Graph Learning

1 Spectral Graph Theory基础

我们来看看Spectral Graph Theory，这是进行graph研究的必备工具，主要研究图各种矩阵的特征值和特征空间与图性质之间的关系。我们这里先只考虑无向图，设图有n个点，顶点集为V，邻接矩阵为W，邻接矩阵是对称的，边集$E=\{e_{ij}|W_{ij}>0\}$，$\mathcal{W}_i$表示每个顶点的权重。另外$D=diag(d), d_j=\sum_i W_{ij}$是一个对角阵。

图的Laplacian matrix 定义为，

$$L = D - W$$

Laplacian matrix 有一个有用的变换：对任意向量$f \in R^n$,

$$\begin{aligned}
&\textstyle\sum_{ij} W_{ij}(f_i - f_j)^2 \\
&= \textstyle\sum_{ij} W_{ij} f_i^2 - 2\sum_{ij} W_{ij} f_i f_j + \sum_{ij} W_{ij} f_j^2 \\
&= \textstyle\sum_i f_i^2 \sum_j W_{ij} - 2\sum_{ij} W_{ij} f_i f_j + \sum_j f_j^2 \sum_i W_{ij} \\
&= \textstyle\sum_i f_i^2 d_i - 2\sum_{ij} W_{ij} f_i f_j + \sum_j f_j^2 d_j \\
&= 2\textstyle\sum_i f_i^2 d_i - 2\sum_{ij} W_{ij} f_i f_j \qquad \text{换下脚标} \\
&= 2f^T D f - 2f^t W f \\
&= 2f^T L f
\end{aligned}$$

它有如下的重要性质：

- 其全部特征值是非负实数，也就是$\lambda_n \geq \cdots \geq \lambda_2 \geq \lambda_1 \geq 0$，且$\lambda_1 = 0$，其对应的特征向量$\boldsymbol{v}_1 = \mathbf{1}$。
- 其有几个特征值为0(也就是the multiplicity of 0)，那么图就有几个连通分量。

图中没有单独的点时($d_i = 0$)，normalized adjacency matrix定义为：

$$D^{-1/2} W D^{-1/2}$$

其中，

$$D^{-1/2} = \begin{bmatrix} \frac{1}{\sqrt{d_1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{d_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \vdots & \frac{1}{\sqrt{d_n}} \end{bmatrix}$$

normalized Laplacian matrix定义为：

$$\begin{aligned}
L^{norm} &= I - D^{-1/2} W D^{-1/2} \\
&= D^{-1/2} D D^{-1/2} - D^{-1/2} W D^{-1/2} \\
&= D^{-1/2}(D - W) D^{-1/2} \\
&= D^{-1/2} L D^{-1/2}
\end{aligned} \tag{1}$$

则我们可以算出：

$$L_{ij}^{norm} = \frac{L_{ij}}{\sqrt{d_i d_j}} = \begin{cases} 1 & i = j \\ \frac{-W_{ij}}{\sqrt{d_i d_j}} & i \neq j \end{cases}$$

normalized Laplacian matrix的特征值有这样的性质：

- $0 = \lambda_1^{norm} \leq \lambda_2^{norm} \leq \cdots \lambda_n^{norm} \leq 2$。
- λ_1^{norm}对应的特征向量$\boldsymbol{v}_1^{norm} = \sqrt{d}\mathbf{1}$。
- 同样地，其有几个特征值为0，那么图就有几个连通分量。

现在我们看看Spectral Clustering。我们现在先考虑把V划分两部分C_1, C_2。ratio-cut [1]和normalized-cut [2]是两个著名的谱聚类算法，他们(和其他很多Spectral Clustering 算法)有公共形式的loss 函数：

$$\frac{cut(C_1, C_2)}{weight(C_1)} + \frac{cut(C_1, C_2)}{weight(C_2)}$$

其中，

$$cut(C_1, C_2) = \sum_{i \in C_1} \sum_{j \in C_2} W_{ij}, weight(C) = \sum_{i \in C} \mathcal{W}_i$$

Ratio-cut中所有点的权重都是1，Normalized-cut中顶点的权重是d_i。

我们设置一个向量$q \in R^n$来表示每个顶点的划分结果：

$$q_i = \begin{cases} +\sqrt{\frac{\eta_2}{\eta_1}} & i \in C_1 \\ -\sqrt{\frac{\eta_1}{\eta_2}} & i \in C_2 \end{cases}$$

其中，$\eta_i = weight(C_i)$。那么

$$\begin{aligned}
2q^T Lq &= \textstyle\sum_{ij} W_{ij}(f_i - f_j)^2 \\
&= \sum_{i \in C_1} \sum_{j \in C_2} W_{ij}(f_i - f_j)^2 + \sum_{i \in C_2} \sum_{j \in C_1} W_{ij}(f_i - f_j)^2 \qquad \text{原理见下图1} \\
&= \sum_{i \in C_1} \sum_{j \in C_2} W_{ij}(\sqrt{\frac{\eta_2}{\eta_1}} + \sqrt{\frac{\eta_1}{\eta_2}})^2 + \sum_{i \in C_2} \sum_{j \in C_1} W_{ij}(-\sqrt{\frac{\eta_1}{\eta_2}} - \sqrt{\frac{\eta_2}{\eta_1}})^2 \\
&= \sum_{i \in C_1} \sum_{j \in C_2} W_{ij}(\frac{\eta_2}{\eta_1} + \frac{\eta_1}{\eta_2} + 2) + \sum_{i \in C_2} \sum_{j \in C_1} W_{ij}(\frac{\eta_1}{\eta_2} + \frac{\eta_2}{\eta_1} + 2) \\
&= cut(C_1, C_2)(\frac{\eta_2}{\eta_1} + \frac{\eta_1}{\eta_2} + 2) + cut(C_1, C_2)(\frac{\eta_1}{\eta_2} + \frac{\eta_2}{\eta_1} + 2) \\
&= 2cut(C_1, C_2)(\frac{\eta_2}{\eta_1} + \frac{\eta_1}{\eta_2} + 2) \\
&= 2cut(C_1, C_2)(\frac{\eta_2 + \eta_1}{\eta_1} + \frac{\eta_1 + \eta_2}{\eta_2}) \\
&= 2(\eta_1 + \eta_2)(\frac{cut(C_1, C_2)}{\eta_1} + \frac{cut(C_1, C_2)}{\eta_2}) \\
&= 2(\eta_1 + \eta_2)(\frac{cut(C_1, C_2)}{weight(C_1)} + \frac{cut(C_1, C_2)}{weight(C_2)})
\end{aligned}$$

所以我们的目标函数变为：

$$\frac{q^T Lq}{\eta_1 + \eta_2}$$

Figure 1: 矩阵遍历求和

令$\mathcal{W}_g = diag(\mathcal{W}_i)$，那么：

$$
\begin{aligned}
q^T \mathcal{W}_g q &= \textstyle\sum_i q_i^2 \mathcal{W}_i \\
&= \textstyle\sum_{i \in C_1} q_i^2 \mathcal{W}_i + \sum_{i \in C_2} q_i^2 \mathcal{W}_i \\
&= \textstyle\sum_{i \in C_1} \frac{\eta_2}{\eta_1} \mathcal{W}_i + \sum_{i \in C_2} \frac{\eta_1}{\eta_2} \mathcal{W}_i \\
&= \textstyle\frac{\eta_2}{\eta_1} \sum_{i \in C_1} \mathcal{W}_i + \frac{\eta_1}{\eta_2} \sum_{i \in C_2} \mathcal{W}_i \\
&= \textstyle\frac{\eta_2}{\eta_1} \eta_1 + \frac{\eta_1}{\eta_2} \eta_2 \\
&= \eta_1 + \eta_2
\end{aligned}
$$

所以我们的loss 函数变为：

$$\frac{q^T L q}{q^T \mathcal{W}_g q}$$

q本来是离散的，但是不好求解，所以我们放宽成任意实数:

$$\arg\min_{q \in R^n} \frac{q^T L q}{q^T \mathcal{W}_g q}$$

那么上述问题就是generalized Rayleigh quotient ，它最小化解就是广义特征值问题:

$$Lx = \lambda \mathcal{W}_g x$$

的最小特征值对应的特征向量。然而，我们可以发现$\mathbf{1}$是最小特征值$\lambda_1 = 0$对应的特征向量，但是这是一个平凡的划分，没有意义。所以我们求其第二小特征值对应的特征向量v_2，作为这个问题的放松解。按理说我们接着应该做一个如下的离散化从而得到我们的解：

$$q_i = \begin{cases} i \in C_1 & \text{if } v_2(i) \geq 0 \\ i \in C_2 & \text{if } v_2(i) < 0 \end{cases}$$

然而，实际中我们求第二小，第三小，$\ldots$,第$l+1$小特征值对应的特征向量，由此便形成node的长度为l的embedding，接着基于这个embedding 做k-means 便能得到一个节点的划分。对于多划分，推导和求解过程都是类似的。我们可以看到Spectral Clustering 也是先学习一个node embedding，然后基于node embedding 做聚类。

