GNN 教程：漫谈谱图理论和GCN的起源

引言

图神经网络的逐层更新公式简单优雅而高效，以GCN为例，节点Embedding是由自身和邻居节点Embedding的聚合之后再进行非线性变换而得到。如此简单优雅的更新规则是怎么推导出来的呢，背后有没有什么理论依据？在GCN的论文中，作者介绍了两种启发GCN逐层线性传播法则的方法，分别是从谱图卷积的角度和Weisfeiler-Lehman算法的角度。本篇博文将详细介绍如何从图拉普拉斯矩阵出发，通过定义图上的傅里叶变换和傅里叶逆变换而定义图上卷积公式，最后推导出优雅的GCN逐层更新公式。至于Weisfeiler-Lehman算法，因为涉及到图神经网络的表示能力的问题，后面我们会出一个专题详细的介绍它。

Spatial Domain 和Spectral Domain

Spatial domain，翻译过来就是空间域，是最直观感受GCN逐层传播算法的域，即：节点v的Embedding是其所有邻居节点Embedding(包括自己的Embedding)的聚合结果。因此在一些文献上spatial domain又被称做"vertex domain"。但是与CNN所应用的图像不同，空间域中图节点邻居的数目不是一定的，而且节点之间没有相对的次序性。这就产生了一个问题，对于不同邻居个数的节点，卷积怎么定义呢？这就引出了spectral domain的概念。spectral domain，即频谱域，借助卷积定理，我们可以通过定义频谱域上的卷积操作来得到空间域图上的卷积操作。

谱图理论主要研究的是图的拉普拉斯(Lapalacian)矩阵的特征值和所对应的特征向量对于图拓扑性质的影响，是对图空间拓扑的数学描述。下面先来介绍什么是拉普拉斯矩阵。

拉普拉斯矩阵

定义

对于**无向图** $\mathcal{G}=(\mathcal{V},\mathcal{E})$，其Laplacian矩阵定义为$L=D-A$，其中$D$是节点的度矩阵(只在对角线上有元素)，$A$是图的邻接矩阵。拉普拉斯矩阵还有几种扩展定义：

- $L^{sys}==D^{-1/2}LD^{-1/2}$ 称为对称正规拉普拉斯矩阵(Symmetric Normalized Laplacian)，论文中一般用的是这种Laplacian的定义。
- $L^{rw}=D^{-1}L$ 称为随机游走正规拉普拉斯矩阵(Random Walk Normalized Laplacian)。

由于$L=D-A$，且无向图中D为对称矩阵，因此拉普拉斯矩阵是对称矩阵，可以进行特征分解(谱分解)，谱分解对于图从空间域到频谱域的变换至关重要，因为我们用来变换的工具傅里叶变换(Fourier Transform)需要利用Laplacian矩阵的特征值和特征向量，通过变换，从**拓扑结构的图**(spatial domain)到**拉普拉斯矩阵**再到**谱图**(spectral domain)这条链路就形成了。下面就先来介绍拉普拉斯矩阵的谱分解。

谱分解

矩阵的**特征分解**，**对角化**，**谱分解**都是同一个概念，是指将矩阵分解为由其特征值和特征向量表示的矩阵乘积的方法。只有含有n个线性无关的特征向量的n维方阵才可以进行特征分解。

拉普拉斯矩阵是半正定矩阵，有如下三个性质：

1. 是实对称矩阵，有n个线性无关的特征向量
2. 这些特征向量都可以正交单位化而得到一组正交且模为 1 的向量
3. 所有的特征值非负

性质1告诉我们拉普拉斯矩阵一定能进行特征分解(谱分解)，且有如下形式：

$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^{-1} \tag{1}$$

其中$U = (u_1, u_2, \ldots, u_n)$为**列向量**$u_i$组成的单位特征向量矩阵，$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$是$n$个特征值组成的对角阵。

由性质2可知，U是正交矩阵，即$UU^\top = I$，故$U^\top = U^{-1}$所以特征分解又可以写成：

$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^\top \tag{2}$$

这种写法是大部分文献中的写法。

图上的傅里叶变换

傅里叶正变换：从Spatial域到Spectral域

介绍了Laplacian矩阵，我们将图从Spatial domain变换到Spectral domain的准备工作就完成了。下面我们来介绍一下傅里叶变换。

连续域的傅里叶变换定义为：

$$F(\omega) = \mathcal{F}[f(t)] = \int f(t) e^{-i\omega t} dt \tag{3}$$

即信号$f(t)$与基函数$e^{-i\omega t}$的积分，其中基函数$e^{-i\omega t}$满足：

$$\Delta e^{-i\omega t} = \frac{\partial^2}{\partial t^2} e^{-i\omega t} = -\omega^2 e^{-i\omega t} \tag{4}$$

