蘑菇先生学习记

Graph Convolutional Networks

2019-02-24 | 推荐系统 | 阅读量 1414

Graph Convolutional Networks图卷积网络涉及到两个重要的概念，Graph和Convolution。传统的卷积主要应用于Euclidean Structure的数据上（排列很整齐、Grid形式的），如图像、语句等，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Non-Euclidean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Graph等Non-Euclidean数据上，GCN应运而生。那么GCN是如何将卷积推广到Graph上的呢？

- 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。即：Convolution —— Fourier
- 傅里叶变换又可以通过谱图理论推广到Graph上进行变换。Fourier —— Spectral Graph

  因此自然而然，Convolution —— Fourier —— Spectral Graph，Convolution通过傅里叶变换和Graph发生了联系。

从整个的研究进程来看，首先是研究GSP（Graph Signal Processing）的学者提出了Graph上的Fourier Transformation，进而定义了Graph的Convolution，最后与深度学习结合起来，发展出来GCN。

下文主要先介绍数学中的傅里叶变换，再介绍Graph上的傅里叶变换。最后介绍卷积如何应用在Graph上。

傅里叶变换

傅里叶变换可以从多种角度进行表述。

从数学角度，傅立叶变换就是将**周期函数**转化为一组**正交基**下的**坐标表示**，这个**坐标表示**就是傅立叶变换的结果。换句话说，周期函数是这些正交基的**线性组合**(向量的叠加)，线性组合**系数构成的向量**就是傅立叶变换的结果。

从信号处理领域角度，傅里叶变换将一个周期函数从**时域**（时间与振幅的关系）转化为**频域**（频率与振幅的关系）。做个类比，正交基选择的是正弦函数，每个正弦函数有个**频率**参数值，而每个正弦函数的**振幅**参数就是该基下对应的坐标值。所有正弦函数的**振幅构成的向量**就是傅立叶变换的结果。

下面以信号处理领域为例，来进一步理解傅里叶变换。

时域和频域

理解傅里叶变换，需要理解两个核心概念：

- 时域：时间和振幅的关系图，横坐标是时间，纵坐标是振幅。
- 频域：频率和振幅的关系图，横坐标是频率，纵坐标是振幅。

傅里叶级数

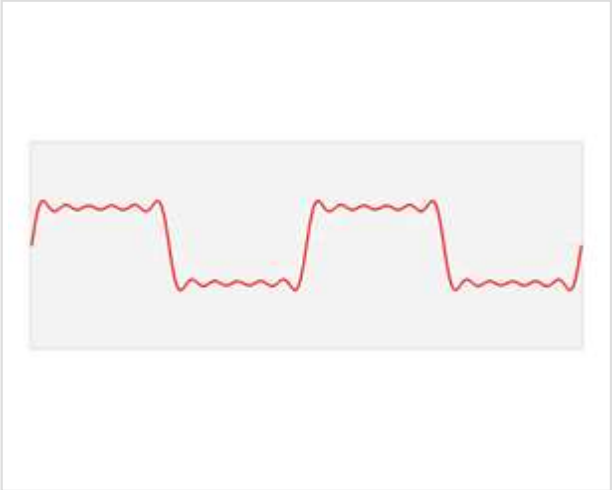
任何**周期(T)**函数，都可以使用**傅立叶级数展开法**将它们分解为有限或无限个不同**频率**不同**振幅**的正弦函数的叠加。傅里叶级数展开公式如下：

$$f(x) = \sum_{n=1}^{\infty} \left(a_n \cos(\frac{2\pi n}{T}x) + b_n \sin(\frac{2\pi n}{T}x) \right) + C, C \in \mathbb{R}$$

振幅a_n, b_n有具体的计算公式。$\frac{2\pi n}{T}$是频率。求和叠加就是线性组合。如果把函数f看成离散点构成的向量，那么就是这些正弦函数**基向量**的线性组合。

举例

举例：分解动图如下：



某个周期函数$s_6(x)$,可以分解为一系列正弦函数的和：

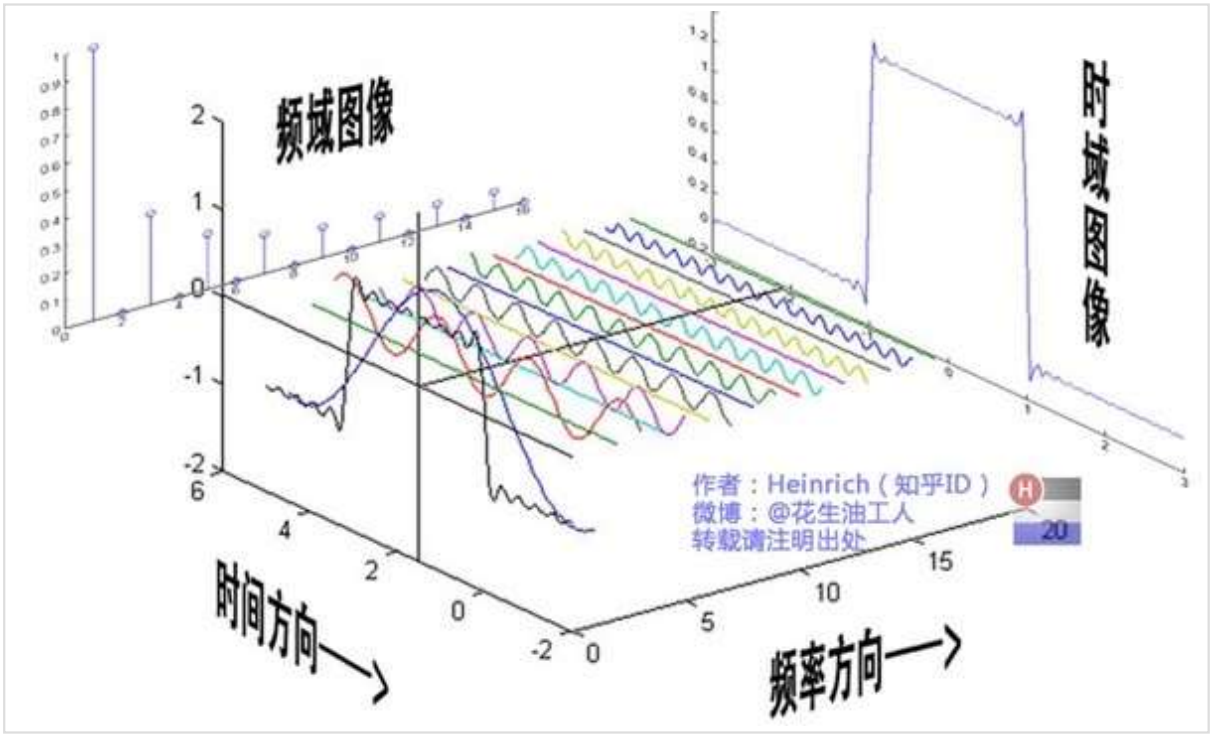
$$f(t) = A_0 + A_1 \sin wt + A_2 \sin 2wt + \ldots + A_\infty \sin(\infty)wt$$