我们可以看到，这类算法首先在很强的限制下推导出一个漂亮的数学形式，然后说这么强的限制搞不定，让我们去掉这些限制吧，但是呢我们依然使用那个漂亮的数学形式，而且去掉这些限制后立马再来个新的数学理论来求解我们漂亮的数学形式，最后呢发现好像还是不行，那就再找点计算机同学们的东西做补丁，结果效果还不错，那就整件事就算妥了。这个套路还不算罕见。

我们再来介绍两个与graph partitioning 相关的概念。

设S是顶点集V的任意非空子集，他的boundary 被定义为：

$$\partial S = \{(i,j) | W_{ij} > 0, i \notin S, j \in S \text{ or } i \in S, j \notin S\}$$

也就是连接S和非S之间的边集(图2)。

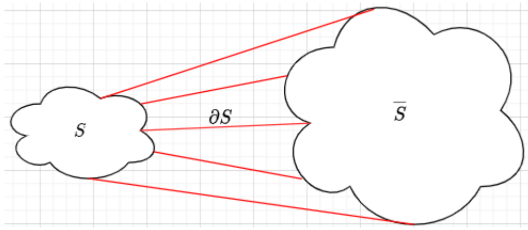


Figure 2: boundary示意图

S的isoperimetric ratio 定义为：

$$\theta(S)=\frac{|\partial S|}{|S|}$$

其中，$|\cdot|$表示集合的元素个数，这个表示了从图G中切除S损失的边的比例。图G 的isoperimetric number 为：

$$h(G)=\min_{|S|\leq n/2}\theta(S)$$

图的isoperimetric number 其实告诉我们把图做二划分(划分成$S,V-S$两部分)所损失的最小边数占比。关于isoperimetric number 的Cheeger 不等式为：

$$2h(G)\geq\lambda_2\geq\frac{h^2(G)}{2d_{max}(G)}$$

其中，$d_{max}(G)$表示G中节点度的最大值，λ_2是Laplacian矩阵的第二小特征值。图是连通的当且仅当isoperimetric number 为正。如果isoperimetric number 是较小，那么图中就存在两个顶点子集，他们之间通过少数几条边链接，这就是所谓的bottleneck，此时适合做图划分(在bottleneck 砍断，图(3))：

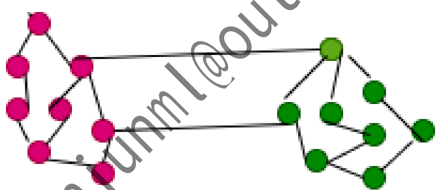


Figure 3: bottleneck示意图

图G 的conductance 被定义为：

$$\phi(G)=\min_{S\subset V}\phi(S)$$

其中

$$\phi(S)=\frac{|\partial S|}{\min\{d(S),d(V-S)\}},\quad d(S)=\sum_{i\in S}\sum_{j}W_{ij}=\sum_{i\in S}d_i$$

关于conductance 的Cheeger 不等式为：

$$\lambda_2^{norm}/2\leq\phi(G)\leq\sqrt{2\lambda_2^{norm}}$$

其中，λ_2^{norm} 是normalized Laplacian matrix 的第二小特征值。

注意，我们可以发现：$|\partial S|=cut(S,\overline{S})$，另外$d(S)$就是$weight(S)$，只是每个顶点是度(与normalize cut一样)。所以conductance的定义与下面表达式等价：

$$\phi(G)=\min_{S\subset V}\frac{|\partial S|}{\min\{d(S),d(\overline{S})\}}=\min_{S\subset V}\frac{cut(S,\overline{S})}{\min\{weight(S),weight(\overline{S})\}}$$

这个形式跟normalize cut很像，我们或许可以说conductance 的Cheeger 不等式告诉我们一个图做二划分能得到的最小损失的上下界，那么任何loss 大于上界的二划分都不是错误的。

Spectral Graph Theory 还有很多内容，我们这里仅仅涉及了其做图切割的部分知识。

2 信号处理初步

我们这里复习下信号变换。先看下三角函数：

$$y = A\sin(Bx + C)$$

，其振幅为A，周期为$\frac{2\pi}{B}$，频率为$\frac{B}{2\pi}$(单位Hz)，角频率为$\omega = B$，相移(相角)为$-\frac{C}{B}$，(见图(4))。

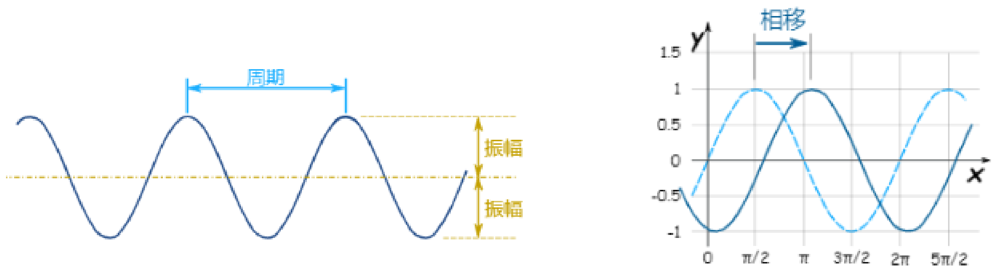


Figure 4: 振幅-周期-相移的示意图

三角函数$1, \sin x, \cos x, \sin 2x, \cos 2x, \ldots$是一个正交函数系，即他们中任何两个不同的成员的乘积在区间$[-\pi, \pi]$上的积分为0，比如：

$$\begin{aligned}
&\int_{-\pi}^{\pi} \sin(nx)\cos(mx)dx \\
&= \int_{-\pi}^{\pi} \frac{1}{2}\left(\cos((n+m)x)) + \cos((n-m)x)\right) dx \qquad \text{注意} \cos\alpha\cos\beta = \tfrac{1}{2}(\cos(\alpha+\beta) + \cos(\alpha-\beta)) \\
&= \frac{1}{2(n+m)}\left(\sin((n+m)x)) + \sin((n-m)x))\right)\Big|_{-\pi}^{\pi} = 0
\end{aligned}$$

傅里叶指出，任何周期函数可以用一系列简单的正弦、余弦波之和表示。比如我们可以通过对三角函数进行叠加得到一个方波(公式为(2，是一个傅里叶级数)，示意图为图(5))。

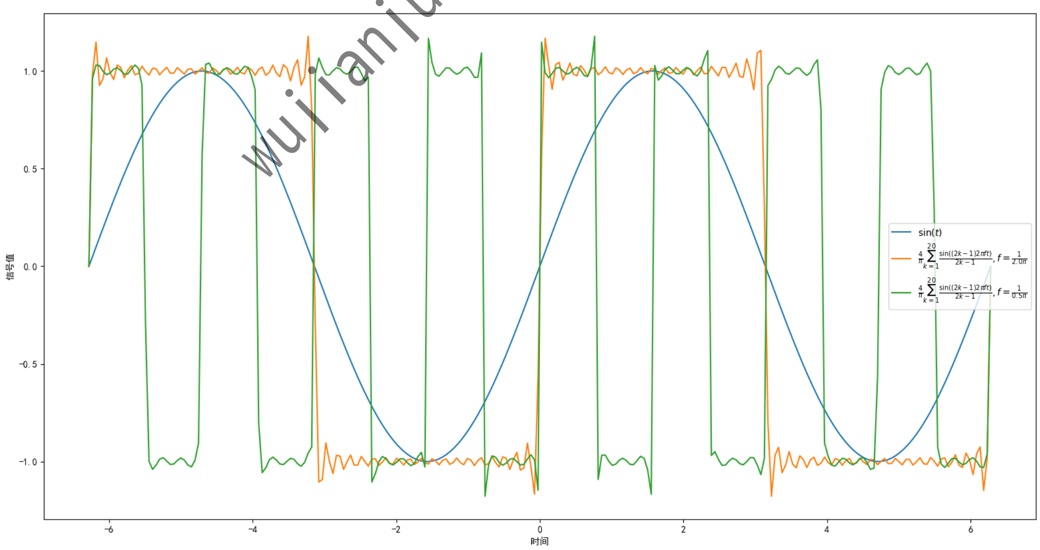


Figure 5: 三角函数叠加得到方波

$$x(t) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin((2k-1)2\pi f t)}{2k-1} = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{\sin(\omega_k t)}{2k-1} \tag{2}$$

我们把累加的每个正弦波叫做频率分量，可以看见其振幅越来越小，频率越来越高。

再看看复数的基本概念。任意一个复数可以表示成：

$$z = x + yi$$

，其中，x和y是实数，分别被称为实部x=real(z)和虚部y=imag(z)，且有$i^2 = -1$。复数有四则运算。另外还有几个重要的复数公式：

$$\cos\theta + i\sin\theta = e^{i\theta} \qquad \text{欧拉公式}$$
$$(\cos\theta + i\sin\theta)^n = \cos n\theta + i\sin n\theta$$

复数$z = a + ib$的共轭为：$\bar{z} = a - ib$，模为：$|z| = \sqrt{a^2 + b^2}$。一个复数也可以表示成指数形式：

$$z = re^{i\varphi} = r(\cos\varphi + i\sin\varphi)$$

，其中$r = |z|$是其模，φ是其幅角，另外两个复数之间的距离为：$d(z, w) = |z - w|$。一个复数$re^{i\varphi}$在复平面上表示见图(6)，φ变化时红色箭头对应地旋转。

