

# GCN相关的基础文献

图卷积网络是针对于图结构数据的有力分析工具，这几篇文章是图卷积网络领域内比较重要的工作，对于我们理解和使用图卷积网络有着很重要的意义

## 目录

- 自己的理解
- 谱视角(Spectral method)
  - before deep learning graph neural network--图卷积
  - convolutional neural networks on graphs with fast localized spectral filtering--图卷积的快速计算
  - Semi-supervised classification with graph convolutional network-iclr2017--一阶近似与半监督学习
  - Deeper insights into graph convolutional networks for semi-supervised learning aaai2018
  - Diffusion convolutional recurrent neural network Data-driven traffic forecasting--扩散卷积
  - Adaptive Graph Convolutional Neural Networks AAAI2018--根据输入动态变化的拉普拉斯矩阵
  - Fastgcn fast learning with graph convolutional networks via importance sampling iclr2018--采样与推断式的学习
- 空间视角(spatial method)
  - Geometric deep learning on graphs and manifolds using mixture model CNNs CVPR2017--一般化的框架
  - Inductive Representation Learning on Large Graphs--推断式的图卷积
  - Structure-Aware Convolutional Neural Networks NIPS2018--具备局部拓扑结构感知能力的图卷积
  - Graph attention networks ICLR2018--注意力机制的引入
  - Gaan Gated attention networks for learning on large and spatiotemporal graphs--注意力机制的进一步延伸

## 自己的理解

图神经网络是一种提取图结构数据特征的特征提取器。

图结构的数据相较于欧式空间内的数据，具备着更加强的先验，在欧式空间中每一个位置的数据局部的拓扑都是相同的，但是在图中，每一个节点的局部拓扑都不必然相同，而不同的局部拓扑也就蕴含着更加丰富的信息，所以图神经网络最本质的想法就是吧蕴含着图结构信息的特征信息从数据中提取出来，以供更进一步的处理。

而不同的图神经网络形式的关键点就在于采取什么样的形式将图结构的信息融入到数据中去，在这样

的思想之下我们分析一下以上文章中所做出的尝试：

首先是最早期的基于谱分析的方法，就是对欧式空间的卷积进行类比，卷积操作本质上是傅里叶变换之后在频域的乘积，在图结构中不存在相同的傅里叶变换，因此需要我们找到在图结构中的傅里叶变换形式。傅里叶变换的想法就是通过基函数的线性组合得到目标函数，而基函数选择的标准是其散度算子，也就是拉普拉斯算子的特征函数，所以在图领域我们需要找到对应的散度算子，找到它的特征函数（特征向量），进而就可以进行图领域内的傅里叶变换。而在图领域内所定义的拉普拉斯算子就是拉普拉斯矩阵，这里在数学上有比较严格的推导，我们只需要有一个直观上的理解。欧式空间内的拉普拉斯算子计算得到的是梯度场在每一个节点的聚拢和发散程度，进而言之是描述整体函数变化趋势的一个算子。而经过拉普拉斯矩阵相乘的输入  $X \in \mathbb{R}^N$ ,  $L = D - W \in \mathbb{R}^{N \times N}$ , 可以得到的是一个与节点数同维的向量，每一个位置是： $x'_i = d_i x_i - \sum_{j \in \mathcal{N}_i} w_{ij} x_j$ , 这也是相类似的，每一个节点在周围节点影响下的变化趋势的体现。

拉普拉斯矩阵是根据图的结构所生成的，同样的也是描述了图中信息传递的方式，所以最早期的图卷积网络都是以直接的拉普拉斯矩阵作为融合图结构与数据信息的桥梁，通过增加一些适用于深度学习的变形将图卷积引入到深度学习的殿堂中。

进一步的，有学者探究了在直接拉普拉斯矩阵之外的其他形式，这就是扩散卷积，他更加直接的考虑图中信息传递的方式，将这样的过程与随机游走过程进行类比，从而得到了另外一种形式的图神经网络。

在另一方面就会有人考虑，当先验的图结构并不准确的情况下，我们应该如何依据先验的图结构，再通过数据内涵的信息提取更加准确的拉普拉斯矩阵，这样就得到了Adaptive Graph Convolutional Neural Networks文中的工作。

同样，学者通过探究图卷积网络的本质，发现利用拉普拉斯矩阵的方式进行图卷积是对每一个节点最周围节点进行加权和，也就是某一种形式的拉普拉斯平滑，而深度的卷积网络会导致每一个节点的输入被更远节点的信息所冲淡，这样就限制了原本的依赖于拉普拉斯矩阵的图卷积形式在深度学习中进行应用。

另外，单纯依赖于拉普拉斯矩阵的方式，会导致无法适用于推断式的任务，也就是说无法对于没有出现过的节点，或者没有出现过的图结构下的数据进行有效的特征提取，就需要进行不依赖于拉普拉斯矩阵整体的图神经网络的方法

这样的动机下出现了fast gcn，这是一种基于采样的图神经网络，这种算法依旧脱胎于原始的图卷积网络，只不过通过引入一个特殊的概率空间视角，使得整体网络的训练不再依赖于严格的图拉普拉斯矩阵，也就可以适用于推断式的任务。但是这样的算法不一定可以很好的适用于大规模的稀疏图。

于是，学者们不再固定于谱视角之下，整体宏观的图结构信息，但是从空间视角入手，单纯的观察每一个节点与其周围节点直接的关系，每一个节点沿着它所连接的边，获取邻域节点的信息。但是直接这样的方法会因为局部拓扑结构不一样而导致没有办法生成适用于图中所有节点的统一网络结构，所以采取了先采样，然后进行pool的方式来保证网络中数据流的一致，这样就有了GraphSAGE的工作；另外的学者，希望得到能够对局部拓扑结构敏感的卷积方式，这样就可以适用于具备不同的局部拓扑的节点，这样就引入了另外一种形式的图卷积。

同时，也有学者抛开卷积这个固有的视角，将注意力机制引入到图神经网络中，通过多头的注意力机制来权衡每一个节点位置的输出对周围节点影响的程度，这样就有GAT的工作，这也是我文章阅读之中最为有效的一类方法；更进一步的通过门控制的方式来确定不同的注意力头信息的重要性也取得了更好的效果。

# 谱视角

## 图卷积

在Euclidean domains(图像，声音信号等类型)的数据中，卷积操作就是将函数进行傅里叶变换映射到频域空间，之后进行相乘再做逆傅里叶变换得到的结果。对于图结构的数据，如果我们想要将卷积领域进行扩展，就需要合理的定义在图领域的傅里叶变换，进而可以定义图领域的卷积操作。

把Euclidean domains中的傅里叶变换迁移拓展到图领域中，最核心的步骤在于把拉普拉斯算子的特征函数 $e^{i\omega t}$ 在图领域中做出对应的映射，而这个对应的映射就是图拉普拉斯矩阵的特征向量

- 传统领域的傅里叶变换

传统的傅里叶变换定义为： $F(\omega) = \mathcal{F}[f(t)] = \int f(t)e^{i\omega t} dt$ ，这是信号 $f(t)$ 与基函数 $e^{i\omega t} dt$ 之间的积分，而基函数选择它的原因在于 $e^{i\omega t} dt$ 是拉普拉斯算子的特征函数