其中各个分量的**振幅**A_i（有正有负）的确定有专门的公式，此处不详述。而各个分量的**频率**aw（w是角频率，$f = \frac{w}{2\pi}$）恰好是原来函数频率的整数倍(公式中$2\pi n/T$为整数)，是确定的。

经过分解后的公式完全可以用另一幅图来表示，横坐标为各个分量的**频率**（不断递增的整数），纵坐标为对应**振幅**。即图中，侧面观看到的$S(f)$。也就是说，频域图中每个点**(频率，振幅)**对应的就是一个正弦函数中的**频率和振幅**参数。所有正弦函数的频率和振幅参数点组成整幅频域图。

故，周期函数可以通过**傅立叶级数**画出频域图。

进一步欣赏下列图：



傅里叶级数是向量

首先，我们一般描述向量的时候，都有对应的基，即在某组基下的坐标表示构成了向量。默认是单位基时，则不显示提到。

我们发现，给定周期函数T,频域图中的横坐标频率值实际上可以确定了，即$\frac{2\pi n}{T}$。进一步，横坐标频率代表的正弦函数实际上就是傅里叶级数的基，即：$[1, cos(\frac{2\pi n}{T}), sin(\frac{2\pi n}{T})]$，可以证明这些基相互正交。而系数$(C, a_n, b_n)$构成的**向量就是傅里叶级数**。因此，傅里叶级数是向量。

也就是说，频域下，**某个曲线**是表示成了关于**正弦函数正交基下**的傅里叶级数向量。而在时域下，某个曲线是表示成了关于**时间**的周期函数。不管时域还是频域，其实反映的都是同一个曲线，只是一个是用函数的观点，一个是用向量的观点。

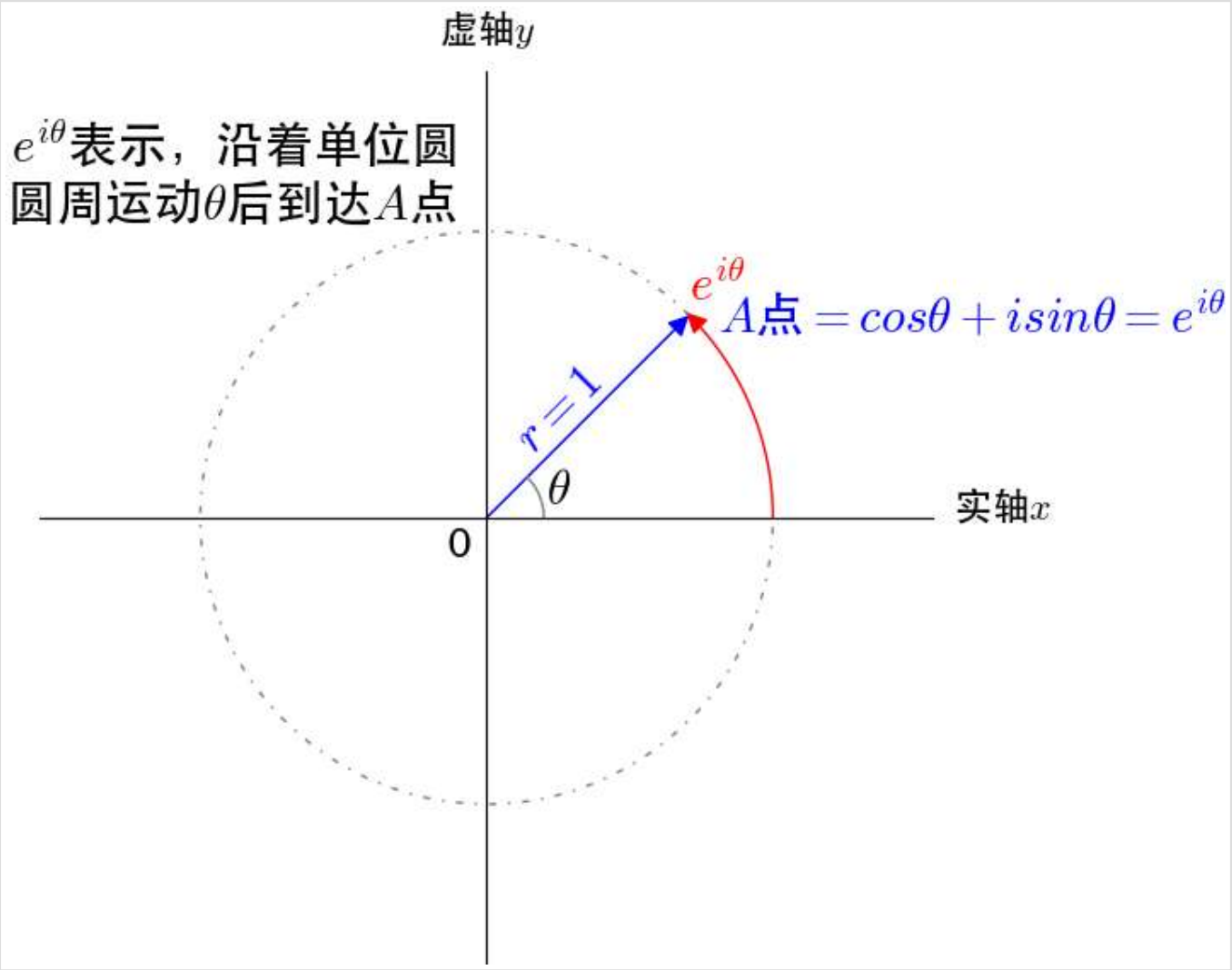
欧拉公式

首先介绍欧拉公式。我们知道，根据欧拉公式有：（参考如何通俗地解释欧拉公式）

$$e^{i\theta} = \cos\theta + i\sin\theta$$

欧拉公式可以根据泰勒展开推出。通常的理解，我们可以把$e^{i\theta}$看做是**单位圆的圆周运动**来

描述单位圆上的点，$\cos\theta + i\sin\theta$通过**复平面的坐标**来**描述单位圆上的点**，是同一个点不同的描述方式，所以有$e^{i\theta} = \cos\theta + i\sin\theta$.



