

# — GCN 算法整理

## 1. GCN 原理

Graph Convolutional Networks (GCN)涉及到两个重要的概念，图和卷积。传统的卷积主要应用于欧式空间的数据上（排列很整齐、网格形式的），如图像、语音等，主要是因为欧式结构数据能够保证卷积的性质，即平移不变性，而非欧式空间无法保证平移不变性，通俗理解就是在拓扑图中每个顶点的相邻顶点数目都可能不同，那么当然无法用一个同样尺寸的卷积核来进行卷积运算。为了能够将卷积推广到图等非欧式空间数据上，GCN 应运而生。那么 GCN 是如何将卷积推广到图上的呢？

① 卷积和傅里叶变换有着密不可分的关系。在数学上，两个函数的卷积等于各自求傅里叶变换转成频域后乘积的逆傅里叶变换

② 傅里叶变换又可以通过图谱理论推广到图上进行变换。

图上的信号一般表达为一个向量。假设有  $n$  个节点，我们将图上的信号记为：

$$x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n \quad (1)$$

每一个节点上有一个信号值。类似于图像上的像素值。 $i$  节点上的值为  $x(i) = x_i$ 。

连续形态下的傅里叶变换和反变换：

$$F(\omega) = F[f(t)] = \int_R f(t) d^{-2\pi i \omega t} dt \quad (2)$$

$$f(w) = F^{-1}[F(w)] = \int_R F(t) d^{2\pi i \omega t} dt \quad (3)$$

离散形态下的傅里叶变换和反变换：

$$F(\omega) = \sum_{t=1}^n f(t) e^{-i \frac{2\pi}{n} \omega t} \quad (4)$$

$$F(t) = \frac{1}{n} \sum_{\omega=1}^n F(\omega) e^{i \frac{2\pi}{n} \omega t} \quad (5)$$

傅里叶变换中不同频率的余弦函数可视为基函数，其傅里叶系数表示基的振幅，如图 1：



图 1 信号傅里叶变换

在经典傅里叶变换中，让我们来看一下本质何在：

傅里叶反变换的本质是：把任意一个函数表示成了若干个正交基函数的线性组合。

傅里叶正变换的本质是：求线性组合的系数。也就是由原函数和基函数的共轭的内积求得。

### 图傅里叶变换

傅里叶反变换的本质是把任意一个函数表示成了若干个正交基函数的线性组合。对应图 (graph) 上的信号  $x \in \mathbb{R}^n$ ，如果要进行一个傅里叶变换，那就需要我们也找到一组正交基，通

过这组正交基的线性组合来表达  $x \in \mathbb{R}^n$ 。

图傅里叶变换, 实际上使用的是拉普拉斯矩阵的特征向量, 作为图傅里叶变换的基函数:

$$U = (\vec{\mu}_1, \vec{\mu}_2, \dots, \vec{\mu}_n) \quad (6)$$

使用拉普拉斯的特征向量作为基函数, 任意的图上的信号可以表示为:

$$x = \hat{x}(\lambda_1)\vec{\mu}_1 + \hat{x}(\lambda_2)\vec{\mu}_2 + \dots + \hat{x}(\lambda_n)\vec{\mu}_n \quad (7)$$

拉普拉斯矩阵是用来研究图结构性质的核心对象, 其定义为  $L = D - A$ , 其中  $D$  是一个对角矩阵,  $A$  是邻接矩阵,  $D_i = \sum_j A_{i,j}$  表示  $i$  节点的度。拉普拉斯矩阵还有一种正则化的形式(symmetric normalized laplacian)。

$$\mathcal{L}_{sym} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \quad (8)$$

拉普拉斯矩阵的定义源自于拉普拉斯算子, 拉普拉斯算子它是  $n$  维欧式空间种的二阶微分算子, 在二维空间的图像中表示就是我们熟悉的边缘检测算子

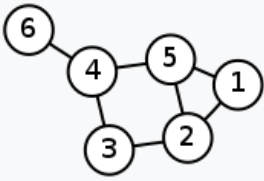
Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

图 2 拉普拉斯矩阵计算

回到公式 (8), 从表达式上我们可以清晰的看到, 正则部分相当于衡量相邻节点之间的差异性, 因为  $A_{i,j}$  是邻接矩阵上的元素, 两个节点处于邻接关系时为 1, 其他为 0. 这样的正则项, 使得拓扑图上相邻节点尽可能相似, 物以类聚, 这是很自然的想法。从信号的角度上来看, 减少该正则项, 就是期望经过模型之后的图信号更加平滑。从频域上来看, 是对图信号做了低通滤波的处理, 但是这样的假设可能会限制模型的表达能力, 因为图上边不只包含相似度的信息, 还可能包含其他相关信息。