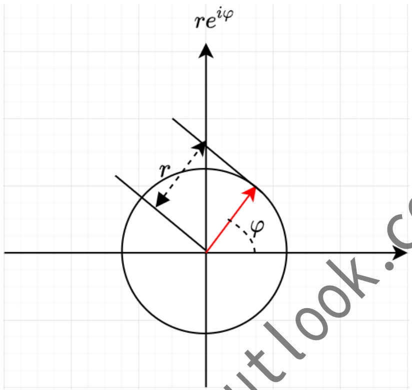


Figure 6: 复数的复平面表示

我们再复习下卷积的概念。两个$R \to R$函数的卷积仍然为一个$R \to R$函数，定义为：

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

两个数列的卷积为：

$$(f \star g)_n = \sum_{m=-\infty}^{\infty} f_m g_{n-m} \sum_{m=-\infty}^{\infty} f_{n-m} g_m$$

卷积满足很多性质，如：

$$f \star g = g \star f$$
$$f \star (g + h) = f \star g + f \star h$$
$$\frac{d}{dx}(f \star g) = \frac{df}{dx} \star g = \frac{dg}{dx} \star f$$

我们看看square-integrable 函数簇，其定义为：

$$L^2[a, b] = \{f : [a, b] \to R | \int_a^b f(x)dx < \infty \text{ 且 } \int_a^b f^2(x)dx < \infty\}$$

可以证明$L^2[a, b]$是线性空间，其内积定义为：

$$(f(x), g(x)) = \int_a^b f(x)g(x)dx$$

度量定义为：

$$||f(x) - g(x)||^2 = \int_a^b (f(x) - g(x))^2 dx$$

一个$R\to R$的函数$f(x)$的Fourier transform 为：

$$\mathcal{F}(f):\hat{f}(\xi)=\int_{-\infty}^{\infty}f(x)e^{-2\pi ix\xi}dx$$

其逆变换为：

$$f(x)=\int_{-\infty}^{\infty}\hat{f}(\xi)e^{2\pi ix\xi}d\xi$$

根据欧拉公式有,我们可以得到：

$$\int_{-\infty}^{\infty}f(x)e^{-2\pi ix\xi}dx=\int_{-\infty}^{\infty}f(x)\cos(2\pi x\xi)dx-i\int_{-\infty}^{\infty}f(x)\sin(2\pi x\xi)dx$$

所以经过Fourier transform 后我们得到的是一个复数。我们说x代表一个时刻，$f(x)$表示这个时刻的信号值，ξ代表频率，$|\hat{f}(\xi)|$代表该频率下的振幅。Fourier变换与卷积(用$\star$表示)之间的关系为：

$$\widehat{f\star g}=\hat{f}\odot\hat{g},\quad \text{或者}\quad \mathcal{F}(f\star g)=\mathcal{F}(f)\odot\mathcal{F}(g) \tag{3}$$

一个离散实数序列$\{x_n\}_{n=0}^{N-1}$的Fourier变换为：

$$y_k=\sum_{n=0}^{N-1}x_ne^{-2\pi i\frac{n}{N}k},\quad 0\le k\le N-1$$

可以看出$y_0=\sum_{n=0}^{N-1}x_n$，当N为偶数时，$y[1]$到$y[N/2-1]$(递减地)包含正频率项，$y[N/2+1]$到$y[N-1]$(递增地)包含负频率项；且$y[k]$与$y[N-k]$共轭，其中$1\le k\le N/2-1$；我们在绘制Fourier 变换结果时一般只画正频率部分；对于正频率项y_k，它对应的振幅为$|y_k|$，对应的相位为$\arctan\frac{imag(y_k)}{real(y_k)}$，而对应的频率为：$\frac{k}{TN}$，其中$T$表示信号在时域上的采样间隔(必须是均匀采样)。我们现在看看方波经过离散Fourier变换后的图像(图7)。

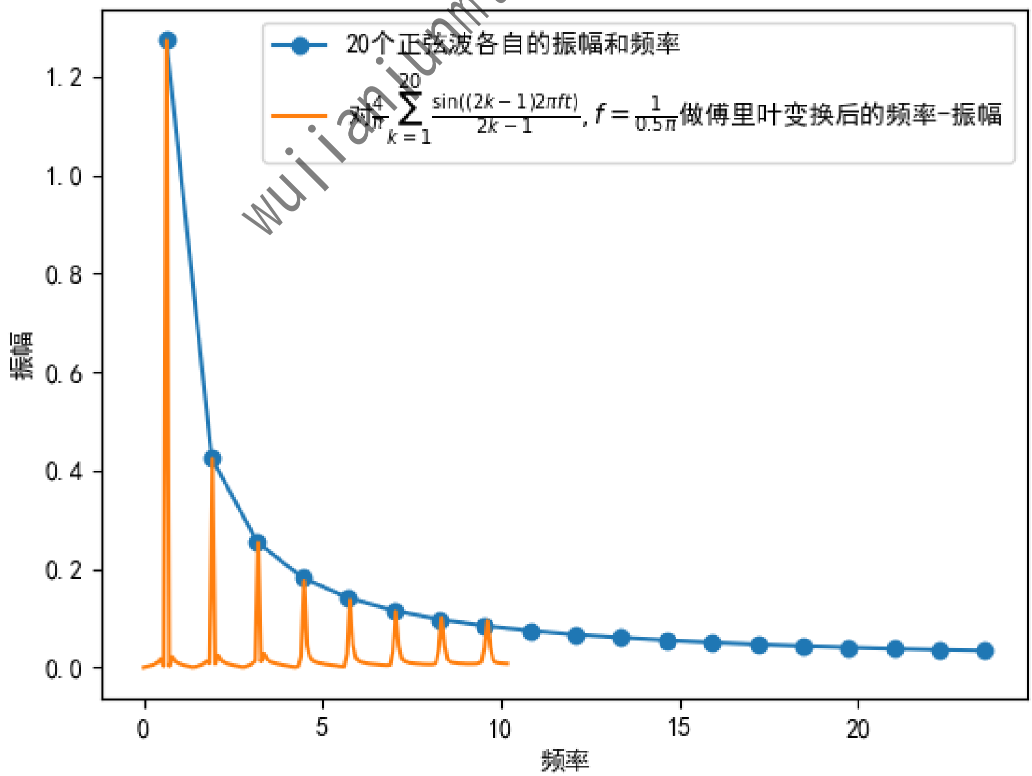


Figure 7: Fourier transform效果图

图5和图7的绘制代码如下：

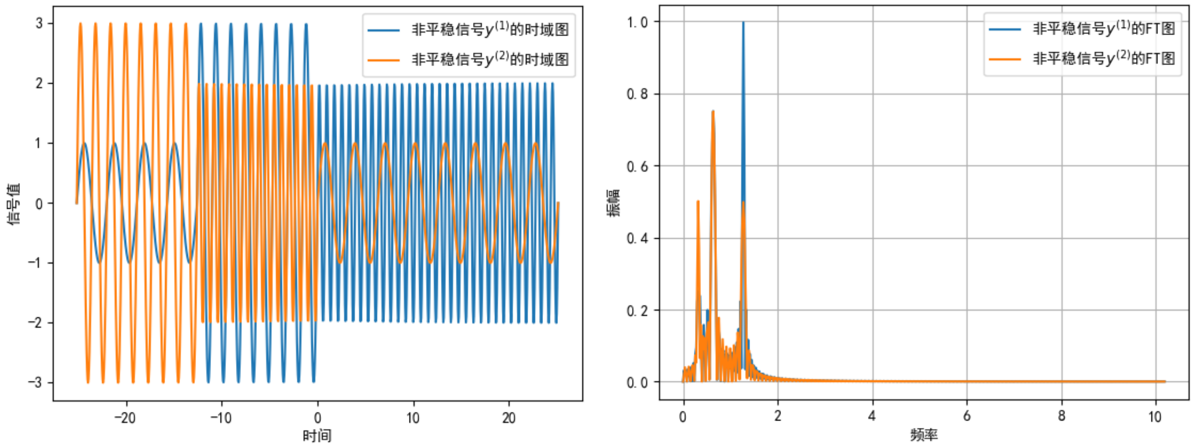
```
t_num = 256 # 时间轴上的点数
t = np.linspace(-2 * np.pi, 2 * np.pi, t_num, endpoint=True) # 时间轴
T = 4 * np.pi / t_num # 时间轴上点的间隔长度

# 方波生成函数
def sin_square(f):
    y = np.zeros(t_num)
    yf = []  # 振幅
    xf = []  # 频率
    for k in range(1, 20, 1):
        b = 4.0 / np.pi * np.sin((2 * k - 1) * 2 * np.pi * f * t) / (2 * k - 1) # 第k个正弦波
        yf.append(4.0 / np.pi / (2 * k - 1)) # 第k个振幅
        xf.append((2 * k - 1) * f)  # 第k个频率
        y = b + y # 累加第k个正弦波
    return y, yf, xf
y1 = np.sin(t)
plt.plot(t, y1, label=r"$\sin(t)$")
y2, yf2, xf2 = sin_square(f=1.0 / (2.0 * np.pi))
plt.plot(t, y2, label=latex_sin_square + r"$,f=\frac{1}{2.0\pi}$")
y3, yf3, xf3 = sin_square(f=1.0 / (0.5 * np.pi))
plt.plot(t, y3, label=latex_sin_square + r"$,f=\frac{1}{0.5\pi}$")
plt.xlabel('时间')
plt.ylabel('信号值')
plt.legend()
plt.show()

label=r"对$\frac{4}{\pi}\sum_{k=1}^{20}\frac{\sin((2k-1)2\pi f t)}{2k-1}$," r"$f=\frac{1}{0.5\pi}$做傅里叶变换后的频率-振幅"
plt.plot(xf3, yf3, 'o-', label=r"20个正弦波各自的振幅和频率")
yf = fft(y3)  # 对方波做傅里叶变换
xf = np.linspace(0.0, 1.0 / (2.0 * T), t_num // 2)
plt.plot(xf, 2.0 / t_num * np.abs(yf[:t_num // 2]), label=label)
plt.xlabel('频率')
plt.ylabel('振幅')
plt.legend()
plt.show()
```