接着，根据欧拉公式，重新表示$\sin(t)$

在我们的例子中，$\theta \to t, e^{i\theta} \to e^{it}$。有：

$$e^{it} = \cos t + i\sin t$$

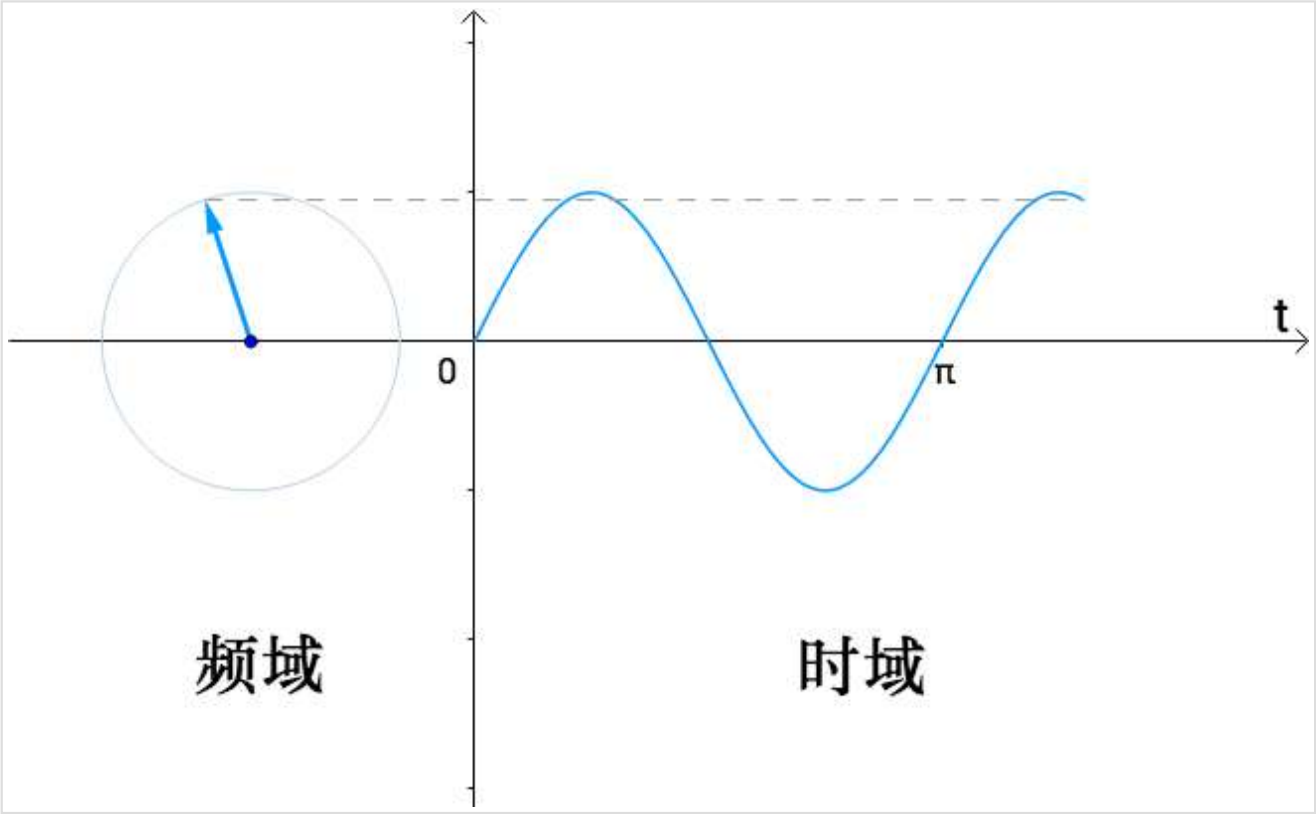
随着时间流逝，t从0开始增长，e^{it}这个向量就会旋转起来，2π会转1周，也就是$T = 2\pi$。随着圆周的运动，我们把e^{it}向量的**虚部**记录下来（也就是纵坐标），得到的就是$\sin t$；在时间轴t上，把e^{i2t}的虚部记录下来就是$\sin 2t$。以此类推。