同样的，当我们想将卷积拓展到图领域时，因为图的拉普拉斯矩阵就是离散的拉普拉斯算子，所对应的选择的基函数就应当是图拉普拉斯矩阵的特征向量

- 图拉普拉斯矩阵的定义与分解

图的拉普拉斯矩阵通常定义为 $L = D - A$ ，其中 $D$ 指顶点度数组成的对角矩阵， $A$ 指邻接矩阵或者边权重矩阵；在运算中通常采用归一化后的拉普拉斯矩阵 $L^{sys} = D^{-1/2} L D^{-1/2}$

图拉普拉斯矩阵是对称半正定矩阵，它的特征向量之间相互正交，所有的特征向量构成的矩阵成为正交矩阵，因此我们可以知道拉普拉斯矩阵一定可以进行谱分解，并且分解后有特殊的形式：

$$L = U \begin{bmatrix} \lambda_1 & & \\ & \dots & \\ & & \lambda_n \end{bmatrix} U^T$$

其中 $U$ 是由拉普拉斯矩阵特征向量组成的矩阵，而 $\lambda$ 代表着拉普拉斯矩阵的特征值

- 图领域的傅里叶变换

仿照传统领域下的傅里叶变换定义，我们就可以得到图领域的傅里叶变换

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^N f(i) u_l^*(i)$$

$f$ 是图上的N维向量， $f(i)$ 表示节点*i*上对应的输入， $u_l(i)$ 表示第*l*个特征向量的第*i*个分量，特征值就对应了在不同基函数下对应的分量，也可以在一定程度上认为是对应的频率。 $f$ 的图傅里叶变换就是与 $\lambda_l$ 对应的特征向量 $u_l$ 进行内积运算

进一步的，我们图傅里叶变换的矩阵形式写成：

$$\begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix} = \begin{bmatrix} u_1(1) & u_1(2) & \cdots & u_1(N) \\ u_2(2) & u_2(2) & \cdots & u_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(N) & u_2(N) & \cdots & u_N(N) \end{bmatrix} \begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix}$$

也就是说  $\hat{f} = U^T f$ , 逆傅里叶变换也可同样的形式推广:  $f = U \hat{f}$

- 图卷积

对于卷积核  $h$ , 我们将其进行傅里叶变换之后的结果写成对角矩阵的形式就是:  $g(\Lambda) = diag(\hat{h}(\lambda_1), \hat{h}(\lambda_2), \dots, \hat{h}(\lambda_N))$ , 所以  $h$  与  $f$  的卷积就可以写成:

$$f *_g h = U g(\Lambda) U^T f$$

## 图卷积的快速计算

回顾传统二维图像中的卷积, 我们可以发现二维的卷积具备着两个很好的特性:

- 局部连接: 每一个卷积核在不同的位置只对这一个位置的局部具备感受的能力, 不接收其他区域的信息
- 参数共享: 每一个卷积核在不同的位置都使用相同的参数, 这就极大的减少了所需学习的参数数量

但是在上述图卷积的操作中这两点都不具备, 每一个卷积核所需要学习的参数  $g(\Lambda) = diag(\hat{h}(\lambda_1), \hat{h}(\lambda_2), \dots, \hat{h}(\lambda_N))$  的规模为  $\mathcal{O}(N)$ ,  $N$  表示节点数目, 在多通道, 多卷积核的情况下参数的数目相当的庞大; 另外在这样的操作下就意味着每一个节点都可以看到所有的节点的信息, 这样就不具备局部连接的特性

因此, 为了适应深度学习的需求, 学者们对图卷积做了一定程度的改变使得它具备了局部感知特性并且降低了参数数量:

### Polynomial parametrization for localized filters

多项式参数化就是将原本作为卷积核参数的对角矩阵, 由简单的学习对角矩阵对角线上每一个元素改变为学习一个多项式的系数, 即:

$$g(\Lambda) = diag(\theta_1, \theta_2, \dots, \theta_N) \rightarrow g(\Lambda) = \sum_i^{K-1} \theta_i \Lambda^k$$

这样原本的卷积操作就变成:

$$\begin{aligned}
f *_g h &= U g(\Lambda) U^T f \\
&= U \left( \sum_i^{K-1} \theta_i \Lambda^k \right) U^T f \\
&= \left( \sum_i^{K-1} \theta_i L^k \right) f
\end{aligned}$$

这样就将原本的参数量从  $\mathcal{O}(N)$  降低到了  $\mathcal{O}(K)$ ，同时由于拉普拉斯矩阵的特殊性， $K$  具备着明确的含义，即每一个节点所能看到的节点的最近距离，这样就达到了降低参数量同时使得卷积核具备了局部连接性的目的

## Recursive formulation for fast filtering

在进行了上述的变化之后，虽然降低了整体的参数量也避免了拉普拉斯矩阵的特征值分解计算，但在实际的计算中计算代价仍然不小（拉普拉斯矩阵的分解步骤可以在训练前就分解完成，整体更新参数时只要调用分解完成的特征向量矩阵即可，而同时因为需要计算拉普拉斯矩阵的乘积，空间存储本身的复杂度就在  $\mathcal{O}(N^2)$  级别，所以整体避免分解只是略微降低了空间复杂度）

因此，为了避免  $\mathcal{O}(N^3)$  的矩阵乘（虽然在算法层面有更快级别的算法），作者提出采用切比雪夫多项式逼近的方法来通过递归计算拉普拉斯矩阵乘向量的方式降低计算的复杂度，即：

$$g(\Lambda) = \sum_i^{K-1} \theta_i \Lambda^i \approx \sum_i^{K-1} \theta_i T_i(\tilde{L})$$

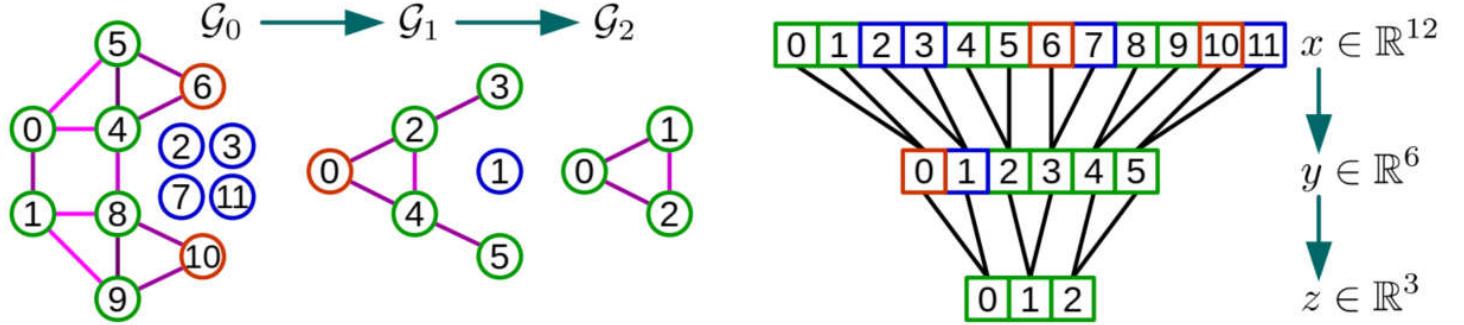
其中  $\tilde{L} = 2L/\lambda_{max} - I_N$  这是为了保证切比雪夫多项式的数学性质而做出的变换，切比雪夫多项式的计算公式为： $T_k(L) = 2xT_{k-1}(x) - T_{k-1}(x)$ ,  $T_1(x) = x$ ,  $T_0(x) = 1$ ，这样，如果我们将  $T_i(\tilde{L})f$  记做  $x_k$  的话，卷积操作就变成  $f *_g h = \sum_i^{K-1} x_i$ ，而每一个  $x_i$  可以通过  $x_i = 2\tilde{L}x_{i-1} - x_{i-2}$  的方式递归计算得到，而由于拉普拉斯矩阵是一个稀疏矩阵，那么整体的计算复杂度就降低到  $\mathcal{O}(\mathcal{E})$ ， $\mathcal{E}$  表示边的数量，这样就大大的降低的计算的复杂度，使得图卷积操作适应了深度学习的要求