最初标准的**第一代 GCN** 出自《Spectral Networks and Locally Connected Networks on Graphs》, 其对于图上的卷积有如下定义:

$$g_\theta * x = Ug_\theta U^T x \quad (8)$$

这里  $U$  是图的标准化拉普拉斯 (Laplacian) 矩阵  $\mathcal{L}_{sym}$  (公式 2) 的特征向量矩阵, 因此有  $\mathcal{L}_{sym} = U\Lambda U^T$ ,  $x$  是数据中提取出来的特征, 也就是输入, 定义  $g_\theta = \text{diag}(\hat{h}(\lambda_1))$  是特征值的响应函数, 进一步为了类比 CNN 中的共享卷积核的设计, 在神经网络中将其设置为可训练参数的卷积核, 如  $\text{diag}(\theta_1)$ 。而  $U^T x$  的本质是将  $x$  映射到傅里叶空间,  $U$  是傅里叶空间的一组正交基对于第一代版本的 GCN 主要有以下几点弊端:

- (1) 每一层的计算涉及  $U\text{diag}(\hat{h}(\theta_1))U^T$  三个矩阵乘法, 复杂度为  $O = (N^2)$ , 代价较高
- (2) 卷积核具有  $n$  个参数,  $n$  是  $L$  的特征值个数。
- (3) 计算 Hermitian Matrix 特征值的代价非常昂贵

过高的计算复杂度和过大的计算参数, 在实际运用中都是不允许的, 于是基于以上两点的考虑, 就产生了**第二代版本的 GCN**

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k = \begin{pmatrix} \sum_{j=0}^{k-1} \theta_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^{k-1} \theta_j \lambda_n^j \end{pmatrix} \quad (9)$$

经过这样的设计，原本需要训练  $n$  个参数，缩小到了  $K$  个，同时将公式 (4) 带入公式 (3) 得到公式 (10)，从公式 (10) 中不难发现，此时我们不在需要在求矩阵的特征向量了，可直接使用拉普拉斯矩阵进行运算。第二代版本的 GCN 解决了第一代的 2、3 两个问题的弊端，但是矩阵乘法的复杂度依旧是  $O = (N^2)$ 。

$$g_{\theta} * x = \sum_{j=0}^{k-1} \theta_j L^j x \quad (10)$$

计算多个大型稠密矩阵的乘法复杂度是很高的，这运用中在实际场景是不被允许的，尤其是计算像社交关系网络这种超大拓扑图的数据，显然是无法实现的，为解决这个问题，我们采用切比雪夫多项式 (Chebyshev polynomials) 来近似  $g_{\theta}(\Lambda)$ ，

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^K \theta'_k T_k(\tilde{\Lambda}) \quad (11)$$

这里  $\tilde{\Lambda} = \frac{2}{\lambda_{max}} \Lambda - I_N$  是对  $\Lambda$  的一种缩放， $\lambda_{max}$  是  $L$  的最大特征值， $\theta' \in \mathcal{R}^k$  是切比雪夫系数向量，切比雪夫的递归式定义为  $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ ，设  $T_0(x) = 1, T_1(x) = x$ ，由此就可以的得到一个计算近似特征值的方法，将公式(6)带入之前的卷积定义公式(3)，我们有：

$$g_{\theta} * x = \sum_{k=0}^K \theta'_k T_k(\tilde{L}) x \quad (12)$$

这里  $\tilde{L} = \frac{2}{\lambda_{max}} L - I_N$ ，这个表达式是局部化的，它是拉普拉斯式的一个  $K$  阶多项式，也就是说，它只依赖于距离中心节点( $K$  阶邻域)，公式 (14) 的复杂度是  $O = (\mathcal{E}^2)$ ，即计算复杂度取决于图的边的数量。

为了简化问题，同时减轻由于局部节点度过大而导致过拟合的问题，我们设置  $K=1$ ，即只考虑中心节点的一阶邻近，将  $T_0(x) = 1, T_1(L) = \tilde{L}$  带入公式(12)，进一步，我们将尺度变换的  $\lambda_{max}$  固定为 2。于是我们就得到卷积新的近似表达

$$g_{\theta} * x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta \left( I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x \quad (13)$$

由盖尔圆盘定理我们知道，公式 (13) 中  $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$  的特征值范围为  $[0, 2]$ ，重复的进行卷积操作，可能会导致数值不稳定，在神经网络中多层的图卷积之后，容易导致梯度消失或者梯度爆炸。为了解决这个问题，我们将其进一步标准化：