同理，在时间轴t上，把e^{it}的**实部**记录下来，就是$\cos t$。

更一般的我们认为，我们具有两种看待$\sin(x), \cos(x)$的角度：

$$e^{iwt} \Leftrightarrow \begin{cases} sin(wt) \\ cos(wt) \end{cases}$$

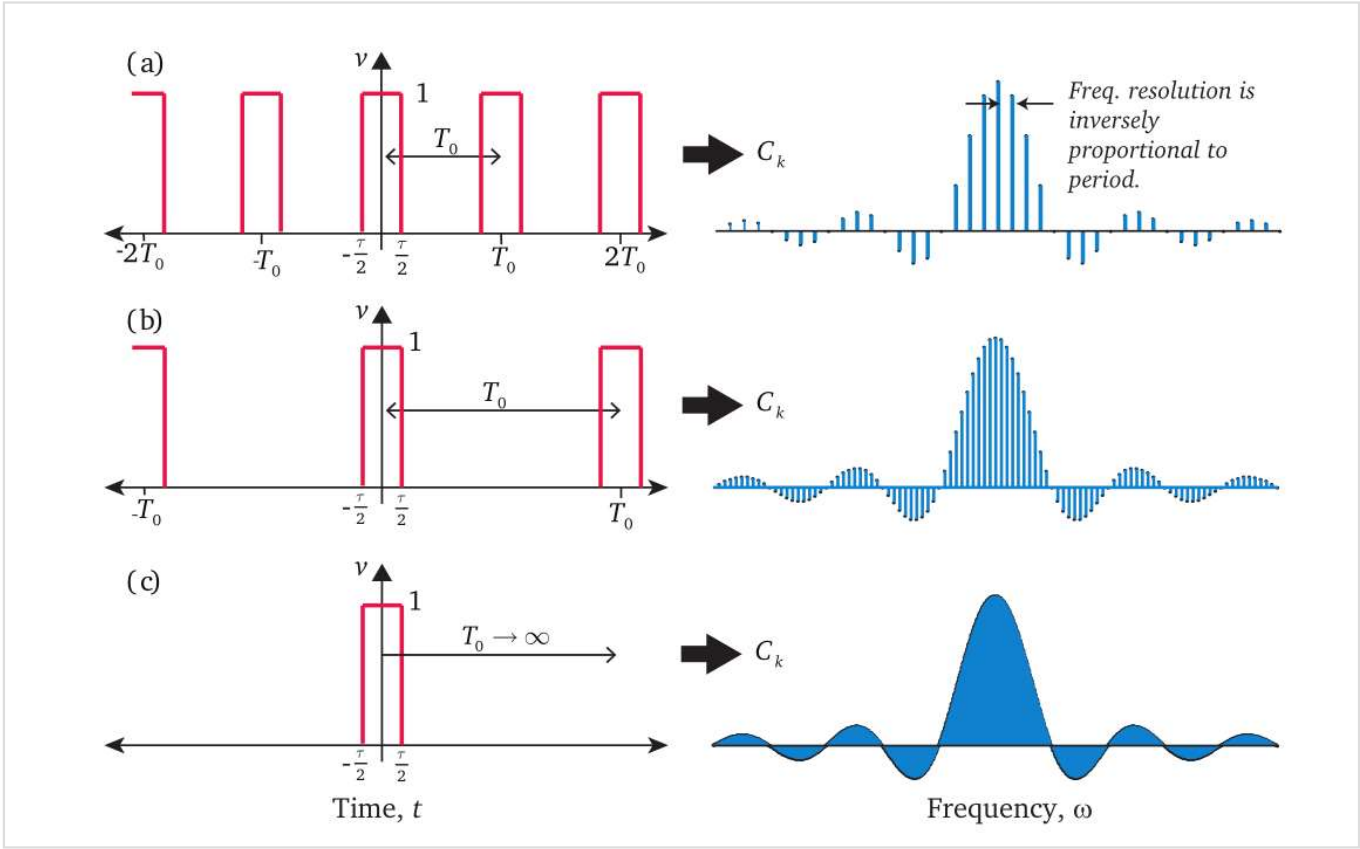
可以用下图来形象化刻画：



这两种角度，一个可以观察到旋转的频率(w)，所以称为**频域**；一个可以看到流逝的时间(t)，所以称为**时域**。

傅里叶变换

实际上，周期越大，画出的频域图越密集，当$T \to \infty$时，傅里叶级数就为傅里叶变换，此时频域图是**连续的曲线**。如下图：



傅里叶级数展开公式中有正弦波，也有余弦波，画频域图也不方便，通过欧拉公式，可以修改为复数形式：

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \cdot e^{i\frac{2\pi n}{T}x}$$

复数形式也可以理解为向量，$e^{i\frac{2\pi n}{T}x}$是基，c_n是该基下的坐标。不过c_n也是复数，不太好画频域图。

当周期T无穷大时，

$$f(x) = \int_{-\infty}^{\infty} F(w)e^{iwx}dw = \sum_{w} F(w)e^{iwx}$$

上面简化了一下，用w代表频率。这个公式也叫做**逆傅里叶变换**。

从数学角度，$f(x)$是函数f在x处的取值，所有基都对该处取值有贡献，即把每个$F(w)$投影到e^{iwx}基方向上分量累加起来，得到的就是该点处的函数值。

其中可以计算，

$$F(w) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x)e^{-iwx}dx$$

$F(w)$就是**傅里叶变换**，得到的就是**频域曲线**。每个频率w下都有对应的振幅$F(w)$。

从数学角度，$F(w)$就是每个基下对应的坐标值，所有的x对该基都有贡献，即把每个$f(x)$投影到e^{-iwx}基方向上的分量全部累加起来，得到的就是该基方向的坐标值。

下面两者称为傅立叶变换对，可以相互转换：

$$f(x) \Leftrightarrow F(w)$$

正如之前说的，这是看待同**一**个数学对象的两种形式，**一**个是函数，**一**个是**向量**（频域曲线纵坐标构成的向量，基为频域曲线横坐标对应的的基函数）。

Graph傅里叶变换

图上的傅里叶变换是通过下述联系实施的：

- 图拉普拉斯算子， Laplacian Operator—— Graph Laplacian Matrix。
- 图拉普拉斯的谱分解，Graph Laplacian Matrix —— Spectral Decomposition。
- Graph上Dirichlet Energy最小的基， Dirichlet Energy—— Orthonormal Basis —— Spectral Decomposition —— Eigenvectors
- 傅里叶变换，Fourier —— Fourier Basis —— Laplacian eigenfunctions

根据4，可以证明，Fourier basis = Laplacian eigenfunctions，即Fourier的基(和频率一一对应)是拉普拉斯算子的特征函数 (满足特征方程)。根据1，在Graph上，拉普拉斯算子为拉普拉斯矩阵。根据2，拉普拉斯矩阵的谱分解得到的特征向量(和特征值一一对应)类比特征函

数。因此，传统傅里叶变换在Graph的拓展就是将**正弦函数**基替换换成Graph拉普拉斯矩阵的**特征向量**，正弦函数与频率一一对应，特征向量与特征值一一对应。而根据3，这一的替换的根源意义在于，Graph拉普拉斯矩阵的**特征向量**作为一组基的话，这组基是Graph上Dirichlet Energy最小的基。

因此，可以通过这一系列类比/桥接，实现在图上的傅里叶变换。

图拉普拉斯算子

首先是标准的拉普拉斯算子定义：

$$\Delta f = \sum_i \frac{\partial f}{\partial x^2}$$

即，非混合二阶偏导数的和。f是拉普拉斯算子作用的**函数**，求函数各向二阶导数再求和，定义为f上的拉普拉斯算子。注意这个是其严谨的数学定义，所有的xxx拉普拉斯算子都是其一个特例，或是某种情况下(比如离散情况下)的一种近似。由于拉普拉斯算子和二阶偏导数密切相关，而二阶偏导数能够衡量函数的smoothness（不变化或缓慢变化时二阶导为0；突变或噪声，则数值不为0。因此容易检测出突变点）。