## Fast Pooling of Graph Signals

进一步地，在图卷积中我们可以构建一个类似于池化的操作，把相近似的节点相互聚合起来，这样的操作需要满足两点：1) 我们需要可以控制，或者明确知道每一轮类比的池化操作之后图会缩减成什么尺寸；2) 类比的池化操作需要可以快速的实现，最好可以容易的并行化计算，这样就可以进行加速。首先我们需要介绍一下 Graclus multilevel clustering 算法，这是一个基于贪心规则的算法，具体的步骤是，在未标记的点中选中一个，然后在它未被标记的邻接节点中选取可以最大化  $W_{ij}(\frac{1}{d_i} + \frac{1}{d_j})$  的节点，将这两个节点合并成一个，把他们的与其他节点的边权重加和作为新节点的边权重，重复直至所有的节点都探索，这样整体上图的大小会被缩小一半（会有一些单独节点 singleton 的存在）

在实际的计算中，为了便于并行化与快速的计算，整体上通过平衡二叉树的方式将节点之间在不同层级的合并关系组织起来，具体来说就是：1) 每一层的节点都是前一层合并而来的两个的父节点；2) 单子节点 (singleton) 将会配备一个虚拟节点与它一起作为下一层父节点的兄弟节点，每一个虚拟节点都

设置成为中性值(neutral value), 也就是0, 这样以ReLU函数作为激活函数同时通过max pooling的方式就不会因为添加了虚拟节点而造成影响:



## 一阶近似与半监督学习

有了前面的基础之后, 学者们进一步提出了更加简化的图卷积框架, 在上述的图卷积操作 $g(\Lambda)x = \sum_i^{K-1} \theta_i T_i(\tilde{L})x$ 中, 如果我们将 $K$ 的值限定为2, 在实际卷积中就意味着每个节点只能看到与自己邻接的节点信息, 图卷积的操作就会变为:

$$g(\Lambda)x = \theta_0 x + \theta_1 \tilde{L}x$$

因为神经网络中对数据的尺度变换和归一化操作, 我们可以进一步假设 $\lambda_{max} \approx 2$ , 再加上这里我们使用的是拉普拉斯矩阵的归一化形式: $L = I_n - D^{-1/2}AD^{-1/2}$ 那么我们可以将上式进一步简化为:

$$g(\Lambda)x = \theta_0 x + \theta_1 (L - I_n)x = \theta_0 x - D^{-1/2}AD^{-1/2}\theta_1 x$$

再进一步地, 为了简化参数这里指定 $\theta_0 = -\theta_1 = \theta$ 上式就简化为:

$$g(\Lambda)x = \theta(I_n + D^{-1/2}AD^{-1/2})x$$

随后文章进行了进一步的化简, 为了方便下一节讲述deeper insights, 这里就不再进一步化简, 详细的化简参见文章内容

有了这样的图卷积操作, 就可以进行快速的半监督学习, 具体的方法就是当网络最后的输出之后, 只对有标记的节点计算交叉熵损失, 然后进行梯度回传更新网络参数, 输出时预测所有节点类别

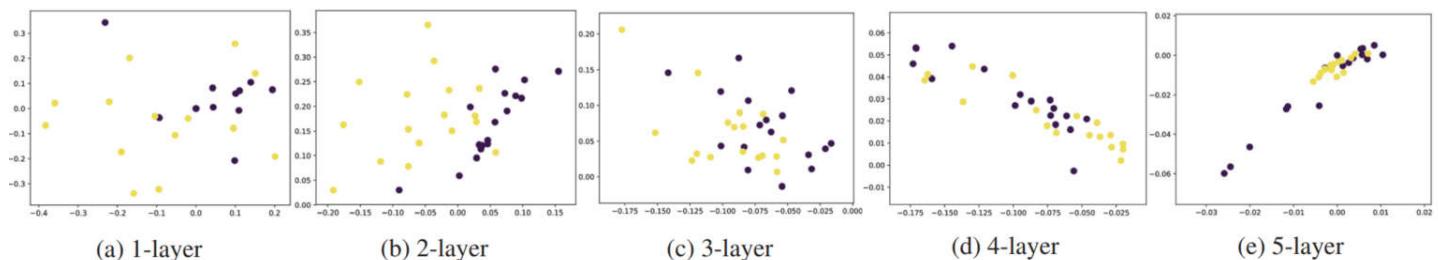
## deeper-insights

为什么GCN可以取得很好的效果呢? 当我们卷积操作 $g(\Lambda)x = \theta(I_n + D^{-1/2}AD^{-1/2})x$ 聚焦在单一的某个节点时, 就会得到:

$$h_i^r = \theta(h_i^{r-1} + \sum_{j \in \mathcal{N}(i)} \frac{h_j^{r-1}}{\sqrt{d_i d_j}})$$

这就是另外一种形式的拉普拉斯平滑，这样的平滑本质上的含义在于，每一个节点与自己相邻近节点相近似，每一层的图卷积会让每一个节点与自己相近的节点（也更大可能上是处于同一类别的节点）交换信息，这也就是为什么图卷积网络在半监督学习中会取得较好结果的原因

但是图卷积网络与传统卷积网络不同的一点在于，并不是越深层的卷积层数就会带来更好的效果，事实上，在半监督分类问题中，层数过多会降低整体网络效果：



这源自于拉普拉斯平滑本身的特性，快速的进行连续几层的拉普拉斯平滑会导致图中联通区域内的节点的值趋于一致。

而这样的特性本身限制了图卷积网络的效果，深层的网络会导致过度平滑而使得连通区域趋于一致，但浅层的网络并不能使节点充分利用到整个图的信息，在这篇文章中作者提出了一种联合训练的方式来避免在深层图卷积网络中带来的过度平滑的问题，同时又可以通过深层的网络将信息传播到整个图中。