$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \quad (14)$$

其中  $\tilde{A} = A + I_N$ ， $\tilde{D} = \sum_j \tilde{A}_{ij}$ ，此时我们就将其特征值规范到了  $[0, 1]$  之间。可以看到， $\tilde{A}$  相当于每个节点增加了自连接关系，直观上来看，就是每个节点的跟新与邻近节点的特征以及节点上一层的特征有关。将公式 (14) 带入公式 (13) 我们的图卷积表达式最终可写成如下形式

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Phi \quad (15)$$

## 2. GCN 的应用场景

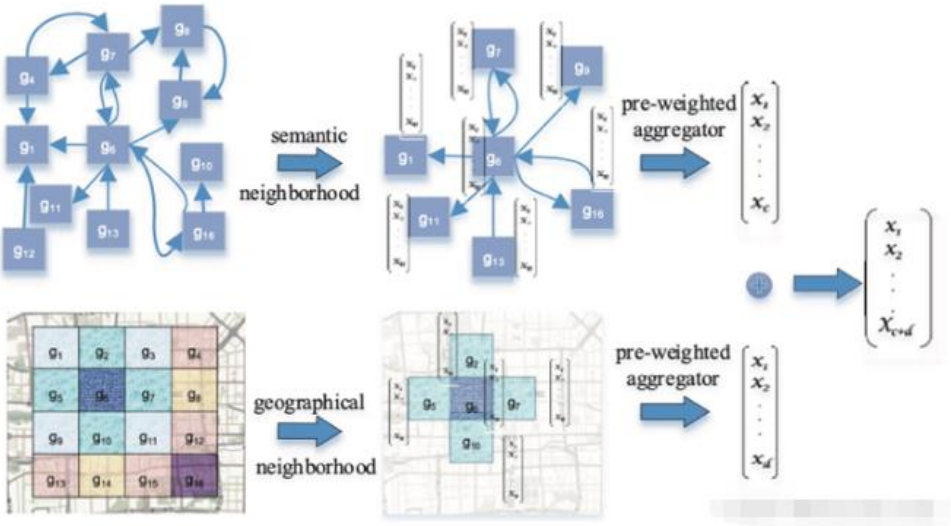
(1) 交通 (交通流量预测、出租车或网约车需求预测、交通流异常检测、停车位可用性预测)

交通流量预测：交通流量预测问题，即给定前  $T$  个时刻的历史流量信息，预测未来  $T$  时刻的交通信息。通过图卷积操作，结合交通系统层次结构，包括道路网络的微观层和宏观层来进行流量的预测。

出租车或网约车需求预测：进行出租车或网约车 OD 需求矩阵的预测，OD 需求矩阵的预测比一般的需求预测更加具有挑战性，预测一个区域用车需求的同时还要预测乘客目的地的分布。

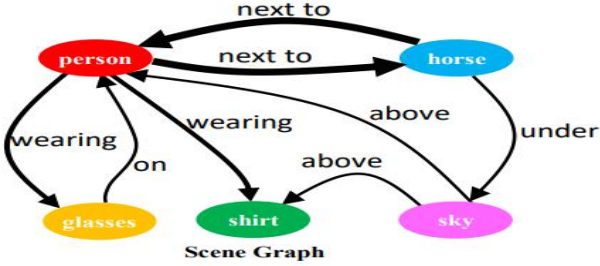
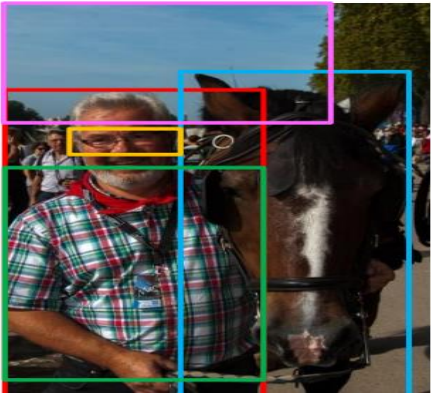
交通流异常检测：交通流异常检测通常要考虑时间信息、空间信息等信息，这让交通流异常检测变得具有挑战性。可以利用交通的时空数据，空间信息采用图卷积网络捕获，时间依赖性采用深度神经网络建模。同时捕捉时空特性并建立预测交通流模型，利用异常分数来判断由交通事故、或短暂事件引起的非经常性交通流异常。

停车位可用性预测：对城市停车位可用性的有效预测可以提高停车效率，帮助城市规划，最终缓解城市拥堵。该任务面临非欧几里得的停车场空间自相关性（停车场的可用性会受附近的停车场占用情况的影响）问题。可以通过一个层次图卷积模块来捕获停车场之间的非欧几里得空间相关性，分别捕获局部空间依赖性和全局依赖性。