接着看图像上的处理。图像是一种离散数据（看做是函数的一种特殊形式），那么其拉普拉斯算子必然要进行离散化。

从一阶导数定义说起：(一阶差分近似)

$$f'(x) = \frac{\partial f(x)}{\partial x} \approx f(x+1) - f(x)$$

则二阶导数近似等于其二阶差分。

$$\begin{aligned} f''(x) = \frac{\partial^2 f(x)}{\partial x^2} &= f'(x) - f'(x-1) \\ &= f(x+1) - f(x) - (f(x) - f(x-1)) \\ &= f(x+1) + f(x-1) - 2f(x) \\ &= [f(x+1) - f(x)] + [f(x-1) - f(x)] \end{aligned}$$

某个二阶导数等于其在所有自由度上微扰之后获得的增益。一维函数其自由度可以理解为2，分别是+1方向和-1方向，增益分别为：$f(x+1)-f(x)$ 和 $f(x-1)-f(x)$。总增益为所有方向增益的和。

同理对于图像而言，

$$\begin{aligned}(\Delta f)_{x,y} &= \frac{\partial f(x,y)}{\partial x^2} + \frac{\partial f(x,y)}{\partial y^2} \\ &\approx f(x+1,y)+f(x-1,y)-2f(x,y)+f(x,y+1)+f(x,y-1)-2f(x,y) \\ &= f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y) \\ &= [f(x+1,y)-f(x,y)]+[f(x-1,y)-f(x,y)]+[f(x,y+1)-f(x,y)] \\ &\quad +[f(x,y-1)-f(x,y)]\end{aligned}$$

上下左右共4个自由度$(1,0),(-1,0),(0,1),(0,-1)$（当然还可以任意的定义自由度，比如对角线也算的话，就是8个自由度。在卷积时，使用的拉普拉斯模板就对应着1种方式的自由度定义）。在图像上某一点(x,y)，其拉普拉斯算子的值$(\Delta f)_{x,y}$，**即为对其进行扰动，使其变化到相邻像素后得到的增益。**

这给我们一种形象的结论：**拉普拉斯算子就是在所有自由度上进行微小变化后获得的增益(另一种说明，Informally, the Laplacian measures how different the value of f at p is from the average value of the neighbors)。**

那么推广到Graph, 对于有N个节点的Graph, 其邻接矩阵为W。这个Graph的自由度为N。**因为如果该图是一个完全图，即任意两个节点之间都有一条边，那么对一个节点进行微扰，它可能变成任意一个节点。**

那么上面的函数f就理所当然是一个N维的向量，即：

$$\boldsymbol{f} = (f_1, \ldots, f_N)$$

其中，f_i表示函数在节点i的值（**跟节点相关的信息，如节点属性等，可以是标量或向量，**

最基本的向量信息就是节点编号one-hot向量，这里先用标量），类比$f(x,y)$在(x,y)的值。$\boldsymbol{f}$可以表示Graph（权重还没考虑）。

对于任意节点i，对i节点进行微扰，它可能变为任意一个与他相邻的节点$j \in N_i$。

前面提到，拉普拉斯算子就是在所有自由度上进行微小变化后获得的增益。对于图而言，从节点i变化到节点j增益是$f_j - f_i$。不过通常这里头写作取负数的形式（无非就是L=D-W还是W-D的问题），即$f_i - f_j$。**再考虑边的权重**，可以认为增益是：$w_{ij}(f_i - f_j)$。那么对于节点i，总的增益即为拉普拉斯算子在节点i的值：

$$
\begin{aligned}
(\Delta \boldsymbol{f})_i &= \sum_j \frac{\partial f_i}{\partial j^2} \\
&\approx \sum_{j \in N_i} w_{ij}(f_i - f_j) \\
&= \sum_j w_{ij}(f_i - f_j) \\
&= (\sum_j w_{ij}) f_i - \sum_j w_{ij} f_j \\
&= (\boldsymbol{D}\boldsymbol{f})_i - (\boldsymbol{W}\boldsymbol{f})_i \\
&= [(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{f}]_i
\end{aligned}
$$

上述$j \in N_i$去掉是因为非邻接点$w_{ij} = 0$。粗体代表矩阵或向量。

对于任意的i,都有上述的结论。故：

$$
\Delta \boldsymbol{f} = (\boldsymbol{D} - \boldsymbol{W})\boldsymbol{f}
$$

这个公式全称：图拉普拉斯算子作用在由图节点信息构成的向量上$\boldsymbol{f}$得到的结果等于图拉普拉斯矩阵和向量$\boldsymbol{f}$的点积。

左式$\Delta \boldsymbol{f}$要看成一个整体(而不是乘法)，即，图拉普拉斯算子作用在由图节点信息构成的向量上$\boldsymbol{f}$得到的结果。右边是矩阵乘法，即$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ 拉普拉斯矩阵和向量$\boldsymbol{f}$乘法（当节点的信息为向量时，$\boldsymbol{f}$为矩阵）。通常，$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ 图拉普拉斯矩阵也直接称作图拉普拉

斯算子，即$\Delta = \boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$，此时就可以认为，$\Delta \boldsymbol{f}$是乘法，只不过实际上**这种写法是等式右边的式子**，而不是左边的式子。

我们可以发现，图拉普拉斯算子作用在**节点信息**f_i上，并不会改变f_i的形状。当$f_i \in \mathbb{R}^h$，则$(\Delta f)_i \in \mathbb{R}^h$。只不过上述讨论都是基于$f_i$为标量展开的。

拉普拉斯矩阵的谱分解

拉普拉斯矩阵的谱分解就是特征分解。由于拉普拉斯矩阵时**半正定对称矩阵**的，因此拥有诸多优秀性质：

- **对称矩阵一定n个线性无关的特征向量**（n是Graph节点数）
- **半正定矩阵的特征值一定非负**
- **对阵矩阵的特征向量相互正交，即所有特征向量构成的矩阵为正交矩阵。**

谱分解：

$$\boldsymbol{L}\boldsymbol{\phi}_k = \lambda_k \boldsymbol{\phi_k}, k = 1, 2 \ldots, n$$

由于半正定对称矩阵，故有：

$$\boldsymbol{L} = \boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^T$$

$\boldsymbol{\Phi} = (\boldsymbol{\phi_1}, \boldsymbol{\phi_2}, \ldots, \boldsymbol{\phi_n}) \in \mathbb{R}^{n \times n}$ 是**列向量**为单位特征向量构成的正交矩阵。
$\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_n)$是由对应特征值构成对角阵。

Dirichlet Energy

Dirichlet Energy衡量了函数的平滑性Smoothness。Dirichlet Energy定义为：

$$E_{Dir}(\boldsymbol{f}) = \boldsymbol{f}^T \boldsymbol{\Delta} \boldsymbol{f}$$