我们从图7中看到黄色的线中有8个突起的点，而黄线其他部分非常接近0，这8个点正好对应我们原始信号中前面8个正弦波。我们需要指出我们的方波是一个所谓的stationary signal，也就是其频率是不变的，我们现在看看傅里叶变换对nonstationary (频率随时间变化)的信号的效果，例如：

$$y_t^{(1)} = \begin{cases} \sin(2t) & 0 \leq t < T/4 \\ 3\sin(4t) & T/4 \leq t < T/2 \\ 2\sin(8t) & otherwise \end{cases}, \quad y_t^{(2)} = \begin{cases} 3\sin(4t) & 0 \leq t < T/4 \\ 2\sin(8t) & T/4 \leq t < T/2 \\ \sin(2t) & otherwise \end{cases} \tag{4}$$

这两个信号按时间分成3段，每段的频率不一样，但是两个信号含有3个同样的频率分量，只是出现的时间不同。从图8中可以看出Fourier变换后二者几乎完全重合，无法判断每个频率分量出现的时间。

傅里叶变换只能算出信号包含哪些频率成分，没办法得知信号在不同时间的频率成分，不适合用来分析一个频率会随着时间而改变的信号。而短时傅里叶变换(Short-time Fourier Transform, STFT)把整个时域分成无数个等长的窗口，每个时间窗口内近似平稳，再做傅里叶变换，就知道在哪个时间点上出现什么频率了。STFT需要我们固定时间窗口大小，但是窗太窄，窗内的信号太短，会导致频率分析不够精准，频率分辨率差。窗太宽，时域上又不够精细，时间分辨率低。基于此，小波变换(wavelet transform)被提出。



(a) 原始非平稳信号　　(b) Fourier变换的频率分布

Figure 8: 非平稳信号的Fourier变换

小波变换用小波基(常用符号ψ表示，$\bar{\psi}(z)=\psi(\bar{z})$表示复数共轭)来表示信号，如下：

$$X(a,\tau)=\frac{1}{\sqrt{a}}\int_{-\infty}^{\infty}f(x)\bar{\psi}(\frac{x-\tau}{a})dx$$

$f(x)$是原始的时域信号，$X(a,\tau)$就是变化后的coefficients。小波做的改变就在于，将Fourier 里面无限长的三角函数基$e^{-2\pi i x\xi}$ 换成了有限长的且会衰减的小波基$\psi(\frac{x-\tau}{a})$。不同于三角函数基中只有频率ξ一个参数，小波基则有两个参数：尺度a和平移τ，尺度a与频率成反比，平移τ对应时间，我们得到的结果也由Fourier 的$(\xi,|\hat{f}(\xi)|)$变成小波的$(a,\tau,|X(a,\tau)|)$，由二维变成了三维，增加了一个时间维。 任何函数只要满足几个条件(如能量有限，零均值)就可以成为小波基，比如最简单的Haar 小波基(也称为db1)的定义和其图像如下(图9)：

$$\psi(t)=\begin{cases}1 & 0\le t<1/2\\ -1 & 1/2\le t<1\\ 0 & \text{otherwise}\end{cases}$$

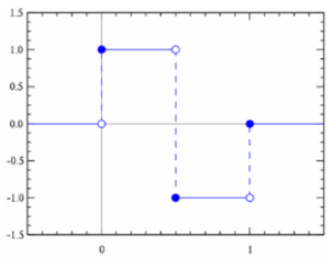


Figure 9: Haar小波

Morlet小波基的定义为：

$$\psi(t)=e^{-t^2/2}\cos(5t)$$

Morlet小波基的图形见图10(来源)。可以看到正弦波是全局震荡的，Morlet则是局部震荡的。

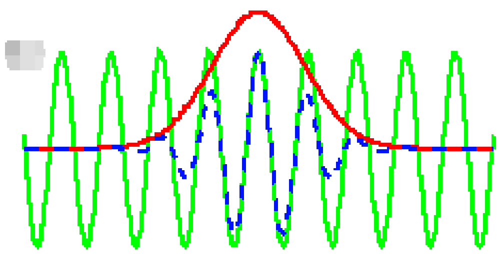


Figure 10: morlet小波基(蓝色虚线表示morlet，绿色表示sin，红色表示Gaussian。)

$\arg\max_t(\psi(t) \neq 0) - \arg\min_t(\psi(t) \neq 0)$称为窗口长度，也即是非0的取值范围。从图10中可以看到Morlet小波基的窗口。我们首先把原始信号跟我们的小波相乘，此时只有窗口内的信号被保留，其他时间点的信号因为与0相等变成0，故而被我们扔掉了，得到的结果就是当前时间窗口内的原始信号与当前(频率的)小波的重合度(图11)：

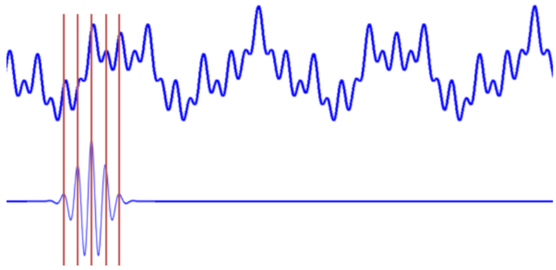


Figure 11: 小波乘法示意图

然后我们对小波做一个位移后再与原始信号相乘，此时我们的(时间)窗口移动了；接着我们改变小波基的scale，也就是频率，重复这个过程。于是我们得到了原始信号在不同时间段内不同频率的大小。scale与频率的转换公式为：

$$f = \frac{f_c}{a}$$

其中f是频率，f_c是小波的中心频率，每个小波都有中心频率，其计算比较复杂，不过很多程序包实现此计算，比如pywt(这是一个开源的python接口的wavelet变换包) 中的scale2frequency 函数用于计算指定小波基的不同尺度对应的频率。

[3]是一个优秀的wavelet博客，这里有一个关于小波的全面细致的课程。近年来，出现了小波变换与deep learning结合起来的工作 [4,5]，后续进一步研究他们。

小波变换是20世纪最辉煌科学成就之一，也是一个博大精深的领域，涉及很多概念。Continuous Wavelet Transform (CWT)中小波基的尺度和平移两个参数可以取任意值，Discrete Wavelet Transform (DWT) 中尺度和平移两个参数只能取离散的值(比如尺度取2的幂2^j，平移取整数)。CWT中常用的小波基有Morlet, Meyer, Mexican Hat等，而DWT中常用的小波基有Haar, Daubechies等。

对于Daubechies 小波，$\psi(t)$也被称为mother wavelet，它还有一个尺度函数$\phi(t)$(也称之为father wavelet)的概念用于计算尺度。对于满足一些条件的尺度函数ϕ，

$$\{\ \phi_{jk}(t) = 2^{j/2}\phi(2^j t - k), j, k \in Z\ \}$$

是一个正交函数簇，且每个函数$f(t) \in L^2[-\infty, \infty]$都可以被他们展开称小波级数(wavelet series)：

$$f(t) = \sum_j \sum_k c_{jk}\phi_{jk}(t)$$

。可以由尺度函数ϕ经过一系列计算步骤构造出小波ψ，被构造出的ψ被称为正交小波，而

$$\{\ \phi_{jk}(t), j, k \in Z\ \}$$

称为$L^2[-\infty, \infty]$的正交小波基。小波的逆变换为：

$$f(x) = \frac{1}{C_\psi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(a, \tau)\frac{1}{\sqrt{a}}\tilde{\psi}(\frac{x - \tau}{a})d\tau\frac{1}{a^2}da$$