文章的思路是只用浅层的图卷积（一般是两层），然后通过自训练与随机游走过程的联合训练的方式增强在半监督学习中的效果。

### 随机游走模型：

根据选定的随机游走模型来确定，确定下来随机游走的分布矩阵 $\mathcal{P}$ ，其中 $\mathcal{P}_{ij}$ 表示从第*i*个节点随机游走到第*j*节点终止的分布概率，这样我们就可以根据这个随机游走分布矩阵 $\mathcal{P}$ ，针对每一个类别*k*，可以计算一个置信度向量

$$\mathbf{p} = \sum_{j \in \mathcal{S}_k} \mathcal{P}_{:,j}$$

其中 $\mathbf{p}_i$ 是第*i*个节点属于第*k*类的置信度，在这个向量中选取前*k*项，作为半监督学习中补充的被标记数据，作为扩大的数据集

### 自训练过程

自训练过程是指在卷积训练中，会根据图卷积网络的输出结果，计算所有类别中前*k*项置信度最高的作为补充被标记数据集

通过取这两个过程的并集，模型在半监督学习分类问题中取得了最好的效果

## 扩散卷积

扩散卷积(Diffusion Convolutional)是根据随机游走过程而实现的一种图卷积的形式，这种形式的图卷积就是将图中信息传递的过程与随机游走过程类比，随机游走过程就是我们假设从一个节点*i*出发，以

一定规律随机选取周围的节点进行游走，同时以一定概率 $\alpha$ 终止，如果下一步节点选取的方式是均匀随机选取，那么就会有得到一个随机游走最终结果的分布矩阵：

$$\mathcal{P} = \sum_{i=0}^{\infty} \alpha(1 - \alpha)^i (D_O^{-1} W)^i$$

其中 $D_o$ 是节点的出度组成的对角矩阵，仿照这个过程就可以得到扩散卷积的计算方式：

$$\mathbf{X}_{:,p} *_{\mathcal{G}} f_{\theta} = \sum_{k=0}^{K-1} (\theta_{k,1}(D_o^{-1} W)^k + \theta_{k,2}(D_i^{-1} W)^k) \mathbf{X}_{:,p}$$

这样的图卷积替代传统的图卷积在GCRNN中的作用会得到更好的效果

## 根据输入动态变化的拉普拉斯矩阵

在传统的谱视角下的卷积中，图的拉普拉斯矩阵是给定的，但是由于往往这种拉普拉斯矩阵是根据物理世界的设定(例如分子之间的化学键)或者手工指定的，这就导致有可能两个相连节点之间的关系可能并没有不相连节点之间的关系更大，因此本文提出了一种根据数据动态学习拉普拉斯矩阵的算法，同时由于拉普拉斯矩阵本身是根据特征变换和设定的距离矩阵计度量方式计算出来，只需要要求输入的样本具有相同的输入特征维数即可，不要求具备相同的拓扑结构，因此是一种可以适应不同图结构的方法

这篇文章中整体的结构仍然是传统的图卷积的方式，但是增加了一个SGC-LL(Spectral Graph Convolution layer with Laplacian Learning)层，这一层的构建需要有以下几个关键点：

### Training Metric for Graph Update

在图结构的数据中，有时不相连的节点之间往往具备更加关键的相关程度，因此节点之间距离度量就需要根据任务和训练的过程而动态的变化，同时据此生成变化的拉普拉斯矩阵。不同的任务下度量标准的生成目标也不痛：非监督学习任务下最优的度量标准是让类间距离最大，类内距离最小；而监督学习之下任务就是让损失函数最小化。这里给出一个一般化的Mahalanobis distance定义：

$$\mathbb{D}(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$$

通过 $M = W_d W_d^T$ 限定 $M$ 是对称半正定矩阵，其中 $W_d \in \mathbb{R}^{d \times d}$ 是一组空间变换的基，这里需要注意的是，如果只通过矩阵 $M$ 进行变换，这只不过是将欧式空间内的距离进行空间变换得到的结果，如果要处理更加复杂的空间距离分配，就需要进行更加高维的变换，文中在这里采用了高斯核变换的方式：

$$\mathbb{G}_{x_i, x_j} = \exp(-\mathbb{D}(x_i, x_j)/(2\sigma^2))$$

通过上面的公式就可以得到一个稠密图的邻接矩阵  $\tilde{A}$ ,  $W_d$  是需要学习的参数

### Re-parameterization on feature transform

这篇文章中对这一项的解释有一些奇怪，我的理解上如下的变换形式导致的是输出层中不同的通道由原来的均匀的从输入通道之间加和，改变成为通过不同的权重进行取舍，这样不会要求很多的参数同时也增加了模型的复杂程度：

$$Y = (U g_\theta(\Lambda) U^T X) W + b$$

### Residual Graph Laplacian

在度量标准的衡量之中，如果我们完全以随机初始的  $M$  计算拉普拉斯矩阵，会导致整体的收敛过程较难，所以为了加速训练过程，增强训练稳定程度，文中假设动态的拉普拉斯矩阵是在给定的固有的拉普拉斯矩阵  $L$  之上的扰动：

$$\hat{L} = L + \alpha L_{res}$$

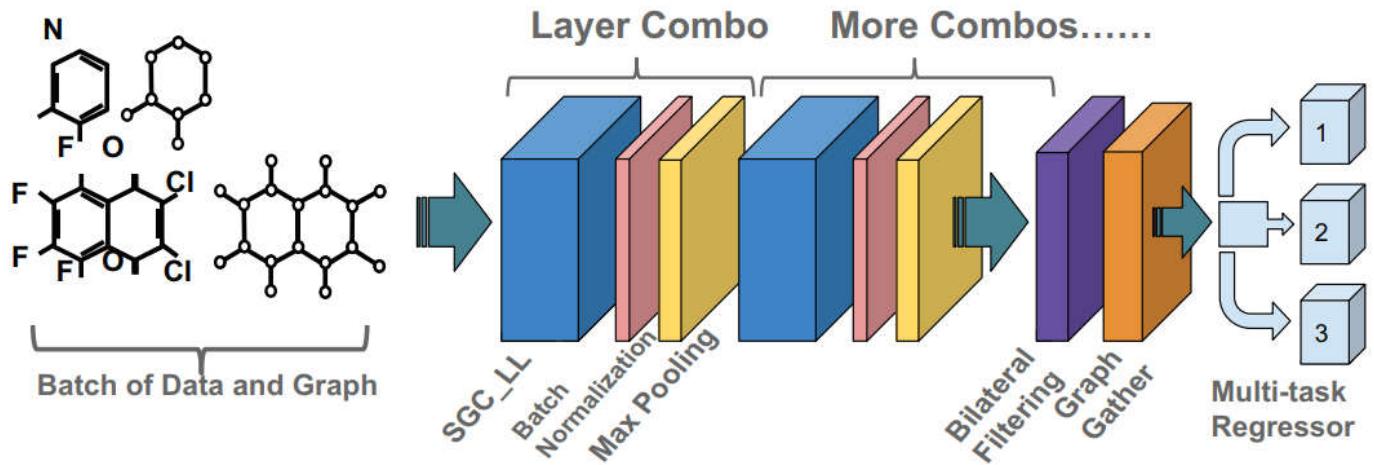
因此我们只需要通过学习得到拉普拉斯矩阵的残差即可。有了SCG-LL再通过增加一些双边滤波器的方式就可以得到完整的网络结构，这部分细节可以参考原文