Δ 是拉普拉斯算子，即n维欧式空间中的一个二阶微分算子$\Delta f = \nabla^2 f$，拉普拉斯算子是对函数f的一种变换。

联想到广义特征值方程的定义：$Au = \lambda u$，其中A是矩阵(在线性代数上矩阵左乘表示对向量进行线性变换)，u是特征向量或者特征函数(无穷维的向量)，λ是u对应的特征值。由类比可知$e^{-i\omega t}$是Δ的特征函数，ω和对应的特征值密切相关。

因此，若要将傅里叶变换迁移到图上，我们有了拉普拉斯矩阵(拉普拉斯矩阵是离散的拉普拉斯算子)，下一步自然就是去找拉普拉斯矩阵的特征向量了(相当于我们有了Δ，下一步是要找Δ相应的$e^{-i\omega t}$)。

小结一下：傅里叶变换迁移到图上核心的工作就是把拉普拉斯算子的特征函数$e^{-i\omega t}$对应到图拉普拉斯矩阵的特征向量上。因此求解图拉普拉斯矩阵的特征向量是至关重要的。而根据上文的介绍，图拉普拉斯矩阵是半正定矩阵，可以通过特征分解(谱分解)求得特征向量，即$u_1, \cdots, u_n$。更加详细的内容可以参考 [The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains]

有了特征向量$u_1, \cdots, u_n$之后，我们就可以定义图上的傅里叶变换了，将连续的傅里叶变换写成离散积分(求和)的形式，仿上定义可得图上的傅里叶变换为：

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} f(i) u_l(i) \tag{5}$$

其中f是对图上节点的变换，比如说返回节点Embedding，$f(i)$和图上的节点一一对应，$u_l(i)$表示第l个特征向量的第i个分量。那么节点Embedding$f(i)$在特征值λ_l下的图傅里叶变换就是与λ_l对应的特征向量u_l进行内积运算。

利用矩阵乘法将图上的傅里叶变换推广到矩阵形式：

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix} \tag{6}$$

即f在图上的傅里叶变换的矩阵形式为：

$$\hat{f} = U^{\top} f \tag{7}$$

简而言之，给定输入节点Embedding f, 左乘$U^{\top}$ 即可以得到f 经图上傅里叶变换的输出Embedding $\hat{f}$ 。

傅里叶逆变换：从spectral域到spatial域

那么，我们将节点从空间域变换到频率域之后，怎么变换回空间域呢？传统的傅里叶逆变换是对频率ω积分：

$$\mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\Pi} \int F(\omega) e^{i\omega t} d\omega \tag{8}$$

类比到到离散域(图上)则为对特征值λ_l求和：

$$f(i) = \sum_{l=1}^{N} \hat{f}(\lambda_l) u_l(i) \tag{9}$$

利用矩阵乘法将图上的傅里叶逆变换推广到矩阵形式：

$$\begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_2(1) & \dots & u_N(1) \\ u_1(2) & u_2(2) & \dots & u_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(N) & u_2(N) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} \hat{f}\left(\lambda_1\right) \\ \hat{f}\left(\lambda_2\right) \\ \vdots \\ \hat{f}\left(\lambda_N\right) \end{pmatrix} \tag{10}$$

即f在图上的傅里叶逆变换的矩阵形式为：

$$f = U\hat{f} \tag{11}$$

简而言之，给定节点Embedding的在频率域上的表示$\hat{f}$，左乘U 即可得到节点Embedding在空间域上的表示。

图上的卷积

上面我们介绍了作用于图上的傅里叶变换和傅里叶逆变换的定义，在上面的基础上，下面我们利用卷积定理推导图上进行卷积运算的形式：

> 卷积定理：函数卷积的傅里叶变换是函数傅里叶变换的乘积，即对于函数$f(t)$与$h(t)$两者的卷积是其傅里叶变换乘积的逆变换：

$$f(t) \star h(t) = \mathcal{F}^{-1}[\hat{f}\left(\omega\right)\hat{h}(\omega)] = \frac{1}{2\Pi}\int \hat{f}\left(\omega\right)\hat{h}(\omega)e^{i\omega t}d\omega \tag{12}$$