其中，$\tilde{\psi}$是ψ的对偶小波，对偶小波又是一个不容易理解的概念。

我们看看式(4)两个信号经过小波变换后的图像(图12)。 可以看到二者有明显的区别。图12的绘

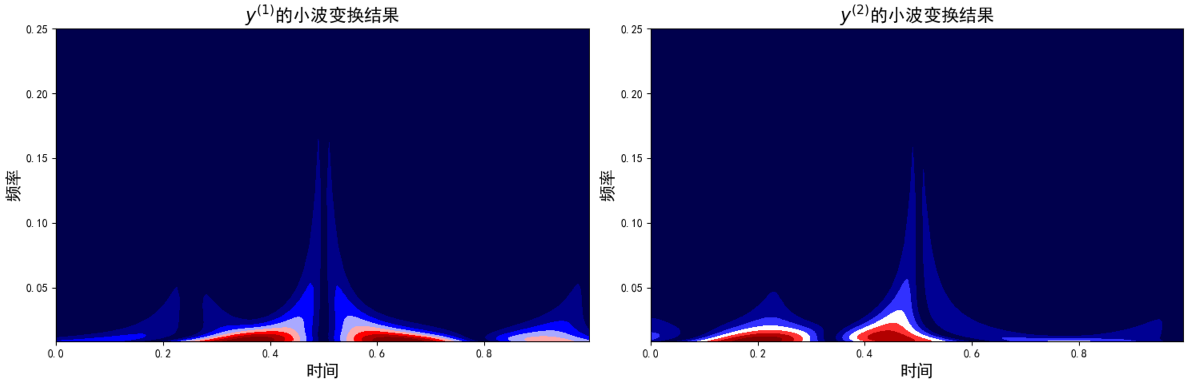


Figure 12: 非平稳信号的小波变换

制代码如下(顺便演示了pywt的基本用法)：

```
t = np.linspace(0, 1, 200, endpoint=False)
dt = 1 - 0 # 采样的时间跨度
y1 = nonstationary1(t) # 生成y^(1)
y2 = nonstationary2(t) # 生成y^(2)

# sig:是原始信号
# widths:是给定的一组scale 参数，例如: widths = np.arange(1, 31)
# 返回值为:
# cwtmatr: 每个scale 下每个时间点的变换后的系数，也就是小波变换积分的结果(可以是一个复数)
# freqs: 每个scale 对应当前小波基的中心频率(把scale转为频率)
# cwtmatr, freqs = pywt.cwt(sig, widths, 'morl')

widths = np.arange(1, 31)
wavelet=pywt.ContinuousWavelet('mexh')
cwtmatr1, freqs1 = pywt.cwt(y1, widths, wavelet, dt)
cwtmatr2, freqs2 = pywt.cwt(y2, widths, wavelet, dt)
# 注意, 我们这里积分结果全是实数，没有复数，所以实数本身就是振幅了。

# 纵坐标为频率(scale), 横坐标为时间，像素值为振幅
fig, axarr = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))
axarr[0].contourf(t, freqs1, np.abs(cwtmatr1), extend='both', cmap=plt.cm.seismic)
axarr[0].set_title(r"$y^{(1)}$的小波变换结果", fontsize=16)
axarr[0].set_ylabel("频率", fontsize=15)
axarr[0].set_xlabel("时间", fontsize=15)

axarr[1].contourf(t, freqs2, np.abs(cwtmatr2), extend='both', cmap=plt.cm.seismic)
axarr[1].set_title(r"$y^{(2)}$的小波变换结果", fontsize=16)
axarr[1].set_ylabel("频率", fontsize=15)
axarr[1].set_xlabel("时间", fontsize=15)

plt.tight_layout()
plt.show()
```

3 Graph Signal Transformation

我们先介绍一个新的函数逼近方法。先看：

$$
\begin{aligned}
\cos(\theta) &= \cos(\theta) \\
\cos(2\theta) &= 2\cos^2(\theta) - 1 \\
\cos(3\theta) &= 4\cos^3(\theta) - 3\cos(\theta) \\
\cos(4\theta) &= 8\cos^4(\theta) - 8\cos^2(\theta) + 1
\end{aligned}
$$

于是我们猜测$\cos(n\theta)$是$\cos(\theta)$的n次多项式，其实$\cos(n\theta) = T_n(\cos(\theta))$，也可以写成$T_n(\theta) = \cos(n \arccos(\theta))$。这里的$T_n(x)$就是一个$x$的$n$次多项式，称为(第一类)Chebyshev(切比雪夫) polynomials。举例如下：

$$
\begin{aligned}
T_0 &= 1 \\
T_1 &= x \\
T_2(x) &= 2x^2 - 1 \\
T_3(x) &= 4x^3 - 3x \\
T_4(x) &= 8x^4 - 8x^2 + 1 \\
T_5(x) &= 16x^5 - 20x^3 + 5x
\end{aligned}
$$

其递推式如下：

$$
T_0 = 1, T_1 = x, T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \tag{5}
$$

可以看出$T_n(x)$中，x^n的系数为2^{n-1}。因为：

$$
\int_{-1}^{1} \frac{1}{\sqrt{(1-x^2)}} T_n(x) T_m(x) dx = \begin{cases} 0 & m \neq n \\ \pi/2 & m = n \neq 0 \\ \pi & m = n = 0 \end{cases}
$$

所以$\{T_k(x)\}$是一组正交多项式(细节见这里)。切比雪夫多项式还有很多其他性质。它还满足如下重要定理(细节见此教程的第17-24页):

Theorem 3.1 *设$p(x)$是任意首项系数为1的k此多项式，令*

$$
\widetilde{T}_k(x) = \frac{T_k(x)}{2^{k-1}}
$$

那么：

$$
\max_{x \in [-1,1]} |\widetilde{T}_k(x)| \leq \max_{x \in [-1,1]} |p(x)|, \ for \ \forall p(x)
$$

由定理3.1可以推论出：任意k阶多项式$f_k(x)$，其首项系数为a_k，它的最佳$k-1$阶多项式逼近为：

$$
f_k(x) - a_k \widetilde{T}_k(x)
$$

函数$f(x) \in L^2[-1,1]$可以做如下展开：

$$
f(x) = \frac{a_0}{2} + \sum_{k=1} a_k T_k(x), x \in [-1,1]
$$

我们现在总结一下Chebyshev 多项式的优点：

- 它是近似的Minimax 多项式(图13)，而Minimax 多项式是任何函数的最佳多项式近似。
- 它有递归式，无需解析式就可以被高效地计算，也无需做指数乘法(这存在数值稳定性问题)。

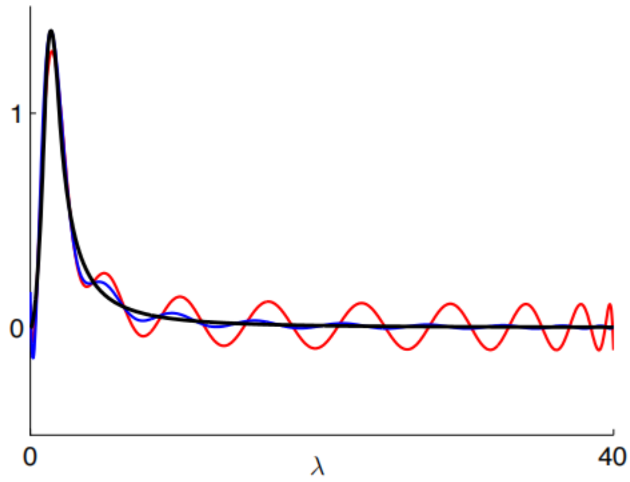


Figure 13: Chebyshev多项式对小波基的近似。黑色表示真实的小波基的图像，蓝色的是Chebyshev多项式，红色的是Minimax多项式近似。图片来自 [6]。

设多项式

$$g(x) = \sum_{n=0} a_n x^n$$

那么，对于任意对角矩阵

$$\Lambda = \begin{bmatrix} \ddots & & \\ & \lambda_l & \\ & & \ddots \end{bmatrix}$$

，我们记：

$$g(\Lambda) = \begin{bmatrix} \ddots & & \\ & \sum_{n=0} a_n \lambda_l^n & \\ & & \ddots \end{bmatrix} = \sum_{n=0} a_n \Lambda^n$$

对于Chebyshev 多项式我们有：

$$\begin{aligned} T_0(\Lambda) &= I \\ T_1(\Lambda) &= \Lambda \\ T_2(\Lambda) &= 2\Lambda^2 - I \\ &\vdots \end{aligned}$$

此时其地推式为:

$$T_k(\Lambda) = 2\Lambda T_{k-1}(\Lambda) - T_{k-2}(\Lambda)$$

这里是一个关乎Chebyshev 多项式的不错博客。

一个graph signal是一个把每个顶点映射成一个实数的函数$f : V \to R$，$f(i)$表示顶点i映射的值。graph Fourier transform (GFT) 把一个graph signal f变成如下形式：

$$\mathcal{GF}[f] : \hat{f}_l = \hat{f}(\lambda_l) = \sum_i f(i)\boldsymbol{v}_l(i) \tag{6}$$

其中，λ_l是图的Laplacian matrix的第l小特征值，$\boldsymbol{v}_l$是其对应的特征向量。可以看出$\hat{f}$是一个长度为n(图的顶点个数)的向量，也就是输入一个顶点对应的向量，GFT输出一个新的顶点对应的向量：$f \to \hat{f}$，而这个计算过程用到Laplacian matrix 的特征向量。我们说f在vertex domain，而$\hat{f}$在spectral domain。

我们这里再看下Fourier公式：

$$y_k=\sum_{n=0}^{N-1} f_n e^{-2\pi i \frac{n}{N} k} \Rightarrow \hat{f}(\frac{Nl}{2\pi}))=\sum_{n=0}^{N-1} f_n e^{-iln} \quad (令 2\pi\frac{k}{N}=l)$$

我们把他与GFT公式放在一起：

$$\begin{array}{l} \sum_{n=0}^{N-1} f_n e^{-iln} \\ \sum_{n=0}^{N-1} f_n \boldsymbol{v}_l(n) \end{array}$$