以下是SCG-LL层的计算流程以及整体上AGCN的网络结构

## Algorithm 1 SGC-LL Layer

**Data**  $\mathbf{X} = \{X_i\}$ ,  $\mathbf{L} = \{L_i\}$ , **Parameter**  $\alpha, M, W, b$

- 1: **for**  $i$ -th graph sample  $X_i$  in mini-batch **do**
- 2:      $\tilde{A}_i \leftarrow Eq.(6), Eq.(7)$
- 3:      $L_{res}(i) \leftarrow I - \tilde{D}_i^{-1/2} \tilde{A}_i \tilde{D}_i^{-1/2}$    ▷  $\tilde{D}_i = diag(\tilde{A}_i)$
- 4:      $\tilde{L}_i = L_i + \alpha L_{res}(i)$
- 5:      $Y_i \leftarrow Eq.(8)$
- 6: **return**  $\mathbf{Y} = \{Y_i\}$



## 采样与推断式的学习

在传统的谱视角下的图卷积，包括上面讲述的可以动态计算拉普拉斯矩阵的图卷积算法，都是基于一个认识：不同节点之间的数据具备着在这个图结构下的相近邻关系，只要我们有着在这个对数据节点之间准确相连关系，就可以通过一定程度上的操作得到在这个图结构下数据的特征。

而本文将提供了一种不同的视角，本文将图卷积视作一个概率空间的概率测度之下特征提取函数的积分变换，这样的视角下就可以通过蒙特卡洛采样的方式对实际的损失进行估计，这样就避免了传统图卷积中对稠密图计算代价过大和无法适用于推断式的任务的问题。

### 通过采样训练与推理

首先我们来看，在机器学习中SGD的应用场景下，我们通常是选取使得模型整体预期的损失最小的参数，作为整体模型的参数，也就是说最小化：

$$L = E_{x \sim D}[g(W; x)]$$

在实际中是通过最小化  $\frac{1}{n} \sum_i g(W; x_i)$  的方式，来对实际的期望进行拟合，由于在传统的问题之下，每一个样本之间可以看做是独立同分布的采样，每一个样本点之间不具备依赖关系，而在图领域下，每一个节点之间具备着显著的相关性，因此需要寻找一种额外的形式才能通过采样对损失误差进行估计。

而文中提出的方法就是从图概率空间的角度入手，假设存在一个无限的图  $G'$ ，和它相对应的节点集合  $V'$ ，存在一个概率空间  $(V', F, P)$ ，对于每一个给定的图  $G$  都是  $G'$  的诱导子图，它的节点集合  $V_d$  都是  $V'$  的一个采样，对于这个样本空间， $V'$  就是采样空间，而  $F$  是所有可能的事件集合，也就是所有  $V$  的集合，而  $P$  则取决于采样的分布

前面说过在传统的机器学习中不同的样本之间是独立的，所以可以通过采样的方式估计误差。但是在图结构的数据下，每一个节点都与周围节点之间紧密联系，为了应对这个问题，在引入前面所述的概率空间的概念之后，文中把每一层图卷积看做是基于一些随机变量的 embedding function，这些函数的变量就是随机采样得到的节点前一层的输出特征：

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

这样，就可以把卷积操作看做是一种积分变换：

$$\tilde{h}^{(l+1)}(v) = \int \hat{A}(v, u) h^{(l)}(u) W^{(l)} dP(u), \quad h^{(l+1)}(v) = \sigma(\tilde{h}^{(l+1)}(v))$$

$$L = E_{v \sim P}[g(h^{(M)}(v))] = \int g(h^{(M)}(v)) dP(v)$$

$M$ 是模型的层数， $h^{(M)}(v)$ 是模型最后一层 $v$ 的输出。这个积分变换之下， $u$ 和 $v$ 都是服从于 $P$ 的独立同分布的随机变量，也就是说 $u$ 和 $v$ 是根据 $P$ 随机采样得到的节点，核函数 $\hat{A}(u, v)$ 是正规化的邻接矩阵 $u, v$ 对应位置的值。

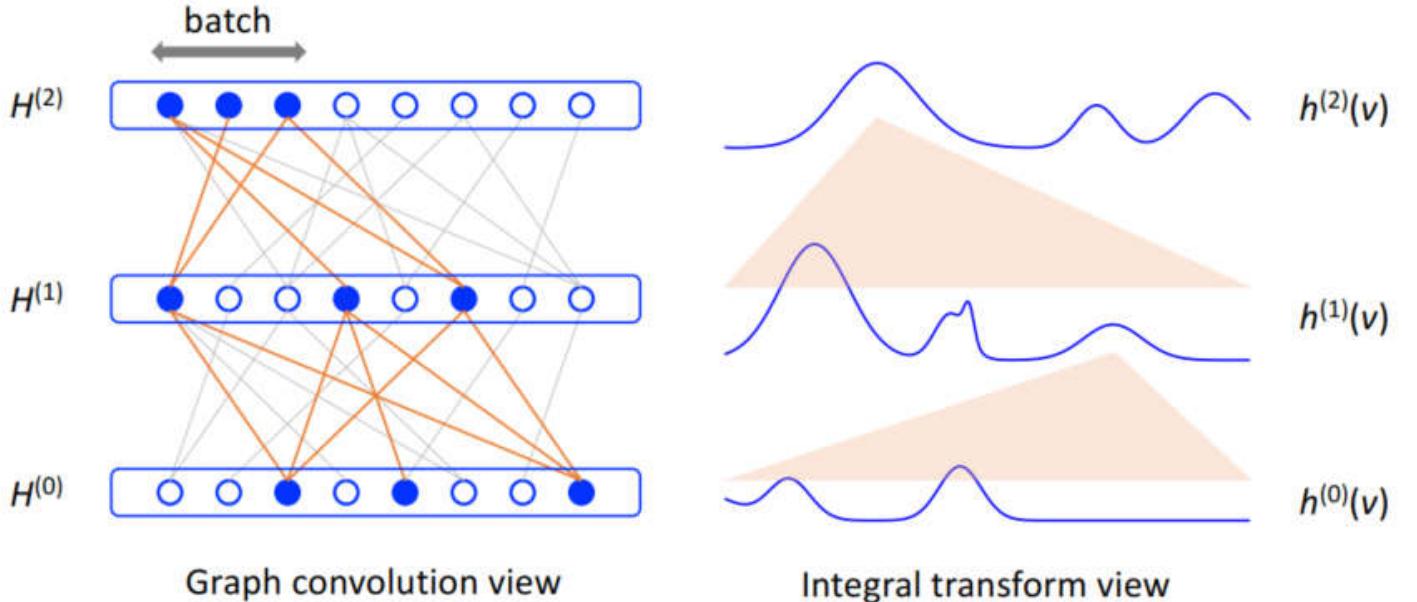
在这样的视角下，我们把每一层卷积看做了一个函数的积分变换，变换的核函数就是拉普拉斯矩阵与相应的权重参数 $W^{(l)}$ ，这样我们就可以通过蒙特卡洛采样的方式对这个积分变换的参数值进行具体的估计：