其中$\star$表示卷积运算，因此f与卷积核h(卷积核即是一个函数，在CNN上，卷积核通常是一个矩阵，表示对输入的线性变换)在图上的卷积操作可按照如下步骤求得：

1. 计算f的傅里叶变换：$\hat{f} = U^\top f$
2. 卷积核 h 的在图上的傅里叶变换为：$\hat{h} = U^\top h$，写成对角矩阵形式为：$\begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix}$，其中$\hat{h}\left(\lambda_l\right) = \sum_{i=1}^{N} h(i)u_l^*(i)$
3. 两者傅里叶变换的乘积为：$\hat{h}\hat{f} = U^\top h \odot U^\top f = \begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^\top f$
4. 再乘以U求两者傅里叶变换乘积的逆变换，则求出卷积：

$$(f \star h)_{\mathcal{G}} = U \begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^\top f \tag{13}$$

讲到这里，我从宏观的角度总结一下以上我们做了什么。下面这几段话对理解全文有很大帮助，不妨多读几遍领会全意。

我们的目的是对空间域中节点的Embedding进行卷积操作(即聚合邻居Embedding信息)，然而图数据和图像数据的差别在于节点邻居个数、次序都是不定的，因此传统用于图像上的CNN模型中的卷积操作(Convolution Operator)不能直接用在图上，因此需要从频谱域上重新定义这样的卷积操作再转换回空间域上(通过卷积定理)。

为了在频谱域和空间域中转换，我们借助了傅里叶公式，并且定义了图上傅里叶变换(从空间域变换到频谱域)和图上傅里叶逆变换(从频谱域域回到空间域)的变换公式。具体操作是我们将节点的Embedding $f(i), i \in (1, \cdots, N)$通过傅里叶正变换从空间域变换到了频谱域$\hat{f}$，在频谱域上和卷积核$h$进行卷积操作，再将变换后的节点Embedding通过傅里叶逆变换回到空间域，参与后续的分类等任务。

卷积核h通俗来讲可以看做为空间域上的一个矩阵，在频谱域上和节点Embedding进行卷积之前，我们同样要通过傅里叶变换将h变换到频谱域$\hat{h}$，卷积核和节点Embedding在频谱域上的卷积定义为$\hat{h}\hat{f}$，最后在通过傅里叶逆变换将更新节点的Embedding变回空间域。

到这里为止，图上的卷积就介绍完了，下面我们看看各种图神经网络模型是怎么利用它的。

GCN 结构的演进

演进 1

上文中定义的图卷积操作

$$(f * h)_{\mathcal{G}} = U \begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^{\top} f \tag{15}$$

是对节点Embedding f在图中做的一次变换，其中图结构信息蕴含在了U中，那么对于图上的学习任务比如分类任务，我们需要找到一个合适的卷积核h，使得f经过卷积核h的卷积变换后能够降低分类任务定义的损失。因此图上的机器学习任务的核心就是找出可以降低损失(loss)的卷积核h，更直接得说，找到上式中适合的$\hat{h}(\lambda_1), \cdots, \hat{h}(\lambda_n)$(它们是$\mathrm{diag}(\hat{h}(\lambda_i))$中的自由参数)。

将$\hat{h}(\lambda_1), \cdots, \hat{h}(\lambda_n)$看做是模型的参数，我们可以利用梯度下降法使模型自动学习这些参数，即根据任务的loss，学习适合的卷积核。Spectral Networks and Locally Connected Networks on Graphs 用的就是这样的思路，它简单粗暴地将$\mathrm{diag}(\hat{h}(\lambda_i))$当做是参数$\mathrm{diag}(\theta_i)$：

$$\begin{aligned} y_{out} &= \sigma(U g_\theta(\Lambda) U^{\top} x) \\ g_\theta(\Lambda) &= \begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{pmatrix} \end{aligned} \tag{16}$$

这个式子可以看做是"第一代"的GCN中的一层(layer)，θ_i和神经网络中的weights一样是任意的参数，可以通过反向传播算法进行学习，x即f，对应于节点的Feature Embedding

这种模型固然简单，但是存在着一些问题：

1. 每一次前向传播(每计算一次y_{out})，都要计算U， $g_\theta(\Lambda)$和$U^{\top}$三者的矩阵乘积，运算复杂度为$\mathcal{O}(n^3)$
2. 卷积核是由$\theta_1, \cdots, \theta_n$ 这样n个参数构成的，其中n是图中的节点数，对于非常大的图，模型的自