我们可以看出，他们非常类似，这里最关键的是把基从三角函数(欧拉公式告诉我们e^{-iln}可以变成三角函数)变成了Laplacian matrix 的特征向量。当然，我们的新基也是相互正交的。我们把λ_l也称为频率，$\hat{f}(\lambda_l)$表示频率为λ_l的波的振幅。 inverse graph Fourier transform (IGFT) 定义如下：

$$\mathcal{IGF}[f]: f_i=\sum_l \hat{f}(\lambda_l)\boldsymbol{v}_l(i)$$

用向量表示则为：

$$\begin{array}{ll} \mathcal{GF}[\boldsymbol{f}]: & \hat{\boldsymbol{f}}=\boldsymbol{V}^T\boldsymbol{f} \\ \mathcal{IGF}[\boldsymbol{f}]: & \boldsymbol{f}=\boldsymbol{V}\hat{\boldsymbol{f}} \end{array} \tag{7}$$

其中$\boldsymbol{V}=[\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_n]\in R^{n\times n}$是特征向量矩阵，一列一个特征向量。我们由3可以类推：

$$\mathcal{GF}(f\star g)=\mathcal{GF}(f)\odot\mathcal{GF}(g)$$

进而：

$$f\star g=\mathcal{IGF}(\mathcal{GF}(f)\odot\mathcal{GF}(g))$$

于是：

$$\begin{array}{l} [f\star g]_i=[\mathcal{IGF}(\mathcal{GF}(f)\odot\mathcal{GF}(g))]_i \\ =\sum_l[\mathcal{GF}(f)\odot\mathcal{GF}(g)]_l\boldsymbol{v}_l(i) \\ =\sum_l[\mathcal{GF}(f)]_l[\mathcal{GF}(g)]_l\boldsymbol{v}_l(i) \\ =\sum_l\hat{f}(\lambda_l)\hat{g}(\lambda_l)\boldsymbol{v}_l(i) \end{array}$$

这样我们就得到了两个graph signal之间做卷积的定义，常见卷积定义的几个常见性质对于这样定义的卷积同样成立。同样地，两个graph signal的卷积也还是一个graph signal。我们可以认为f是原始的graph signal，而g是一个filter。用向量表示则为：

$$\begin{array}{ll} \boldsymbol{f}\star\boldsymbol{g} & \\ =\boldsymbol{V}(\boldsymbol{V}^T\boldsymbol{f}\odot\boldsymbol{V}^T\boldsymbol{g}) & \\ =\boldsymbol{V}(\boldsymbol{V}^T\boldsymbol{g}\odot\boldsymbol{V}^T\boldsymbol{f}) & \\ =\boldsymbol{V}(diag(\boldsymbol{V}^T\boldsymbol{g})\boldsymbol{V}^T\boldsymbol{f}) & \boldsymbol{a}\odot\boldsymbol{b}=diag(\boldsymbol{a})\boldsymbol{b} \\ =\boldsymbol{V}\boldsymbol{G}\boldsymbol{V}^T\boldsymbol{f} & 令\boldsymbol{G}=diag(\boldsymbol{V}^T\boldsymbol{g}) \end{array} \tag{8}$$

我们对g做GF，则：

$$\hat{g}=\boldsymbol{V}^T\boldsymbol{g} \quad \Rightarrow \quad \boldsymbol{G}=diag(\hat{g})$$

我们记符号:

$$\hat{g}(\Lambda)=diag(\hat{g})$$

所以用$\boldsymbol{g}$做filter对$\boldsymbol{f}$进行卷积，结果可以写为：

$$\boldsymbol{f}'=\boldsymbol{V}\hat{g}(\Lambda)\boldsymbol{V}^T\boldsymbol{f} \tag{9}$$

我们还记符号：

$$\hat{g}(L) = \boldsymbol{V}\hat{g}(\Lambda)\boldsymbol{V}^T \tag{10}$$

选定filter函数$\hat{g}$后，$\hat{g}(L)$完全由图的Laplacian矩阵决定，跟输入信号无关：

$$\boldsymbol{f}' = g(L)\boldsymbol{f}$$

我们在谱域选定一个filter函数$\hat{g}$，然后计算其在每个特征值处的取值$\hat{g}(\lambda_l)$，把这些取值排列成对角矩阵，就得到了$g(\Lambda)$，然后利用Laplacian 矩阵的特征向量矩阵，就可以计算出一个图信号被filter之后的结果了。 因为：

$$\begin{aligned} \mathcal{GF}[Lf] &= V^T L f && \text{GF的矩阵形式} \\ &= V^T V \Lambda V^T f && L\text{做对角化} \\ &= \Lambda V^T f && V\text{是正交矩阵} \\ &= \Lambda \hat{f} && \text{对}f\text{做GF} \end{aligned}$$

如果这个filter在spectral domain 是多项式的，也就是：

$$\hat{g}(\lambda_l) = \sum_{k=0} a_k \lambda_l^k \tag{11}$$

写成矩阵形式就是：

$$\hat{g}(\Lambda) = \sum_{k=0} a_k \Lambda^k$$

那么式9就可以写：

$$\begin{aligned} \boldsymbol{f}' &= \boldsymbol{V}(\textstyle\sum_{k=0} a_k \Lambda^k)\boldsymbol{V}^T\boldsymbol{f} \\ &= \textstyle\sum_{k=0} a_k \boldsymbol{V}\Lambda^k \boldsymbol{V}^T \boldsymbol{f} \\ &= \textstyle\sum_{k=0} a_k L^k \boldsymbol{f} \qquad L^k = (V\Lambda V^T)^k = V\Lambda V^T V \Lambda V^T \dots V\Lambda V^T = V\Lambda^k V^T \end{aligned} \tag{12}$$

注意，我们可以得到结论：如果filter是多项式，那么我们计算一个图信号的傅里叶变化不需要对Laplacian矩阵做特征分解了，只需要对Laplacian做乘法就可以了。我们现在需要做的就是找一个最好的多项式簇来近似任意一个filter函数。我们采用Chebyshev 多项式来近似filter函数。因为：

$$\lambda_l \in [0, \lambda_n] \Rightarrow \frac{\lambda_l}{\lambda_n/2} - 1 = \frac{2\lambda_l}{\lambda_n} - 1 \in [-1, 1]$$

从而，我们可以对$\hat{g}(\lambda_l)$做如下多项式展开：

$$\hat{g}(\lambda_l) = \frac{a_0}{2} + \sum_{k=1} a_k T_k\left(\frac{2\lambda_l}{\lambda_n} - 1\right), \lambda_l \in [0, \lambda_n] \tag{13}$$

其中，

$$a_k = \frac{2}{\pi}\int_0^\pi \cos(k\theta)\hat{g}\left(\frac{\lambda_n}{2}(\cos\theta + 1)\right) d\theta \tag{14}$$

我们可以看到只要给定$\hat{g}$，我们就可以计算出所有a_k(计算最大特征值λ_n有很快的算法)。我们令：

$$\overline{T}_k(\lambda_l) = T_k\left(\frac{2\lambda_l}{\lambda_n} - 1\right)$$

那么：

$$\hat{g}(\lambda_l) = \frac{a_0}{2} + \sum_{k=1} a_k \overline{T}_k(\lambda_l)), \lambda_l \in [0, \lambda_n]$$

写成矩阵形式则是：

$$\hat{g}(\Lambda) = \frac{a_0}{2} I + \sum_{k=1} a_k \overline{T}_k(\Lambda)), \lambda_l \in$$

根据式5有：

$$\overline{T}_k(\lambda_l) = 2(\frac{2\lambda_l}{\lambda_n} - 1)\overline{T}_{k-1}(\lambda_l) - \overline{T}_{k-2}(\lambda_l) \tag{15}$$

把$\hat{g}(\lambda_l)$的Chebyshev 多项式带入9有：

$$\begin{aligned}
f' &= V\hat{g}(\Lambda)V^T f \\
&= V\left(\tfrac{a_0}{2}I + \textstyle\sum_{k=1} a_k\overline{T}_k(\Lambda)\right)V^T f \\
&= V\tfrac{a_0}{2}V^T f + \textstyle\sum_{k=1} a_k V\overline{T}_k(\Lambda)V^T f \\
&= \tfrac{a_0}{2}f + \textstyle\sum_{k=1} a_k V\overline{T}_k(\Lambda)V^T f && V\text{是正交矩阵} \\
&= \tfrac{a_0}{2}If + \textstyle\sum_{k=1} a_k\overline{T}_k(L)f && \text{令}\overline{T}_k(L) = V\overline{T}_k(\Lambda)V^T
\end{aligned} \tag{16}$$