## (2) 图像(视觉关系检测)

视觉关系检测：在进行目标检测时，很多时候个目标框之间是存在相对位置关系的，结合 GCN 可以利用这部分信息来提高检测的效果。比如 person 框应该是包括 hat 框和 uniform 框的，且框的出现具有一定的共现关系。定义目标框之间的关系如 contain、overlap 及 no-relation 等，通过图卷积来捕获其相互之间的位置关系和共现关系，提高各目标之间的相互依赖以提高检测效果。



Relationships	Attention Weight
<person, next to, horse>	0.3
<horse, next to, person>	0.3
<person, wearing, shirt>	0.1
<person, wearing, glasses>	0.1
<sky, above, shirt>	0.05
.....	

### (3) 推荐(用户商品偏好推荐)

用户商品偏好推荐: GCN 可以弥补传统神经网络无法处理的(如社交关系、电商交易路径)非规则的结构数据,即图数据。在利用 GCN 提取关联关系的同时,结合商品评论来提取用户对商品的偏好,并进行特征融合可以提高推荐的效果。

### (4)文本(文本分类、信息抽取)

文本分类:将文本中的 word 和 document 作为图中的 node,利用 co-occurrence 信息来构建 edge,可以将文本分类问题看作是 node 分类问题。以词和文档作为顶点,利用词在文档中的 TF-IDF 值作为权重,来表示一个词在某个文本中的重要程度,连接词与词的边,用词与词的互信息作为权重,来表示词之间的相关性,以此建立邻接矩阵,然后通过两层图卷积网络即能有效提取图谱顶点中的特征表示来进行文本分类。

信息抽取:通过双向 LSTM 和 GCN 分别抽取文本的顺序特征和区域依赖特征,通过对每种关系构建完整的关系加权图,以每个词作为图的节点,边表示单词之间的关系概率,再使用 GCN 融合每种关系的信息,同时融入实体和关系之间的交互信息,来预测实体和每个单词对之间的关系,这种基于图卷积的实体关系抽取模型可以有效的解决关系重叠和多关系问题。

## 3. 总结

实际上 GCN 实际上跟 CNN 的作用一样,就是一个特征提取器,只不过它的对象是图数据,GCN 精妙地设计了从图数据中提取特征的方法,让我们可以使用这些特征去对图数据进行节点分类、图分类、边预测,以及得到相应的图嵌入,而现实生活中很多重要的数据都是以网络结构的方式产生,利用图数据的特征提取器来处理基于这部分数据的任务也理所当然。而对于图像、文本等任务,通过抽象位置、共现、上下文等关系形成图结构,通过 GCN 来进行特定或额外特征提取和补充,也让其成为除了 RNN、CNN 及 transformer 等特征提取器外一种新的选择和尝试方向。

**缺点:**① GCN 需要将整个图放到内存和显存,这将非常耗内存和显存,处理不了大图。

② GCN 在训练时需要知道整个图的结构信息(包括待预测的节点)。

## 4. 实现(见附件)

[https://github.com/Frank-qlu/GNN/blob/main/gcn/gcn\\_dgl.py](https://github.com/Frank-qlu/GNN/blob/main/gcn/gcn_dgl.py)

## 二 graphSAGE 算法整理

### 1. graphSAGE 原理

graphSAGE 是 inductive learning (归纳学习), 所谓归纳学习是指可以对训练过程中见不到的数据直接计算而不需要重新对整个图进行学习。而 GCN 是 transductive learning (直推式学习), 它是指所有的数据都可以在训练的时候拿到, 学习的过程是在这个固定的图上进行学习, 一旦图中的某些节点发生变化, 则需要对整个图进行重新训练和学习。

graphSAGE 是基于 GCN 的改进策略, 它对 GCN 进行了两点改进:

①子图随机组合训练: 将 GCN 的全图训练方式改造成以节点为中心的小批量训练方式, 使得模型可以在大规模的图数据上进行训练, 并且可以在新的图结构上做预测, 大大提供了工业场景的应用性

②邻居聚合操作拓展: 提出了替换卷积操作的其他集中提取邻居特征的方式。

GraphSAGE 的算法核心是将整张图的采样优化到当前邻居节点的采样, 因此我们从邻居采样和邻居聚合两个方面来对 GraphSAGE 进行解释。

在 GraphSAGE 之前的 GCN 模型中, 都是采用的全图的训练方式, 也就是说每一轮的迭代都要对全图的节点进行更新, 当图的规模很大时, 这种训练方式无疑是很耗时甚至无法更新的。mini-batch 的训练时深度学习一个非常重要的特点, 那么能否将 mini-batch 的思想用到 GraphSAGE 中呢, GraphSAGE 提出了一个解决方案。它的流程大致分为 3 步:

① 对邻居进行随机采样, 每一跳抽样的邻居数不多于  $S_k$  个, 如图 1.2 第一跳采集了 3 个邻居, 第二跳采集了 5 个邻居;

② 生成目标节点的 embedding: 先聚合二跳邻居的特征, 生成一跳邻居的 embedding, 再聚合一跳的 embedding, 生成目标节点的 embedding;

③ 将目标节点的 embedding 输入全连接网络得到目标节点的预测值。

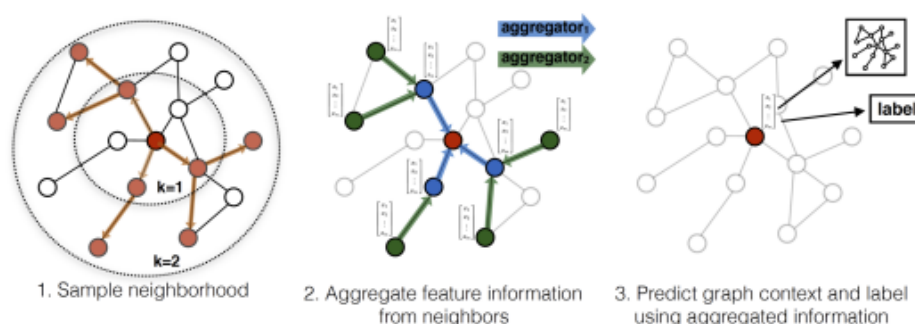


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

从上面的介绍中我们可以看出, GraphSAGE 的思想就是不断的聚合邻居信息, 然后进行迭代更新。随着迭代次数的增加, 每个节点的聚合的信息几乎都是全局的, 这点和 CNN 中感受野的思想类似。GraphSAGE 的伪代码在论文的附录 A 中:



---

**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  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$ ; non-linearity  $\sigma$ ; differentiable aggregator functions  $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$ ; neighborhood function  $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$   
**Output** : Vector representations  $\mathbf{z}_v$  for all  $v \in \mathcal{V}$

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;  
2 for  $k = 1 \dots K$  do  
3   for  $v \in \mathcal{V}$  do  
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;  
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$   
6   end  
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$   
8 end  
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

---

---

**Algorithm 2:** GraphSAGE minibatch forward propagation algorithm

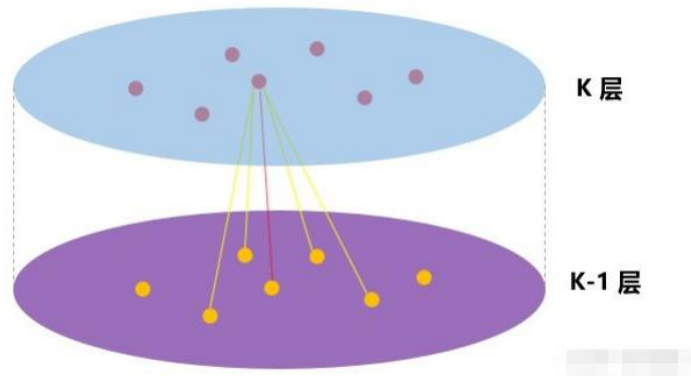
---

**Input** : Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ ;  
input features  $\{\mathbf{x}_v, \forall v \in \mathcal{B}\}$ ;  
depth  $K$ ; weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$ ;  
non-linearity  $\sigma$ ;  
differentiable aggregator functions  $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$ ;  
neighborhood sampling functions,  $\mathcal{N}_k : v \rightarrow 2^{\mathcal{V}}, \forall k \in \{1, \dots, K\}$   
**Output** : Vector representations  $\mathbf{z}_v$  for all  $v \in \mathcal{B}$

```
1  $\mathcal{B}^K \leftarrow \mathcal{B}$ ;  
2 for  $k = K \dots 1$  do  
3    $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^k$ ;  
4   for  $u \in \mathcal{B}^k$  do  
5      $\mathcal{B}^{k-1} \leftarrow \mathcal{B}^{k-1} \cup \mathcal{N}_k(u)$ ;  
6   end  
7 end  
8  $\mathbf{h}_u^0 \leftarrow \mathbf{x}_u, \forall u \in \mathcal{B}^0$ ;  
9 for  $k = 1 \dots K$  do  
10  for  $u \in \mathcal{B}^k$  do  
11     $\mathbf{h}_{\mathcal{N}(u)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_{u'}^{k-1}, \forall u' \in \mathcal{N}_k(u)\})$ ;  
12     $\mathbf{h}_u^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_u^{k-1}, \mathbf{h}_{\mathcal{N}(u)}^k))$ ;  
13     $\mathbf{h}_u^k \leftarrow \mathbf{h}_u^k / \|\mathbf{h}_u^k\|_2$ ;  
14  end  
15 end  
16  $\mathbf{z}_u \leftarrow \mathbf{h}_u^K, \forall u \in \mathcal{B}$ 
```