$\boldsymbol{f}$是函数，$\boldsymbol{\Delta}$是拉普拉斯算子。

我们的目标是寻找**Graph**上Dirichlet Energy最小**一组单位正交基**（每个基都可以看出函数）。

巧合的是，这样的正交基正好就是对拉普拉斯矩阵$\boldsymbol{L}$进行谱分解得到的特征向量$\boldsymbol{\phi_1}, \ldots, \boldsymbol{\phi_n}$。

因此，这也是下文从传统傅里叶变换推广到Graph上傅里叶变换时，进行类比和替换的**理论解释**。

Graph傅里叶变换

回顾下传统的傅里叶变换，

$$F(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx \approx \langle f, e^{-ikx} \rangle$$

$F(k)$是傅里叶系数（即频率为k时的振幅值）。约等号去掉了常数系数，同时x为离散变量时，离散积分等于内积。

e^{-ikx}为Fourier Basis。可以证明e^{-ikx}是**拉普拉斯算子的特征函数**（满足特征方程$AV = \lambda V$），证明：

$$\Delta e^{-ikx} = \frac{\partial e^{-ikx}}{\partial x^2} = -k^2 e^{-ikx}$$

在Graph上作类比，$\boldsymbol{\phi}_k$是图拉普拉斯算子$\boldsymbol{L}$的**特征向量**（满足$\boldsymbol{L\phi_k} = \lambda_k \boldsymbol{\phi_k}$）。即，在Graph中，$\Delta = \boldsymbol{L}, e^{-ikx} = \boldsymbol{\phi_k}$，而$k$和特征值$\lambda_k$有关。

因此，为了在Graph上进行傅里叶变换，可以把传统傅里叶变换中的e^{-ikx}换成$\boldsymbol{\phi_k}$。（换了种**基**）

传统傅里叶变换	**图傅里叶变换**
频率（k）	特征值（λ_k）

传统傅里叶变换	**图傅里叶变换**
正弦函数（e^{-ikx}）	特征向量（$\boldsymbol{\phi}_k$）
振幅（$F(k)$）	振幅（$F(\lambda_k)$）

则，图上的傅里叶变换写作：

$$F(\lambda_k) = \hat{f}_k = \langle \boldsymbol{f}, \boldsymbol{\phi_k} \rangle$$

$\boldsymbol{f} = (f_1, \ldots, f_n)$是由节点信息构成的n维向量。做个类似的解释，即特征值λ_k(频率)下，$\boldsymbol{f}$的Graph傅里叶变换（振幅）等于$\boldsymbol{f}$与λ_k对应的特征向量$\boldsymbol{\phi_k}$的内积。

推广到矩阵形式，**图傅里叶变换**：

$$\hat{\boldsymbol{f}} = \boldsymbol{\Phi}^T \boldsymbol{f}$$

其中，$\hat{\boldsymbol{f}} = (\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n)$，即图傅里叶变换，即不同特征值(频率)下对应的振幅构成的向量。$\boldsymbol{f} = (f_1, .., f_n)$是由节点信息构成的n维向量。

类似的，传统**逆傅里叶变换**：（n个正弦波的叠加）

$$\mathcal{F}^{-1}[F(k)] = f(x) = \sum_k F(k) e^{ikx} = \sum_k \underbrace{\frac{1}{2\pi} \int_{-\infty}^{\infty} f(x') e^{-ikx'} dx'}_{\text{Fourier Coefficient}} e^{ikx}$$

迁移到Graph上的逆傅里叶变换：$e^{ikx} e^{-ikx} = 1$，两个基正交，类比于$(\boldsymbol{\phi}_k^T \boldsymbol{\phi}_k)_i = 1$。

$$f_i = \sum_{k=1}^{n} \hat{f}_k (\boldsymbol{\phi}_{\boldsymbol{k}}^T)_i$$

推广到矩阵形式，

$$\boldsymbol{f} = \boldsymbol{\Phi} \hat{\boldsymbol{f}}$$

个人理解，**Graph傅里叶变换是为了将Graph从Spatial Domain转换到Spectural Domain，使得Graph在Spectural Domain有向量化表示，卷积更方便。这就类比于，传统傅里叶变换为了将Function从Time Domain转换到Frequency Domain，使得Function在Frequency Domain有向量化表示。**

Graph Convolution

卷积定理：函数卷积的傅里叶变换是函数傅立叶变换的乘积，即对于函数f与h两者的卷积是其函数傅立叶变换乘积的逆变换。

$$\boldsymbol{f} * \boldsymbol{h} = \mathcal{F}^{-1}[\hat{\boldsymbol{f}} \circ \hat{\boldsymbol{h}}] = \sum_k (\hat{f}(k) \cdot \hat{h}(k)) e^{ikx}$$

类比到Graph上并把傅里叶变换的定义带入，$\boldsymbol{f} = (f_1 \ldots, f_n)$为Graph节点信息向量，$h$为卷积核，则$f$和$h$**在Graph上的卷积可按下列步骤求出：**

- $\boldsymbol{f}$的Graph傅里叶变换：$\hat{\boldsymbol{f}} = \boldsymbol{\Phi}^T \boldsymbol{f}$
- 卷积核$\boldsymbol{h}$的Graph傅里叶变换为：$\hat{\boldsymbol{h}} = (\hat{h}_1, \ldots, \hat{h}_n)$，其中，$\hat{h}_k = \langle h, \phi_k \rangle, k = 1, 2 \ldots, n$。实际上，$\hat{\boldsymbol{h}} = \boldsymbol{\Phi}^T \boldsymbol{h}$.