这里再啰嗦地说一下：

$$\overline{T}_k(\Lambda) = diag(\overline{T}_k(\lambda_l))$$

也就是计算每个特征值对应的T_k值$\overline{T}_k(\lambda_l)$，然后排成对角线得到的对角阵。因为：

$$\begin{aligned}
\overline{T}_k(L)f &= V\overline{T}_k(\Lambda)V^T f \\
&= V\left(2(\frac{2}{\lambda_n}\Lambda - I)\overline{T}_{k-1}(\Lambda) - \overline{T}_{k-2}(\Lambda)\right)V^T f && \text{带入式15} \\
&= V2(\frac{2}{\lambda_n}\Lambda - I)\overline{T}_{k-1}(\Lambda)V^T f - V\overline{T}_{k-2}(\Lambda)V^T f \\
&= 2(\frac{2}{\lambda_n}V\Lambda - V)\overline{T}_{k-1}(\Lambda)V^T f - \overline{T}_{k-2}(L)f && \text{带入}\overline{T}_{k-2}(L)\text{的定义} \\
&= 2(\frac{2}{\lambda_n}V\Lambda - V)V^T V\overline{T}_{k-1}(\Lambda)V^T f - \overline{T}_{k-2}(L)f && \text{V是正交矩阵} \\
&= 2(\frac{2}{\lambda_n}V\Lambda V^T - VV^T)V\overline{T}_{k-1}(\Lambda)V^T f - \overline{T}_{k-2}(L)f \\
&= 2(\frac{2}{\lambda_n}L - I)V\overline{T}_{k-1}(\Lambda)V^T f - \overline{T}_{k-2}(L)f \\
&= 2(\frac{2}{\lambda_n}L - I)\overline{T}_{k-1}(L)f - \overline{T}_{k-2}(L)f
\end{aligned} \tag{17}$$

另外，我们可以得到：

$$\overline{T}_0(L) = I, \overline{T}_1(L) = L$$

于是我们只要计算出Laplacian矩阵及其最大特征值λ_n，然后就可以根据式14计算出所有的a_k，根据式17计算每个$\overline{T}_k(L)f$(而且这个计算过程只有矩阵-向量乘法，会很快)，带入式16就可以得到f'。当然，我们实际中，只会计算前面M个Chebyshev 多项式。

我们再看看谱域上$\hat{\lambda}_l$是多项式时，卷积的含义：

$$\begin{aligned}
&\textstyle\sum_l \hat{f}(\lambda_l)\hat{g}(\lambda_l)\boldsymbol{v}_l(i) \\
&= \textstyle\sum_l\sum_j f(j)\boldsymbol{v}_l(j)\sum_k a_k\lambda_l^k\boldsymbol{v}_l(i) \quad \text{带入}\hat{f}(\lambda_l)\text{和式(11)} \\
&= \textstyle\sum_l\sum_j\sum_k f(j)\boldsymbol{v}_l(j)a_k\lambda_l^k\boldsymbol{v}_l(i) \\
&= \textstyle\sum_j\sum_k\sum_l f(j)\boldsymbol{v}_l(j)a_k\lambda_l^k\boldsymbol{v}_l(i) \\
&= \textstyle\sum_j f(j)\sum_k a_k\sum_l \lambda_l^k\boldsymbol{v}_l(j)\boldsymbol{v}_l(i)
\end{aligned}$$

注意，我们有$[L^k]_{ij} = \sum_l \lambda_l^k\boldsymbol{v}_l(j)\boldsymbol{v}_l(i)$($L$就是图的Laplacian 矩阵)，并且可以证明，

$$[L^k]_{ij} = 0 \Leftrightarrow \text{顶点}i\text{和顶点}j\text{之间的最短距离大于}k\text{。}$$

证明过程见 [6]的Lemma 5.4。 我们定义:

$$b_{ij} = \sum_k a_k[L^k]_{ij}$$

则

$$[f\star g]_i=\sum_{j\in N_k(i)}f(j)b_{ij}$$

$N_k(i)$表示顶点i的与之最短距离小于等于k的节点集合。所以我们可以得到一个重要的结论，当filter在spectral domain 是多项式时，那么一个graph signal 与之的卷积结果就是graph signal 本身的线性组合，且顶点i对应的结果信号是与之距离小于等于k的邻点们输入信号的线性组合。再定义graph 中的平移操作(也是把一个graph signal 变成另外一个graph signal)：

$$(T_jf)(i)=\sqrt{n}\sum_l\hat{f}(\lambda_l)\boldsymbol{v}_l(j)\boldsymbol{v}_l(i)$$

其他操作，如Dilation，Modulation也可以被定义(参考这个文档吧)，从而形成完整的Fourier操作。

[6]提出spectral graph wavelet transform (SGWT)，用于对graph signal 做小波变换，其定义为：

$$\mathcal{SGWT}(f):X_f(t,i)=\sum_l g(t\lambda_l)\hat{f}(\lambda_l)\boldsymbol{v}_l(i) \tag{18}$$

这个定义有两个特点：

- 先对graph signal 做个graph Fourier transform 得到$\hat{f}$。作者指出，这样做是因为在graph Fourier domain 可以定义尺度缩放操作(t是尺度参数)。
- g是一个$R\to R$的函数，是我们的小波基，$\boldsymbol{v}_l$则用来保持基的正交性。

其他符号同(式6)。我们同样可以采用Chebyshev 多项式来高精度地近似$\hat{f}(\lambda_l)$从而避免Laplacian矩阵的特征分解运算。式18用矩阵表示为：

$$\begin{aligned}X_f(t)=\;&V\begin{bmatrix}\ddots & & \\ & tg(\lambda_l) & \\ & & \ddots\end{bmatrix}\hat{f}\\ &=Vg(t\Lambda)\hat{f}\\ &=Vg(t\Lambda)V^Tf && \text{带入}\hat{f}\text{的矩阵形式}\end{aligned}$$

我们对$g(t\lambda_l)$做Chebyshev 多项式展开类似地有：

$$g(t\lambda_l)=\frac{a_{t,0}}{2}+\sum_{k=1}a_{t,k}\overline{T}_k(\lambda_l),\text{其中},a_{t,k}=\frac{2}{\pi}\int_0^\pi\cos(k\theta)g(t\frac{\lambda_l}{2}(\cos\theta+1))d\theta$$

类似地，我们有：

$$\begin{aligned}X_f(t)&=Vg(t\Lambda)V^Tf\\ &=V(\tfrac{a_{t,0}}{2}I+\textstyle\sum_{k=1}a_{t,k}\overline{T}_k(\Lambda))V^Tf\\ &=\tfrac{a_{t,0}}{2}VV^Tf+\textstyle\sum_{k=1}a_{t,k}V\overline{T}_k(\Lambda)V^Tf\\ &=\tfrac{a_{t,0}}{2}fI+\textstyle\sum_{k=1}a_{t,k}\overline{T}_k(L)f && V\text{是正交矩阵，且令}\overline{T}_k(L)=V\overline{T}_k(\Lambda)V^T\end{aligned}$$

后面的推导与GFT雷同。我们注意到了SGWT与GFT仅仅是多项式系数计算不一样而已，其他完全一样。另外我们也可以把Laplacian 矩阵换成其他任意对称矩阵，比如normalized Laplacian 矩阵。

当然还有其他方法定义图上的小波变换，如Diffusion wavelets [7]。这里有一个关于graph信号处理非常全面的课件。构造图上的信号变换最重要的数学工具是调和分析(Harmonic analysis)，也称为谐波分析。构造后的信号变换方法其理论性质的证明往往会借助调和分析。

4 Graph Convolutional Networks

我们考虑对无向图中的节点进行分类的问题。W_{ij}这里表示两点之间的边权，f代表神经网络，x_i表示顶点i的属性，$f(x_i)$表示顶点i的embedding。Laplacian regularization term 的定义为：

$$
\begin{aligned}
&\tfrac{1}{2}\textstyle\sum_{ij} W_{ij}||f(x_i)-f(x_j)||^2 & \\
&= \textstyle\sum_i f(x_i)^T f(x_i) D_{ii} - \sum_{ij} W_{ij} f(x_j)^T f(x_i) & D_{ii} = \textstyle\sum_j W_{ij} \text{且} W_{ij} = W_{ji} \\
&= \textstyle\sum_{ij} f(x_j)^T f(x_i) D_{ij} - \sum_{ij} W_{ij} f(x_j)^T f(x_i) & D_{ij} = 0, \forall i \neq j \\
&= \textstyle\sum_{ij} f(x_j)^T (D_{ij} - W_{ij}) f(x_i) & \\
&= \textstyle\sum_{ij} f(x_j)^T L_{ij} f(x_i) & L_{ij} = D_{ij} - W_{ij} \\
&= \textstyle\sum_i \sum_j \sum_k f(x_j)_k L_{ij} f(x_i)_k & \\
&= \textstyle\sum_i \sum_j \sum_k f(X)_{jk} L_{ij} f(X)_{ik} & f(X)_{ij} = f(x_i)_j \\
&= \textstyle\sum_i \sum_k f(X)_{ik} \sum_j L_{ij} f(X)_{jk} & L_{ji} = L_{ij} \\
&= \textstyle\sum_i \sum_k f(X)_{ik} (Lf(X))_{ik} & \\
&= \textstyle\sum_k \sum_i f(X)^T_{ki} (Lf(X))_{ik} & \\
&= \textstyle\sum_k (f(X)^T Lf(X))_{kk} & \\
&= trace(f(X)^T Lf(X)) &
\end{aligned}
$$

把这个term放进loss的使得边权越大的节点，他们的embedding越接近。如此我们就得到一个用神经网络的对节点做embedding的简单模型了，节点还可以带着各种属性。