---

补充一个形象的示意图:



每一层的 node 的表示都是由上一层生成的，跟本层的其他节点无关。

### 采样

出于对计算效率的考虑，需要对每个节点的邻居节点进行采样，采样时主要包含两个要素：层数  $K$ （跳数）和每一层上每个节点要采样的数量，训练时可以认为每个 batch 的时空复杂度为  $\prod_i^K S_i$ ，实验发现， $K$  不必取很大的值，当  $K=2$  时，以及两次扩展的邻居节点总数  $S_1 \cdot S_2$  小于等于 500 时，效果就很好了。

由于要对每个顶点采样一定数量的邻居节点，设需要的邻居节点数量为  $S$ ，若顶点邻居节点数小于  $S$ ，则采用有放回的抽样方法，直到采样出  $S$  个顶点，若顶点邻居数大于  $S$ ，则采用无放回的抽样。

### 聚合函数

由于在图中顶点的邻居是天然无序的，所以我们希望构造出的聚合函数是对称的（即改变输入的顺序，函数的输出结果不变），同时具有较高的表达能力。graphSAGE 主要有以下四个聚合函数：

① **mean aggregator** 是将目标顶点和邻居顶点的第  $k-1$  层向量拼接起来，然后对向量的每个维度进行求均值的操作，将得到的结果做一次非线性变换产生目标顶点的第  $k$  层表示向量。

文中用下面的式子替换算法 1 中的 4 行和 5 行得到 GCN 的 inductive 变形：

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})).$$

● 均值聚合近似等价在 transductive GCN 框架[Semi-supervised classification with graph convolutional networks. In ICLR, 2016]中的卷积传播规则。

● 文中称这个修改后的基于均值的聚合器是 convolutional 的，这个卷积聚合器和文中的其他聚合器的重要不同在于它没有算法 1 中第 5 行的 CONCAT 操作——卷积聚合器没有将顶点前一层表示  $\mathbf{h}_v^{k-1}$  和聚合的邻居向量  $\mathbf{h}_{N(v)}^{k-1}$  拼接起来。



● 拼接操作可以看作一个是 GraphSAGE 算法在不同的搜索深度或层之间的简单的 skip connection[Identity mappings in deep residual networks]的形式，它使得模型获得了巨大的提升。

● 举个简单例子，比如一个节点的 3 个邻居的 embedding 分别为 [1,2,3,4],[2,3,4,5],[3,4,5,6]按照每一维分别求均值就得到了聚合后的邻居 embedding 为 [2,3,4,5]。

② **LSTM aggregator** 文中也测试了一个基于 LSTM 的复杂的聚合器[Long short-term memory]。和均值聚合器相比，LSTMs 有更强的表达能力。但是，LSTMs 不是 symmetric 的，也就是不具有排列不变性 (permutation invariant)，排列不变性 (permutation invariance)：指输入的顺序改变不会影响输出的值。因为它们以一个序列的方式处理输入。因此，需要先对邻居节点随机顺序，然后将邻居序列的 embedding 作为 LSTM 的输入。

③ **Pooling aggregator**，它既是对称的，又是可训练的。Pooling aggregator 先对目标顶点的邻居顶点的 embedding 向量进行一次非线性变换，之后进行一次 pooling 操作 (max pooling or mean pooling)，将得到结果与目标顶点的表示向量拼接，最后再经过一次非线性变换得到目标顶点的第 k 层表示向量。

一个 element-wise max pooling 操作应用在邻居集合上来聚合信息：

$$h_{N(v)}^k = AGGREGATE_k^{pool} = \max(\{\sigma(W_{pool}h_u^{k-1} + b), \forall u \in N(v)\})$$
$$h_v^k \leftarrow \sigma(W^k \cdot \text{CONCAT}(h_v^{k-1}, h_{N(v)}^k))$$

其中：max 表示 element-wise 最大值操作，取每个特征的最大值。 $\sigma$  是非线性激活函数。所有相邻节点的向量共享权重，先经过一个非线性全连接层，然后做 max-pooling。按维度应用 max/mean pooling，可以捕获邻居集上在某一个维度的突出的综合的表现。