- 求图傅里叶变换向量$\hat{\boldsymbol{f}} \in \mathbb{R}^{N \times 1}$和$\hat{\boldsymbol{h}} \in \mathbb{R}^{N \times 1}$ 的element-wise乘积，等价于将$\hat{\boldsymbol{h}}$组织成对角矩阵的形式，即$diag[\hat{h}(\lambda_k)] \in \mathbb{R}^{N \times N}$，再求$diag[\hat{h}(\lambda_k)]$和$\hat{\boldsymbol{f}}$矩阵乘法（稍微想一下就可以知道）。

- 求上述结果的逆傅里叶变换，即左乘$\boldsymbol{\Phi}$。

则：图上的卷积定义为：

$$(\boldsymbol{f} * \boldsymbol{h})_{\mathcal{G}} = \boldsymbol{\Phi} \text{diag}[\hat{h}(\lambda_1), \ldots, \hat{h}(\lambda_n)] \boldsymbol{\Phi}^T \boldsymbol{f} \tag{1}$$

Deep Learning中的Graph Convolution

Deep Learning中的**Convolution就是要设计含有trainable**共享参数的kernel。**从公式1看很直观：graph convolution**的参数就是$\text{diag}[\hat{h}(\lambda_1), \ldots, \hat{h}(\lambda_n)]$，也就是说，简单粗暴的将其

变成了卷积核$diag[\theta_1,\dots,\theta_n]$。这也是为什么我们前面不把卷积核$\boldsymbol{h}$的Graph傅里叶变换表示为$\hat{\boldsymbol{h}}=\boldsymbol{\Phi}^T\boldsymbol{h}$的原因，我们要把$\boldsymbol{h}$变换后的结果$\hat{\boldsymbol{h}}$直接作为参数向量$\boldsymbol{\theta}\rightarrow\hat{\boldsymbol{h}}$进行学习。

可以得到第一代的GCN，(Spectral Networks and Locally Connected Networks on Graphs)：

$$\boldsymbol{y}_{output}=\sigma(\boldsymbol{\Phi}\boldsymbol{g}_\theta\boldsymbol{\Phi}^T\boldsymbol{x})=\sigma(\boldsymbol{\Phi}\mathrm{diag}[\theta_1,\dots,\theta_n]\boldsymbol{\Phi}^T\boldsymbol{x})$$

$\boldsymbol{x}$就是graph上对应于每个顶点的feature构成的向量$\boldsymbol{x}=(x_1,x_2,\dots,x_n)$，目前仍然是每个节点信息是标量（即单通道），后续推广到向量很方便（多通道）。 $\boldsymbol{y}_{output}$是该节点卷积后的输出。所有的节点都要经过该操作，得到各自的输出。再σ激活后，传入下一层。$\boldsymbol{g_\theta}=\mathrm{diag}[\theta_1,\dots,\theta_n]$。相当于拿这个卷积核每个节点卷一遍。

这个问题在于，

- 复杂度太高，需要对拉普拉斯矩阵进行谱分解求$\boldsymbol{\Phi}$，Graph很大时复杂度较高。每次前向传播都要计算矩阵乘积，复杂度$O(n^2)$，n为Graph节点数。
- 卷积核的参数为n，当Graph很大时，n非常大。
- 卷积核的spatial localization不好。（相对于下一代GCN而言）

第二代的GCN，Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

为了解决上述问题，首先回顾一下，**图傅里叶变换**是关于特征值(频率)的函数$F(\lambda_1),\dots,F(\lambda_n)$,即，$F(\boldsymbol{\Lambda})$，因此可以将上述卷积核$\boldsymbol{g}_\theta$写作$\boldsymbol{g_\theta}(\boldsymbol{\Lambda})$。接着，将$\boldsymbol{g_\theta}(\boldsymbol{\Lambda})$定义成如下**k阶多项式**形式：

$$\boldsymbol{g_{\theta'}}(\boldsymbol{\Lambda})\approx\sum_{k=0}^{K}\theta'_k\boldsymbol{\Lambda}^k$$

代入可以得到：

$$
\begin{aligned}
\boldsymbol{g}_{\boldsymbol{\theta}'} * \boldsymbol{x} &\approx \boldsymbol{\Phi} \sum_{k=0}^{K} \theta'_k \boldsymbol{\Lambda}^k \boldsymbol{\Phi}^T \boldsymbol{x} \\
&= \sum_{k=0}^{K} \theta'_k (\boldsymbol{\Phi} \boldsymbol{\Lambda}^k \boldsymbol{\Phi}^T) \boldsymbol{x} \\
&= \sum_{k=0}^{K} \theta'_k (\boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^T)^k \boldsymbol{x} \\
&= \sum_{k=0}^{K} \theta'_k \boldsymbol{L}^k \boldsymbol{x}
\end{aligned}
$$

上述推导第三步应用了特征分解的性质。

上式为第二代的GCN。不需要做特征分解了，直接对拉普拉斯矩阵进行变换。可以事先把$\boldsymbol{L}^k$计算出来，这样前向传播的时候，就只需要计算矩阵和相邻的乘法。复杂度为$O(Kn^2)$。如果使用稀疏矩阵（L比较稀疏）算法，复杂度为$O(k|E|)$.

那么上式是如何体现localization呢？我们知道，矩阵的k次方可以用于求连通性，即1个节点经过k步能否到达另一个顶点，矩阵k次方结果中对应元素非0的话可达，为0不可达。因此L矩阵的k次方的含义就是代表k-hop之内的节点。进一步，根据拉普拉斯算子的性质。可以证明，如果两个节点的最短路径大于K的话，那么L^K在相应位置的元素值为0。因此，实际上只利用到了节点的K-Localized信息。

另外，作者提到，可以引入切比雪夫展开式来近似$\boldsymbol{L}^k$，因为**任何k次多项式都可以使用切比雪夫展开式来近似**。(类比泰勒展开式对函数进行近似）。

引入切比雪夫多项式（Chebyshev polynomial) $T_k(x)$的K阶截断获得对$\boldsymbol{L}^k$的近似，进而获得对$\boldsymbol{g}_\theta(\boldsymbol{\Lambda})$的近似，来降低时间复杂度。

$$
\boldsymbol{g}_{\boldsymbol{\theta}'}(\boldsymbol{\Lambda}) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\boldsymbol{\Lambda}})
$$

其中，$\tilde{\boldsymbol{\Lambda}} = \frac{2}{\lambda_{max}} \boldsymbol{\Lambda} - \boldsymbol{I}_n$为经图拉普拉斯矩阵$L$的最大特征值（即谱半径）缩放后的特征向量矩阵（防止连乘爆炸）。$\boldsymbol{\theta}' \in \mathbb{R}^K$表示一个**切比雪夫向量**，$\theta'_k$是第$k$维分量。切比雪夫多