每一层 $l$ 我们进行 $t_l$ 次独立同分布的采样： $u_1^{(l)}, u_2^{(l)}, \dots, u_{t_l}^{(l)} \sim P$ 对这个积分变换进行估计：

$$\tilde{h}_{t_l+1}^{(l+1)}(v) := \frac{1}{t_l} \sum_{j=1}^{t_l} \hat{A}(v, u_j) h_{t_l}^{(l)}(u_j^{(l)}) W^{(l)}, \quad h_{t_l+1}^{(l+1)}(v) = \sigma(\tilde{h}_{t_l+1}^{(l+1)}(v))$$

$$L_{t_0, t_1, \dots, t_M} := \frac{1}{t_M} \sum_{i=0}^{t_M} g(h_{t_M}^{(M)}(u_i^{(M)}))$$

在实际中，每一个给定的图都假定是由超图 $G'$ 采样得到的，因此就需要进行bootstrap的方式进行一致性的估计整体的积分变换中的参数



文中还通过推导的方式确定下来最优的采样分布 $P$ 是这样的：

$$q(u) = \frac{\|\hat{A}(:, u)\|^2}{\sum_{u' \in V} \|\hat{A}(:, u')\|^2}, u \in V$$

这里我还没有做实验，但是从描述来看感觉只能对稠密图或者规模较小的图，具备比较好的效果，或者在采样的过程中需要对batch\_size做出比较精心的设计，否则每一层之间的采样如果都是不相关的数据，就很难学习到有益的信息

## 空间视角

以上的内容都是来自于从谱视角进行的对图特征的提取，这种方法下的图卷积有着严密的数学推导与很好的实验效果，但是一个严峻的问题就在于这样的图卷积操作依赖于图的拉普拉斯矩阵，当图的结构发生变化时往往不能进行很好的泛化，就如下图中展示的：

Dataset	LeNet5 [23]	ChebNet [13]	MoNet
*Full grid	99.33%	99.14%	99.19%
* $\frac{1}{4}$ grid	98.59%	97.70%	98.16%
300 Superpixels	-	88.05%	<b>97.30%</b>
150 Superpixels	-	80.94%	<b>96.75%</b>
75 Superpixels	-	75.62%	<b>91.11%</b>

传统的图卷积(LeNet)与在固定的图结构(前两行)下学习得到的图卷积网络(ChebNet)都取得了很好的效果，但是当数据变成从超像素中提取的时候，因为这是不同图片之间根据超像素算法所提取到的图结构并不相同，就导致了图的结构发生了变化，这是传统的图卷积的准确率产生了严重的下滑，最多下降到75%左右，因此一种不依赖于拉普拉矩阵的图卷积算法需要被探索和研究，这就是空间视角下的图卷积方式

## 一般化的框架

在这篇CVPR2017年的文章中，提出了一般化的可以用于泛化卷积操作的框架，在这个框架之下可以通过设置不同的伪坐标(pseudo-coordinate)形式，与权重计算函数就可以实现对不同形式的卷积。在这个框架之下，可以实现之前各种对于图的卷积与对于流形的卷积

我们将需要进行卷积的输入（图像，音频，图结构的数据，流形等）上的一个节点记做 $x$ ，在不同的输入情况下，我们都考虑节点 $y \in \mathcal{N}(x)$ ，其中 $\mathcal{N}(x)$ 表示 $x$ 的邻域节点，包括 $x$ 本身。对于每一个 $y$ 都可以专门设计出一种伪坐标形式 $\mathbf{u}(x, y)$ ，针对不同的伪坐标值，设计出权值计算函数（也可以叫卷积

核) :  $\mathbf{w}_\Theta(\mathbf{u}) = (w_1(\mathbf{u}), \dots, w_J(\mathbf{u}))$ ), 权值函数由一些可学习的参数 $\Theta$ 确定, 每一个局部的卷积操作表示为:

$$D_j(x)f = \sum_{j \in \mathcal{N}(x)} w_j(\mathbf{u}(x, y))f(y), \quad j = 1, \dots, J$$

那么整体上在非欧几里得空间下的卷积形式就可以写作:

$$(f * g)(x) = \sum_{j=1}^J g_j D_j(x)f$$

在这个框架设计中, 最关键的步骤是伪坐标的确定与权值函数的计算, 以下是一些卷积形式在本文框架之下的具体设定:

Method	Pseudo-coordinates	$\mathbf{u}(x, y)$	Weight function $w_j(\mathbf{u}), j = 1, \dots, J$
CNN [23]	Local Euclidean	$\mathbf{x}(x, y) = \mathbf{x}(y) - \mathbf{x}(x)$	$\delta(\mathbf{u} - \bar{\mathbf{u}}_j)$
GCNN [26]	Local polar geodesic	$\rho(x, y), \theta(x, y)$	$\exp(-\frac{1}{2}(\mathbf{u} - \bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u} - \bar{\mathbf{u}}_j))$
ACNN [7]	Local polar geodesic	$\rho(x, y), \theta(x, y)$	$\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} (\bar{\alpha}_1) \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$
GCN [21]	Vertex degree	$\deg(x), \deg(y)$	$\left(1 -  1 - \frac{1}{\sqrt{u_1}} \right) \left(1 -  1 - \frac{1}{\sqrt{u_2}} \right)$
DCNN [3]	Transition probability in $r$ hops	$p^0(x, y), \dots, p^{r-1}(x, y)$	$\text{id}(u_j)$

上式中GCNN与ACNN是应用于流形之中的卷积形式, GCN是前面描述的基于谱视角下的卷积, DCNN是扩散卷积

在这篇文章中, 在总结出整体一般化的框架之后, 提出了自己所实现的卷积操作, 这种卷积操作并没有采用人工设置固定的权重计算函数, 而是选择了一种可以通过训练而不断更新的权值计算函数:

$$w_j(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u} - \mu_j)^\top \Sigma^{-1}(\mathbf{u} - \mu_j)\right)$$

上式中 $\mu_j, \Sigma$ 是需要学习的参数, 其中为了降低学习的难度,  $\Sigma$ 限制为对角矩阵, 这样的情况下就已经足够提供相当的函数复杂度以达到更好的模型效果。如果要追求更加输入的更高维表示, 可以通过在坐标值输入网络之前进行相应的非线性变换达到。在不同的应用中采取不同的坐标形式, 搭配上式表示的权值计算函数就可以得到比以往模型更好的效果。

当我们仔细观察这个权值计算函数, 就会发现这是多维高斯分布的一种形式, 前述的非欧空间内的卷积就可以解释为一种高斯混合模型, 通过学习一种能够描述节点周围相关性的高斯混合模型来合理的分配对周围节点信息的接纳程度。

## 推断式的图卷积

谱视角下, 图的结构是固定的, 对于图中没有出现过的节点, 甚至是全新的图时, 原有训练得到的卷积核参数往往不能很好的适应图的变化, 因此这篇文章提出了一种可以进行推断式任务的图卷积模型: GraphSAGE(Graph SAmple aggerGatE)