## 2. 总结

GraphSAGE 是学习 GNN 绕不过的一篇经典论文，它最大的贡献在于给图模型赋予了归纳学习的能力，从而大范围的扩大了 GNN 的落地场景。另外 GraphSAGE 的预测速度非常快，因为它只需要选择若干跳的若干个邻居即可，而这两个可以选择的参数往往也比较小，往往取两跳邻居就能得到不错的学习效果。GraphSAGE 的第三个优点是它非常好理解，不需要复杂的图理论基础，对于学习图神经网络也是非常好的入门读物。

不可否认，GraphSAGE 仍然有很多可以提升的地方，例如更加高效的聚合器，更加合理的采样方法等。

### 三 GAT 算法整理

#### 1. GAT 原理

为了解决 GNN 聚合邻居节点的时候没有考虑到不同的邻居节点重要性不同的问题，GAT 借鉴了 Transformer 的 idea，引入 masked self-attention 机制，在计算图中的每个节点的表示的时候，会根据邻居节点特征的不同来为其分配不同的权值。

##### 1.1 注意力系数计算

与 nlp 中的 attention 机制，基本思路相同，都是先计算注意力系数，然后进行加权求和。然后映入多头机制，增强学习能力。

首先，计算顶点  $i$  与邻居节点  $j (j \in \mathfrak{N}_i)$  之间的相似系数。

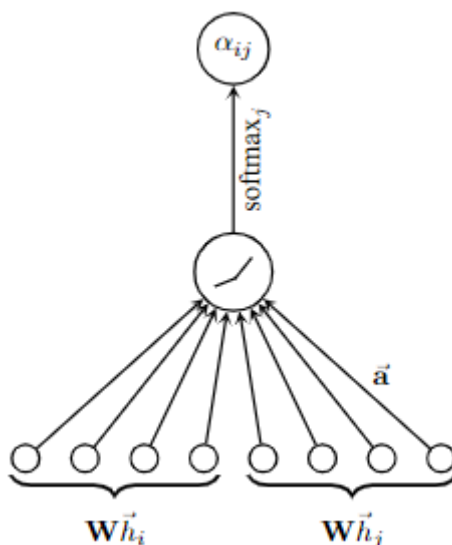
$$e_{i,j} = a([W\vec{h}_i \parallel W\vec{h}_j]) \text{ st. } j \in \mathfrak{N}_i \quad (1)$$

其中， $W$  是共享参数，相当于对  $h_j$  特征进行了增强， $[\parallel]$  表示对变换后的特征进行 concatenate， $a(\cdot)$  把拼接后的特征映射到一个实数上。通过这种方式，选取合适  $W$  和  $a()$ ，可以有效的捕捉到节点  $i, j$  之间的相关性。

在进行加权求和之前还要进行归一化操作：

$$\begin{aligned} \alpha_{i,j} &= \text{softmax}_j(e_{i,j}) \\ &= \frac{\exp(e_{i,j})}{\log_{k \in \mathfrak{N}_i} \exp(e_{i,k})} \\ &= \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T([W\vec{h}_i \parallel W\vec{h}_j])\right)\right)}{\log_{k \in \mathfrak{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^T([W\vec{h}_i \parallel W\vec{h}_k])\right)\right)} \end{aligned} \quad (2)$$

整个过程如下图所示：



## 1.2 加权求和

直接根据得到的权重，对邻居特征进行加权求和，如式(3):

$$\vec{h}_i = \sigma \left( \sum_{j \in \mathbb{N}_i} \alpha_{i,j} W \vec{h}_j \right) \quad (3)$$

这样，GAT 输出的对于每个顶点  $i$  的新特征就融合了邻域信息。

## 1.3 引入多头注意力机制

通过初始化 $k$ 个不同的权重  $W$ ，从不同的角度对原有的 **embedding** 进行增强，提取到不同的注意力参数，再进行加权求和。对于不同的注意力系数，又有两种方法对邻居节点进行聚合。

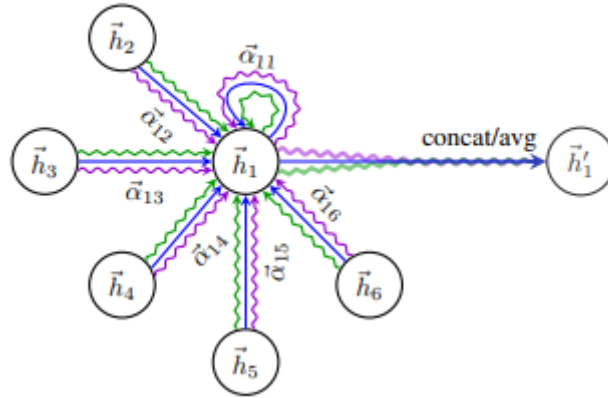
1) 直接横向拼接，这样新的 **embeddin** 向量的维度是原来的 $k$ 倍:

$$\vec{h}_i = \parallel_{k=1}^K \sigma \left( \sum_{j \in \mathbb{N}_i} \alpha_{i,j}^k W^k \vec{h}_j \right) \quad (4)$$