根据式16，我们可以定义一中layer如下：

$$
\begin{aligned}
f^{l+1} &= \sigma(VGV^T f^l) \\
&= \sigma(V\hat{g}(\Lambda)V^T f^l) \\
&= \sigma(\textstyle\sum_{k=0} \theta_k T_k(L) f^l)
\end{aligned}
$$

其中的$\theta_k, k = 0, 1, \ldots$，就是这一层需要学习的参数。然后多堆叠几层，便是是一个图神经网络了。不过一般我们每层神经网络每个样本(顶点)对应一个向量，不同维度我们认为是不同的图信号，所以Spectral CNN [8]定义了graph convolutional layer，如下：

$$H^{l+1}_{:,j} = \sigma(\sum_i VG^{l+1}_{ij} V^T H^l_{:,i}), H^0 = X$$

其中，$H^l_{:,i}$表示输入矩阵的第i列，G^{l+1}_{ij}表示第$l+1$层的一组参数矩阵，这些参数矩阵是对角阵，他用来关联输入的第i列和输出的第j列。这里为什么要做累加$\sum_i(\cdot)$呢？我们最初想到的应该是这样：

$$H^{l+1}_{:,j} = \sigma(VG^{l+1}_j V^T H^l_{:,j}), H^0 = X$$

也就是每个维度单独做图傅里叶变换。这样做有个明显的缺点，那就是输入输出的shape必须一样，顶点embedding(H就是一个embedding)的各个列之间永远是平行的，没有信息融合。这里大牛们又告诉我们一招，如何优雅地更改shape。这种方法的一个问题就是V的计算成本很高。

为了减少每个图卷积层的参数从而使网络变得更深，Graph Convolutional Network (GCN) [11]只使用一阶Chebyshev多项式来近似卷积核且用normalized Laplacian 矩阵换掉Laplacian矩阵(此时$\lambda_n = 2$)，我们有：

$$\hat{g}(\Lambda) = \theta_0 I + \theta_1 \overline{T}_1(\Lambda) = \theta_0 I + \theta_1(\Lambda - I)$$

卷积变为：

$$
\begin{aligned}
f' &= \theta_0 f + \theta_1 L_{norm} f & \text{引用式16} \\
&= \theta_0 f + \theta_1 (I - D^{-1/2} W D^{-1/2}) f & \text{带入式1的第一行} \\
&= \theta'_0 f - \theta'_1 D^{-1/2} W D^{-1/2} f & \theta'_0 = \theta_0 + \theta_1, \theta'_1 = \theta_1 \\
&= \theta (I + D^{-1/2} W D^{-1/2}) f & \text{为减少参数量，我们强制}\theta = \theta'_0 = -\theta'_1
\end{aligned}
$$

为了防止梯度爆炸或消失，用下面的$\widetilde{W}$和$\widetilde{D}$分别替换W和D(原文没有对此详细解释，估计是实验发现的。这个操作其实就是每个顶点加了一个自连接，这样图中所有顶点的度就一定不会等于0，从而使得$\widetilde{D}^{-1/2}$一定存在)：

$$\widetilde{W} = W + I, \widetilde{D} = diag(\sum_j \widetilde{W}_{ij})$$

，然后图卷积操作再近似为(这一步无法数学推导，感觉是强行做的)：

$$f' = \theta \widetilde{D}^{-1/2} \widetilde{W} \widetilde{D}^{-1/2} f$$

因为神经网络中我们的信号是多维的(维度设为p)，并且我们在同一层会使用多个filter(设使用q个filter):

$$H' = \widetilde{D}^{-1/2} \widetilde{W} \widetilde{D}^{-1/2} H \begin{bmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1q} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2q} \\ \dots & \dots & \dots & \dots \\ \theta_{p1} & \theta_{p2} & \dots & \theta_{pq} \end{bmatrix} = \widetilde{D}^{-1/2} \widetilde{W} \widetilde{D}^{-1/2} H \Theta$$

所以最终第$l+1$层为：

$$H^{(l+1)} = \sigma(\widetilde{D}^{-1/2} \widetilde{W} \widetilde{D}^{-1/2} H^{(l)} \Theta^{(l)})$$

其中$\Theta^l \in R^{p \times q}$。另外，$H^{(0)} = X$。我们记

$$\widehat{W} = \widetilde{D}^{-1/2} \widetilde{W} \widetilde{D}^{-1/2}$$

一个两层的GCN网络可以表示为如下：

$$Z = softmax\left(\widehat{W} Relu\left(\widehat{W} X \Theta^{(0)}\right) \Theta^{(1)}\right)$$

我们采用梯度下降方法学习参数矩阵$\Theta^{(0)}, \Theta^{(1)}$。我们从这个式子中可以看出不同layer的使用的是同一个$\widehat{W}$，这就意味着不同layer的图结构是一样的，而图信号不一样。另外，特别注意，[11]在训练时，每轮迭代输入整个图，并没有采样一部分顶点这种mini-batch的方式。

ChebNet [9]使用Chebyshev多项式来近似卷积核，得到一个图卷积层(就是式16)。在卷积层之后，加了一个pooling层。pooling层首先对图做一个多层次聚类(采用的是Graclus multilevel clustering algorithm [10])。当我们想做大小为4的pooling时，我们就做两层聚类。每层每个类最多只允许有下一层次的两个节点，对于那些只有一个节点的类，我们造一个假节点给它(假节点的信号值为0)。如图14，$\mathcal{G}_0$只有8个节点：$(0,1,5,4,8,9,6,10)$，进行一次聚类后变成$(\{0,1\},\{4,5\},\{8,9\},\{6\},\{10\})$。加入两个假节点$\{\underline{7},\underline{11}\}$变成$(\{0,1\},\{4,5\},\{8,9\},\{6,\underline{7}\},\{10,\underline{11}\})$，我们把每个类里面每个成员的信号值加起来作为新的节点的信号值，由此得到了$\mathcal{G}_1 = (0,2,4,3,5)$，做一次大小为2的层次聚类，结果为：$(\{0\},\{2,3\},\{4,5\})$，加入假节点后变为：$(\{0,\underline{1}\},\{2,3\},\{4,5\})$，于是我们的到$\mathcal{G}_2$。然后我们建立图14后边那样的二叉树(注意因为$\mathcal{G}_1$中的假节点$\underline{1}$在$\mathcal{G}_0$中没有子节点所以我们用两个假节点放在$\mathcal{G}_1$中作为$\mathcal{G}_1$中的假节点$\underline{1}$的两个子节点。)我们对$\mathcal{G}_2$中的节点任意排序，这个排序可以传导到$\mathcal{G}_0$从而得到$\mathcal{G}_0$中节点的一个排序，然后我们在这个排序上做1Dpooling。例如:

$$\begin{aligned} z &= [\max(f_0, f_1, f_{\underline{2}}, f_{\underline{3}}), \max(f_4, f_5, f_6, f_{\underline{7}}), \max(f_8, f_9, f_{10}, f_{\underline{11}})] \\ &= [\max(f_0, f_1), \max(f_4, f_5, f_6), \max(f_8, f_9, f_{10})] \qquad \text{假节点的信号值是0} \end{aligned}$$

当然我看到这种pooling要先做层次聚类，这引入了计算代价，而且如何在mini-batch上做层次聚类也是个问题。

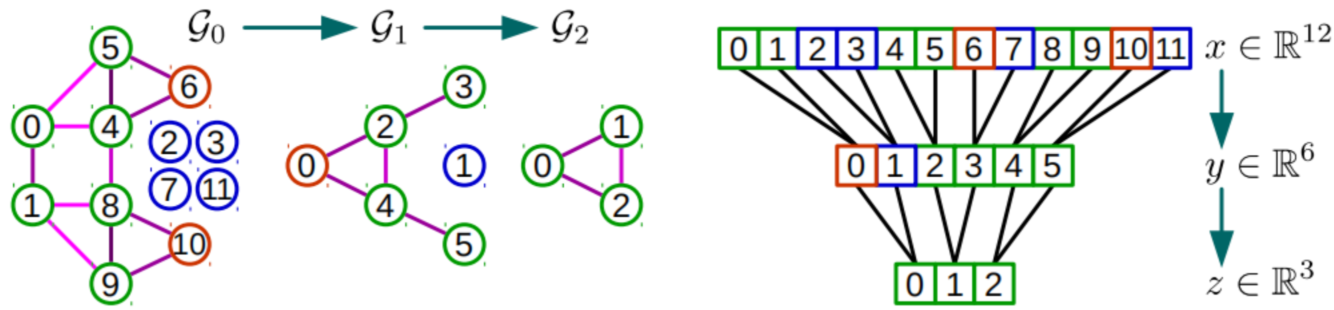


Figure 14: pooling操作示意图。蓝色的是假节点。

References

[1] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[2] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[3] A guide for using the wavelet transform in machine learning. Website, 2018. `https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/`.

[4] Raif M Rustamov and Leonidas J Guibas. Wavelets on graphs via deep learning. In *Vertex-Frequency Analysis of Graph Signals*, pages 207–222. Springer, 2019.

[5] Shin Fujieda, Kohei Takayama, and Toshiya Hachisuka. Wavelet convolutional neural networks. *arXiv preprint arXiv:1805.08620*, 2018.

[6] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[7] Ronald R Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[8] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[10] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.