由度过高

演进 2

上面提到卷积核由n个自由的参数构成，在一些情况下，我们不希望模型的参数过多，否则在数据量不够的情况下，模型可能陷入欠拟合的情况，因此在Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering中作者提出了一种多项式近似方法：

$$\hat{h}(\lambda_i) = \sum_{j=0}^{K} \alpha_j \lambda_i^j \tag{17}$$

其中λ_i^j表示λ_j的 j 次幂，即

$$\begin{aligned} y_{out} &= \sigma(U g_\theta(\Lambda) U^\top x) \\ g_\theta(\Lambda) &= \begin{pmatrix} \sum_{j=0}^{K} \alpha_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^{K} \alpha_j \lambda_n^j \end{pmatrix} \end{aligned} \tag{18}$$

这样做带来了另一个优点，经过推导不难发现，我们可以进一步简化计算：

$$g_\theta(\Lambda) = \begin{pmatrix} \sum_{j=0}^{K} \alpha_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^{K} \alpha_j \lambda_n^j \end{pmatrix} = \sum_{j=0}^{K} \alpha_j \Lambda^j \tag{19}$$

$$y_{out} = \sigma(U g_\theta(\Lambda) U^\top x) = \sigma(\sum_{j=0}^{K} \alpha_j U \Lambda^j U^\top) = \sigma(\sum_{j=0}^{K} \alpha_j L^j) \tag{20}$$

上式中省略了$L^j = U\Lambda^j U^\top$的证明，举个例子：$L^2 = U\Lambda U^\top U \Lambda U^\top = U\Lambda^2 U^\top$，因为$U^\top U = I$，可以利用数学归纳法证明该式，在此不赘述。

因此，最终

$$y_{out} = \sigma(U g_\theta(\Lambda) U^\top x) = \sigma(\sum_{j=0}^{K} \alpha_j L^j x) \tag{21}$$

其中$\alpha_j, j \in (1, \cdots, K)$是模型的参数，可以通过反向传播学习得到。

那么相比于演进1，演进2有几个明显的优点：

1. 模型参数由$\theta_i, i \in (1, \cdots, n)$变到了$\alpha_j, j \in (1, \cdots, K)$，由于$K \ll n$，模型的自由度大大降低，减少了欠拟合的风险
2. 不再需要对拉普拉斯矩阵L进行特征分解$L = U\Lambda U^\top$了，因为输出y_{out}可以直接写成关于L^j的表达式，特征分解的时间节约了(其复杂度为$\mathcal{O}(n^3)$)
3. 通过对卷积核的多项式近似，我们引入了所谓的空间局部性(spatial localization)，K被称为感知

野(receptive field)，表示对每个节点的Embedding更新只会涉及到它K-hop以内邻居Embedding的聚合，Hop k所有节点的加权聚合系数都为α_k

但是，这种方法仍然没有从根本上解决计算复杂度的问题，因为仍需要求解L^j，其复杂度为$\mathcal{O}(n^3)$。

演进 3

演进2中使用多项式近似的方式避免对L进行谱分解，且降低了模型的自由度，但是并没有降低卷积计算的复杂度，论文Wavelets on graphs via spectral graph theory中提出了一种用Chebyshev多项式近似卷积核的方法，可以用来降低卷积运算的复杂度。定义为：

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}) \tag{22}$$

其中θ_k是Chebyshev多项式的系数，$T_k(\tilde{\Lambda})$是取$\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{\max}} - I$的Chebyshev多项式，对$\Lambda$进行变换的原因是Chebyshev多项式的输入要求在$[-1, 1]$之间。

Chebyshev多项式是由如下公式递归定义的：

$$\begin{gathered} T_k(\tilde{\Lambda})x = 2\tilde{\Lambda}T_{k-1}(\tilde{\Lambda})x - T_{k-2}(\tilde{\Lambda})x \\ T_0(\tilde{\Lambda}) = I, T_1(\tilde{\Lambda}) = \tilde{\Lambda} \end{gathered} \tag{23}$$

其中，x是节点的embedding。

相比于演进 2，我们可以看到，计算$g_\theta(\Lambda)$不再需要矩阵相乘了，因为$T_k(\tilde{\Lambda})$的计算可以使用递推的方式实现，递推中的每一步都是线性复杂度的，计算$T_k(\tilde{\Lambda})$的复杂度为$\mathcal{O}(n^2)$，因此计算$g_\theta(\Lambda)$的复杂度为$\mathcal{O}(Kn^2)$

将Chebyshev近似带入到谱图卷积的公式里：

$$y_{out} = \sigma(U g_\theta(\Lambda) U^\top x) \approx \sigma\left(\sum_{k=0}^{K} \theta_k T_k(\tilde{L}) x\right) \tag{24}$$