这种模型不同于之前图卷积方面的方法，直接通过学习的方式得到每一个节点在下一层的隐层状态向量，而是学习不同的AGGREGATE函数用来收集不同跳步数或者搜索深度的信息，在测试阶段直接应用这些训练好的AGGREGATE函数对没有出现过的节点应用，文中提出了非监督的学习方式，但本文的模型同样也能适用于监督学习的训练方式

## 算法流程

首先我们看一下整体算法的流程：

---

### 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  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$ ; 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  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 

```

---

这个算法的具体含义就是，对不同搜索深度的邻域节点，采用不同的AGGREGATE函数进行信息收集，将从周围节点收集到的信息，通过拼接的方式与之前深度的信息进行融合  
并行化进行时，并行的执行算法框架中内循环内容。同时为了保障不同节点之间因为临近节点数不一样而导致的计算轨迹差异，每一轮邻域的生成都采用了随机采样固定大小集合的方式

## 损失函数

在完全非监督学习的情形下，采用以下损失函数对网络整体参数进行更新：

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^T \mathbf{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^T \mathbf{z}_{v_n}))$$

上式中  $\mathbf{z}_u$  表示算法框架中节点  $u$  这一层的输出， $v$  表示与  $u$  节点通过一个固定步数的随机游走相连接的节点， $P_n$  表示负采样的分布状况， $Q$  是负采样的样本数目。这样的损失函数是在要求相关的节点之间尽量相似，不相关的节点尽量有差异。在不同的任务下，这个损失函数可以根据实际需要进行改变，GraphSAGE 模型也同样可以适用于监督学习的问题

## Aggregator Architecture

算法框架中的AGGREGATE需要是对称的，也就是说不会因为采样的顺序而导致最终的输出结果不同，也要可训练，也就是说需要可以回传梯度，另外还需要具备较好的特征表示能力，在文章中主要实验了以下三种不同的AGGREGATE：

- Mean aggregator

这一部分就是通过对采样得到的隐层状态集合 $\{h_u, u \in \mathcal{N}(v)\}$ , 中按元素进行求均值操作得到

- LSTM aggregator

LSTM比简单的均值操作有更加复杂的函数表现能力, 但是由于它本身不是对称的, 本文只是简单的将输入置换成领域节点的随机采样

- Pooling aggregator

最后一种aggregator是先将输入整体经过一个全连接网络, 然后再进行按元素位置的pool操作, 也就是说:

$$AGGREGATE_k^{pool} = \max(\{\sigma(\mathbf{W}_{pool} h_{u_i}^k + \mathbf{b}), \forall u_i \in \mathcal{N}(v)\})$$

实验结果显示Pooling aggregator达到较好的效果, LSTM aggregator其次

## 具备局部拓扑结构感知能力的图卷积

在传统的卷积之中的局部结构表示只能应对局部具备相同拓扑结构的数据, 因此为了解决这个弊端, 文中提出了一种更加一般化的表示方式, 可以量化的对局部拓扑结构进行编码, 进一步的提出了可以感知到局部结构的卷积方式

首先, 我们需要对传统的卷积做一些形式上的变化, 对于一个一维卷积操作, 给定一个输入 $\mathbf{x} \in \mathbb{R}^n$ 和一个卷积核 $\mathbf{w} \in \mathbb{R}^{2m+1}$ , 那么在第*i*个位置的输出值就可以写成:

$$\bar{y}_i = \mathbf{w}^T \mathbf{x}_i = \sum_{i-m < j < i+m} w_{j-i+m} x_j, \quad i \in \{1, 2, \dots, n\}$$

在这里 $\mathbf{x}_i = [x_{i-m}, \dots, x_{i+m}]^T$ 是第*i*个节点的局部输入, 对于任何一个可以表示称为单变量函数的卷积核都可以被等价的表示成:

$$\bar{y}_i = \mathbf{f}_R^T \mathbf{x}_i = \sum_{i-m < j < i+m} f(j - i + m) x_j, \quad i \in \{1, 2, \dots, n\}$$

其中 $f(\cdot)$ 可以被称为函数式的卷积核,  $\mathcal{R} = \{j - i + m | i - m < j < i + m\}$ ,  $\mathbf{f}_R = \{f(r) | r \in \mathcal{R}\}$ , 在一般化的结构中, 我们可以将 $\mathcal{R}$ 看做是包含着一个节点与它邻居节点相连关系的一种编码方式, 称其为局部结构表示。

严格意义上, 不管是否是源自于欧式空间的数据, 都可以被表示为图的形式 $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{R})$ , 其中 $\mathcal{V}$ 存储着节点上的数据,  $\mathcal{E}$ 存储节点之间的相连关系, 而节点关系矩阵 $\mathbf{R}$ 表征着图中的结构信息, 具体的说就是对于一个节点 $i \in \mathcal{V}$ , 它的局部表示可以写作:

$$\mathcal{R}_i = \{r_{ji} | e_{ji} \in \mathcal{E}\}, \quad i \in \{1, 2, \dots, n\}$$

上式中 $r_{ji}$ 是矩阵 $\mathbf{R}$ 中对应位置的元素, 这样我们把保存了所有局部信息节点的集合记做 $\mathcal{S} = \{\mathcal{R}_i | i \in \mathcal{V}\}$ , 这样我们可以定义一个可以感知局部结构的卷积:

$$\bar{y}_i = \mathbf{f}_{\mathcal{R}_i}^T \mathbf{x}_i = \sum_{e_{ji} \in \mathcal{E}} f(r_{ji}) x_j, \quad i \in \{1, 2, \dots, n\}$$

这里  $\mathbf{f}_{\mathcal{R}_i} = \{f(r_{ji}) | e_{ji} \in \mathcal{E}\}$ , 根据不同的  $\mathcal{R}_i$  而变化, 这样就可以具备从不同的局拓扑结构收集信息的能力

要将这样的方式应用深度学习的实际应用之中还需要解决两个关键性的问题: 1) 如何应对无穷多可能的局部结构(这里主要是根据  $r_{ji}$  的变化); 2) 如何确定下来关系矩阵  $\mathbf{R}$

### Polynomial parametrization for functional filters

对于参数式的卷积核可以采用利用函数多项式的形式进行逼近的办法: 对于一个任意的单变量函数  $h(x)$ , 都可以通过一系列基函数  $\{h_1(x), h_2(x), \dots\}$  和相对应的参数  $\{v_1, v_2, \dots\}$  来进行拟合:

$$h(x) \approx \sum_{k=1}^t v_k h_k(x)$$

因此, 文中采取了切比雪夫多项式的方式进行逼近, 那么通过截断的切比雪夫多项式就可以实现上面描述的函数式的卷积:

$$\bar{y}_i = \sum_{e_{ji} \in \mathcal{E}} f(r_{ji}) x_j = \sum_{e_{ji} \in \mathcal{E}} \left( \sum_{k=1}^t v_k h_k(r_{ji}) \right) x_j, \quad i \in \{1, 2, \dots, n\}$$

通过这种拟合的方式就可以实现对不同局部连接强度的边的不同权重分配