项式$T_k(x)$使用递归的方式进行定义：$T_k(x)=2xT_{k-1}(x)-T_{k-2}(x)$，其中$T_0(x)=1,T_1(x)=x$。

此时，可以使用近似的$\boldsymbol{g_{\theta'}}$替换原来的$\boldsymbol{g_\theta}$，可以得到：

$$\begin{aligned}\boldsymbol{g_{\theta'}}*\boldsymbol{x} &\approx \boldsymbol{\Phi}\sum_{k=0}^{K}\theta'_k T_k(\tilde{\boldsymbol{\Lambda}})\boldsymbol{\Phi}^T\boldsymbol{x} \qquad \approx \sum_{k=0}^{K}\theta'_k(\boldsymbol{\Phi}T_k(\tilde{\boldsymbol{\Lambda}})\boldsymbol{\Phi}^T)\boldsymbol{x}\\ &=\sum_{k=0}^{K}\theta'_k T_k(\tilde{\boldsymbol{L}})\boldsymbol{x}\end{aligned}$$

其中，$\tilde{\boldsymbol{L}}=\frac{2}{\lambda_{max}}\boldsymbol{L}-\boldsymbol{I}_n$。

因此有，

$$\boldsymbol{y}_{output}=\sigma(\sum_{k=0}^{K}\theta'_k T_k(\tilde{\boldsymbol{L}})\boldsymbol{x})$$

参数向量$\boldsymbol{\theta'}\in\mathbb{R}^k$，需要通过反向传播学习。时间复杂度也是$O(K|E|)$。

第三代的GCN对上式进一步简化，Semi-Supervised Classification with Graph Convolutional Networks

- 取$K=1$，此时模型是1阶的first-order proximity。即每层卷积层只考虑了直接邻域，类似CNN中3*3的卷积核。
- 深度加深，宽度减小。即，若要建立多阶 proximity，只需要使用多个卷积层。
- 并加了参数的一些约束，如: $\lambda_{max}\approx 2$，引入renormalization trick，大大简化了模型。

  具体推导，首先$K=1,\lambda_{max}=2$代入，

$$\begin{aligned}\boldsymbol{g_{\theta'}} * \boldsymbol{x} &\approx \theta'_0\boldsymbol{x} + \theta'_1(\boldsymbol{L} - \boldsymbol{I}_n)\boldsymbol{x} \\ &= \theta'_0\boldsymbol{x} + \theta'_1(\boldsymbol{L} - \boldsymbol{I}_n)\boldsymbol{x} \\ &= \theta'_0\boldsymbol{x} - \theta'_1(\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2})\boldsymbol{x}\end{aligned}$$

上述推导利用了归一化拉普拉斯矩阵
$\boldsymbol{L} = \boldsymbol{D}^{-1/2}(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{D}^{-1/2} = \boldsymbol{I_n} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$。此时只有两个参数，即每个卷积核只有2个参数，$\boldsymbol{W}$是邻接矩阵。

进一步简化，假设$\theta'_0 = -\theta'_1$，则此时单个通道的单个卷积核参数只有1个θ：

$$\boldsymbol{g_{\theta'}} * \boldsymbol{x} = \theta(\boldsymbol{I_n} + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2})\boldsymbol{x}$$

$\boldsymbol{I_n} + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$谱半径$[0, 2]$太大，使用renormalization trick，

$$\boldsymbol{I_n} + \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \rightarrow \tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{D}}^{-1/2}$$

其中，$\tilde{\boldsymbol{W}} = \boldsymbol{W} + \boldsymbol{I}_n$(相当于加了**self-connection**，本来$\boldsymbol{W}$对角元素为0)，
$\tilde{\boldsymbol{D}}_{i,i} = \sum_j \tilde{\boldsymbol{W}}_{ij}$。

则：

$$\underbrace{\boldsymbol{g_{\theta'}} * \boldsymbol{x}}_{\mathbb{R}^{n\times 1}} = \theta(\underbrace{\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{W}}\tilde{\boldsymbol{D}}^{-1/2}}_{\mathbb{R}^{n\times n}})\underbrace{\boldsymbol{x}}_{\mathbb{R}^{n\times 1}}$$

推广到**多通道**(单个节点的信息是向量，对比图像上3个通道RGB的值构成3维向量)和**多卷积核**(每个卷积核只有1个参数)，即，

$$\boldsymbol{x} \in \mathbb{R}^{N\times 1} \rightarrow \boldsymbol{X} \in \mathbb{R}^{N\times C}$$

其中，N是节点的**数量**，C是通道数，或者称作表示节点的**信息维度数。** $\boldsymbol{X}$是节点的feature矩阵。

相应的卷积核参数变化：

$$\theta \in \mathbb{R} \rightarrow \boldsymbol{\Theta} \in \mathbb{R}^{C \times F}$$

其中，F为卷积核数量。

则卷积结果写作矩阵形式如下：

$$\underbrace{\boldsymbol{Z}}_{\mathbb{R}^{N \times F}} = \underbrace{\tilde{\boldsymbol{D}}^{-1/2} \tilde{\boldsymbol{W}} \tilde{\boldsymbol{D}}^{-1/2}}_{\mathbb{R}^{N \times N}} \underbrace{\boldsymbol{X}}_{\mathbb{R}^{N \times C}} \underbrace{\boldsymbol{\Theta}}_{\mathbb{R}^{C \times F}}$$

最终得到的卷积结果$\boldsymbol{Z} \in \mathbb{R}^{N \times F}$。即，每个节点的卷积结果的维数等于卷积核数量。

上述操作可以叠加多层，对$\boldsymbol{Z}$激活一下，然后将激活后的Z作为下一层的节点的feature矩阵。

第三代GCN特点总结：

- 复杂度为$O(E)$ (稀疏矩阵优化的话)
- 只考虑1-hop，若要建模多hop，通过叠加层数，获得更大的感受野。（联想NLP中使用卷积操作语句序列时，也是通过叠加多层来达到获取长依赖的目的）。

参考

马同学: 从傅立叶级数到傅立叶变换

信号频域和时域的关系？ - 言东的回答 - 知乎

图拉普拉斯算子为何定义为D-W

如何理解 Graph Convolutional Network（GCN）？ - Becca的回答 - 知乎

The Emerging Field of Signal Processing on Graphs

Spectral Networks and Locally Connected Networks on Graphs

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Semi-Supervised Classification with Graph Convolutional Networks

如何理解 Graph Convolutional Network（GCN）？ - superbrother的回答 - 知乎

Semi-Supervised Classification with Graph Convolutional Networks阅读笔记

Graph Neural Network Review

坚持原创技术分享，您的支持将鼓励我继续创作！

赏

机器学习 # Paper # 推荐系统 # Graph Embedding # Convolution

Attention in Deep Learning

生成对抗网络

© 2019 xuetf

16553 | 37621