过组合从不同模型中学到的不同表示来构造每个顶点的全局表示-->全局表示可以用作进一步处理的特征

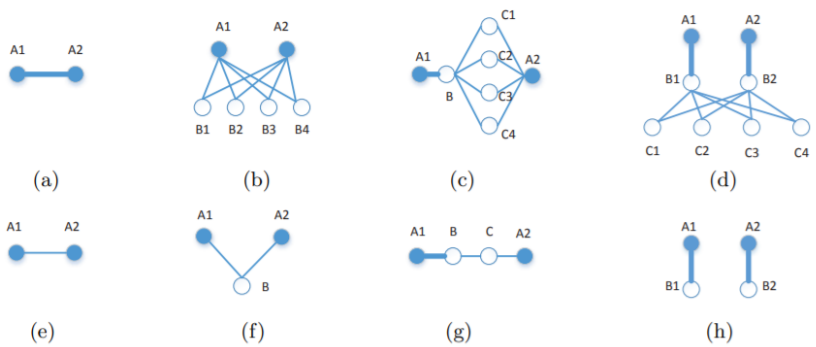


Figure 1: The importance of capturing different k-step information in the graph representations. Here we give examples for $k = 1, 2, 3, 4$.

Graph Attention Network

Attention用在网络图中,可以对邻居中不同的节点指定不同的权重,不需要进行矩阵运算,也不需要事先了解图的全局结构

卷积的两种方式

1. 基于空域的卷积,直接在网络中进行建模
2. 基于频域的卷积,将网络映射到频域后利用卷积变换进行建

注意力机制可以处理可变大小的输入,将注意力集中在最相关的输入部分