### Local structure representations learning

接下来就是连接强度矩阵  $\mathbf{R}$  的产生,  $\mathbf{R}$  是通过从数据中学习得到的, 为了保持不通通道之间输入的结构一致性, 每一层卷积中都只学习一个固定的局部结构表示集合  $\mathcal{S} = \{\mathcal{R}_i | i \in \mathcal{V}\}$ 。对于一个多通道的输入值  $\mathbf{x} \in \mathbb{R}^{n \times c}$ , 其中  $n$  表示节点数目  $c$  表示通道数, 局部的结构表示可以通过以下的方式得到:

$$\mathcal{R}_i = \{r_{ji} = T(\bar{x}_j^T \mathbf{M} \mathbf{x}_i) | e_{ji} \in \mathcal{E}\}, \quad i \in \{1, 2, \dots, n\}$$

上式中  $\mathbf{M} \in \mathbb{R}^{c \times c}$  是一个可学习的参数,  $T(\cdot)$  是  $\text{Tanh}$  函数。这样  $\mathbf{R}$  不一定会是一个对称矩阵, 表示这样的方法有对有向结构也具备学习能力

## 注意力机制的引入

这种算法的主要思想是, 通过注意力机制, 将每一层的输出隐层状态表示称为其周围节点这一层输入的某一种加权和, 这样就可以达到以下几个效果: 1) 计算代价小, 可以很容易的进行并行计算。2) 可以应用于不同的局部结构。3) 可以适用于推断式的任务, 具体的做法如下:

我们先给出GAT每一层结构的输入与输出: 每一层有输入  $\mathbf{h} = \{h_1, h_2, \dots, h_N\}, h_i \in \mathbb{R}^F$ , 其中  $N$

是节点数目， $F$ 是输入层的特征数目，每一层的网络结构就是输出一个隐层状态 $\mathbf{h}' = \{h'_1, h'_2, \dots, h'_N\}$ ,  $h'_i \in \mathbb{R}^{F'}$  其中 $F'$

首先是注意力系数的计算：

$$e_{ij} = a(\mathbf{W}h_i, \mathbf{W}h_j)$$

其中 $\mathbf{W} \in \mathbb{R}^{F' \times F}$ ，是可学习的线性变换系数，然后通过一个共享的注意力计算函数： $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ ，在图结构的数据下，只需要计算有边相连的节点之间的注意力系数。为了让不同的节点之间的系数可以相互比较，对每一个节点 $j$ 通过softmax函数进行归一化：

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

在文中具体的实验中， $a$ 是通过将 $\mathbf{W}h_i, \mathbf{W}h_j$ 进行拼接之后经过内积，然后对每一个注意力系数通过leakyReLU函数引入非线性，所以一个单一的注意力头计算方式可以表示成为：

$$\alpha_{ij} = \frac{\exp(LeakyReLU(\mathbf{a}^T [\mathbf{W}h_i || \mathbf{W}h_j]))}{\sum_{k \in \mathcal{N}_i} \exp(LeakyReLU(\mathbf{a}^T [\mathbf{W}h_i || \mathbf{W}h_k]))}$$

其中 $\mathbf{a} \in \mathbb{R}^{2F'}$ 是需要学习的参数，而 $||$ 表示拼接操作，这样每一层的输出就是通过计算每一个节点在周围节点输入的加权和：

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

在实际的应用中发现，引入多头的注意力机制可以带来很大益处：

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

再将多头的注意力机制进行求均操作就得到了最终这一层的输出

## 注意力机制的进一步延伸

在前一篇文章中应用了多头的注意力机制，不同的注意力头提供了同等的信息，这篇文章的算法引入了一个轻量的子网络，用于计算不同注意力头所提供的的信息重要程度。

首先我们引入一个多头注意力机制下的图神经网络：

$$\mathbf{y}_i = FC_{\theta_o} (\mathbf{x}_i \oplus \parallel_{k=1}^K \sum_{j \in \mathcal{N}_i} w_{i,j}^{(k)} FC_{\theta_v^{(k)}}^h (\mathbf{z}_j))$$

$$w_{i,j}^{(k)} = \frac{\exp(\phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_j))}{\sum_{l=1}^{|N_i|} \exp(\phi_w^{(k)}(\mathbf{x}_i, \mathbf{z}_l))}$$

$$\phi_w^{(k)}(\mathbf{x}, \mathbf{z}) = \langle FC_{\theta_{xa}^{(k)}}(\mathbf{x}), FC_{\theta_{za}^{(k)}}(\mathbf{z}) \rangle$$

上式中  $FC_\theta^\alpha(\mathbf{x}) = \alpha(\mathbf{W}\mathbf{x} + \mathbf{b})$  是一个带激活函数的单层神经网络，其中  $\theta = \{\mathbf{W}, \mathbf{b}\}$  是可学习的参数，没有上标的表示没有激活函数的全连接层。 $\langle \cdot, \cdot \rangle$  表示内积运算，其他的符号表示与前面的含义相同。这里与GAT不同的是，计算  $\phi$  是通过全连接之后直接的内积，而GAT是通过拼接之后与一个参数向量内积的形式。

这时，这篇文章引入了更进一步的门控制的机制，来控制多头的注意力机制所提供的信息的重要程度：

$$\mathbf{y}_i = FC_{\theta_o}(\mathbf{x}_i \oplus \|_{k=1}^K g_i^{(k)} \sum_{j \in N_i} w_{i,j}^{(k)} FC_{\theta_v^{(k)}}^h(\mathbf{z}_j))$$

上式中  $g_i^{(k)}$  就是门控制的体现，它是一个关于  $\mathbf{x}_i, \mathbf{z}_{N_i} = \{\mathbf{z}_j | j \in N_i\}$  的函数，具体的计算方法：

$$\mathbf{g}_i = FC_{\theta_g}^\sigma(\mathbf{x}_i \oplus \max_{j \in N_i}(\{FC_{\theta_m}(\mathbf{z}_j)\}) \oplus \frac{\sum_{j \in N_i} \mathbf{z}_j}{|N_i|})$$

上式中  $\mathbf{g}_i = [g_i^{(1)}, \dots, g_i^{(K)}]$  表示每一个注意力头所提供的信息的重要程度，这样就是GaAN模型的一层，整体上取得了更好的效果：

Models / Datasets	PPI	Reddit
GraphSAGE (Hamilton et al. 2017a)	(61.2) <sup>1</sup>	95.4
GAT (Veličković et al. 2018)	97.3 ± 0.2	-
Fast GCN (Chen et al. 2018)	-	93.7
2-Layer FNN	54.07±0.06	73.58±0.09
Avg. pooling	96.85±0.19	95.78±0.07
Max pooling	98.39±0.05	95.62±0.03
Pairwise+sigmoid	98.39±0.05	95.86±0.08
Pairwise+tanh	98.32±0.18	95.80±0.03
Attention-only	98.46±0.09	96.19±0.07
<b>GaAN</b>	<b>98.71±0.02</b>	<b>96.36±0.03</b>