2)  $k$  个注意力机制的结果取均值:

$$\vec{h}_i = \sigma \left( \frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathbb{N}_i} \alpha_{i,j}^k W^k \vec{h}_j \right) \quad (5)$$

具体计算过程如图:



## 2. 总结

- (1) 训练 GAT 无需了解整个图结构，只需知道每个节点的邻居节点即可
- (2) 计算速度快，可以在不同的节点上进行并行计算
- (3) 既可以用于 Transductive Learning，又可以用于 Inductive Learning，可以对未见过的图结构进行处理

## 三 GIN 算法整理

### 1. GIN 原理

GNN 目前主流的做法是递归迭代聚合一阶邻域表征来更新节点表示，如 GCN 和 GraphSAGE，但这些方法大多是经验主义，缺乏理论去理解 GNN 到底做了什么，还有什么改进空间。<< How Powerful are Graph Neural Networks? >>这篇论文基于 Weisfeiler-Lehman(WL) test 视角理论分析了 GNN，该文的贡献：

- ① 证明了 GNN 最多只和 Weisfeiler-Lehman (WL) test 一样有效，即 WL test 是 GNN 性能的上限
- ② 提供了如何构建 GNN，使得和 WL 一样有效
- ③ 用该框架分析了 GCN 和 GraphSAGE 等主流 GNNs 在捕获图结构上的不足和特性
- ④ 建立了一个简单的神经结构——图同构网络(GIN)，并证明了它的判别/表达能力和 WL 测试一样

该文提出了一个理论框架去分析 GNNs 的表达能力。在学习表示和区分不同的图结构时，描述了不同 GNN 变体的表达能力。Weisfeiler-Lehman 图同构测试(1968)(WL)是一种强大的检验方法，可以区分大量的图。与 GNNs 类似，WL 测试通过聚集网络邻居的特征向量迭代地更新给定节点的特征向量。WL 测试之所以如此强大，是因为它的单射聚合更新将不同的节点邻居映射到不同的特征向量。作者的主要观点是，如果 GNN 的聚合方案具有高度表达性，并且能够对单射函数建模，那么 GNN 可以具有与 WL 测试同样大的区分能力。

为了在数学上形式化上述观点，文中提出的框架首先将给定节点的邻居的特征向量集表示为一个 multiset，即，一个可能有重复元素的集合。然后，可以将 GNNs 中的邻居聚合看作是 multiset 上的聚合函数。因此，为了拥有强大的表示能力，GNN 必须能够将不同的 multiset 聚合到不同的表示中。文中严格地研究了 multiset 函数的几个变体，并从理论上描述了它们的区分能力，即，不同的聚合函数如何区分不同的 multiset。multiset 函数的判别能力越强，GNN 的表示能力就越强。

GNNs 的表达能力是捕获图结构的关键。文中通过在图分类数据集上的实验来验证理论，对比了使用各种聚合函数的 GNNs 的性能。

实验结果表明，在作者的理论中最强大的 GNN，即图同构网络 (GIN)，根据经验判断也具有很高的表示能力，因为它几乎完全适合训练数据，而较弱的 GNN 变体往往严重不适合训练数据。此外，这种表达能力更强的 GNN 在测试集精度方面优于其他 GNNs，并且在许多图分类 benchmarks 上实现了最先进的性能。

#### Multiset 定义

Multiset 是一个广义的集合概念，它允许有重复的元素。更正式地说，一个 multiset 是一个 2 元组  $X = (S, m)$ ，其中  $S$  是由  $X$  的不同的元素组成的子集， $m: S \rightarrow \mathbb{N} \geq 1$  表示了元素的多样性。文中的 multiset 就是节点邻居的特征向量集。

#### 数学上的单射 (injective)

在数学里，单射函数为一函数，其将不同的引数连接至不同的值上。更精确地说，函数  $f$  被称为是单射时，对每一值域内的  $y$ ，存在至多一个定义域内的  $x$  使得  $f(x) = y$ 。

另一种说法为， $f$  为单射，当  $f(a) = f(b)$ ，则  $a = b$  (若  $a \neq b$ ，则  $f(a) \neq f(b)$ )，其中  $a$ 、

b 属于定义域。