其中 $U g_\theta(\tilde{\Lambda}) U^\top \approx U\left(\sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda})\right) U^\top = \sum_{k=0}^{K} \theta_k T_k(\tilde{L})$可由数学归纳法推导而来，即证明 $U T_k(\tilde{\Lambda}) U^\top = T_k(\tilde{L}), k \in (1, \cdots, K)$

当$k = 0$或$k = 1$时，结论显然成立，当$k = n - 1$时，假设$U T_{n-1}(\tilde{\Lambda}) U^\top = T_{n-1}(\tilde{L})$成立，那么当$K = n$时，有

$$U T_n(\tilde{\Lambda}) U^\top = 2T_{n-1}(\tilde{\Lambda}) U\tilde{\Lambda}U^\top - T_{n-2}(\tilde{\Lambda}) = 2T_{n-1}(\tilde{L})\tilde{L} - T_{n-2}(\tilde{L}) = T_n(\tilde{L}) \tag{25}$$

得证。同理，$T_k(\tilde{L})$也可以通过递推的方式计算，避免了与矩阵$U, U^\top$的乘法运算，复杂度为$\mathcal{O}(Kn^2)$。因此根据以上推导，使用Chebyshev近似降低了卷积核计算的复杂度。

演进 4 - GCN

通过使用Chebyshev多项式近似，我们已经成功得降低了卷积核计算的复杂度。Semi-Supervised Classification with Graph Convolutional Networks 这篇文章中进一步对卷积核的计算做了近似，产生了我们所熟知的GCN逐层传播公式。

在GCN模型中，作者首先做了额外的两个假设：$\lambda_{\max} \approx 2, K = 1$, 这两个假设可以大大简化模型。而作者希望假设造成的误差可以通过神经网络参数训练过程来自动适应。在这两个假设的条件下：

$$y_{out} = \sigma(U g_\theta(\Lambda) U^\top x) \tag{26}$$
$$U g_\theta(\Lambda) U^\top x \approx \sum_{k=0}^{1} \theta_k T_k(\tilde{\Lambda}) x = \theta_0 x + \theta_1 \left(L - I\right) x = \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

上式有两个参数θ_0和θ_1。这两个参数可以在整个图所有节点的计算中共享。在实践中，进一步限制参数的个数能够一定程度上避免过拟合的问题，并且减少计算量。因此作者又引入了另一个假设：
$\theta = \theta_0 = -\theta_1$做进一步的近似：

$$U g_\theta(\Lambda) U^\top x \approx \theta \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x \tag{27}$$

注意$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$的特征值被限制在了$[0, 2]$中。由于这一步生成的$y_{out}$可能作为下一层的输入$x$，会再次与$I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$相乘重复这样的操作将会导致数值不稳定、梯度弥散/爆炸等问题。为了缓解这样的问题，作者引入了这样的再正则化(renormalization)技巧：$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$，其中$\tilde{A} = A + I_N, \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$

最后得到了大家耳熟能详的GCN逐层更新公式：输入节点矩阵$X \in \mathbb{R}^{N \times C}$, 每个输入节点有$C$个通道(channels, 即每个图节点有$C$维特征)，卷积操作包含$F$个滤波器(filters)或特征映射(feature maps), 如下：

$$Y = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta\right) \tag{28}$$

其中$\Theta \in \mathbb{R}^{C \times F}$ 是filters的参数矩阵, $Y \in \mathbb{R}^{N \times F}$ 是卷积之后的节点矩阵。

后话

本文详细介绍了GCN的逐层传播公式是如何由谱图卷积一步步推导而来的，我们定义了图上的傅里叶变换和傅里叶逆变换，并通过卷积定理将频谱域上的卷积转换到空间域上，最后通过一系列的近似操作得到了GCN的逐层传播公式，这个简单优雅的公式背后蕴藏着复杂的数学推导，很有启发意义。在GCN的论文中，作者还提到了另一种得出GCN逐层传播公式的方式—从Weisfeiler-Lehman算法的角度，接下来我们将会利用一个专栏的文章介绍Weisfeiler-Lehman算法，并且分析图神经网络的表达能力。

Reference

[The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains]

[Spectral Networks and Locally Connected Networks on Graphs]

[Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering]

[Wavelets on graphs via spectral graph theory]

Semi-Supervised Classification with Graph Convolutional Networks

如何理解 Graph Convolutional Network? 来自superbrother的回答