# 图神经网络

# 目录

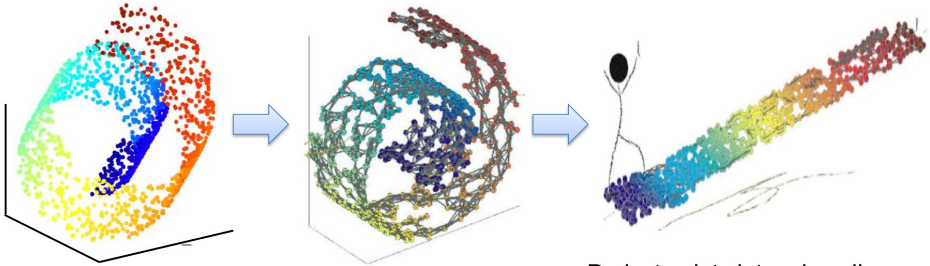
# 图表示的学习

数据降维

网络嵌入

图深度学习

Laplacian Eigenmap



Construct graph from data points (capture local information)

Project points into a low-dim space using “eigenvectors of the graph”

Image credit

1970s ~2000 Spectral Clustering

2000 IsoMap; LLE

2003 Laplacian Eigenmaps

2006 Matrix Factorization

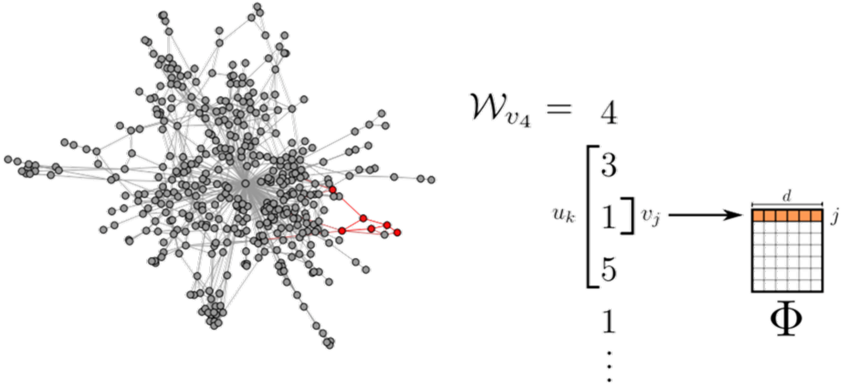
# 图表示的学习

数据降维

网络嵌入

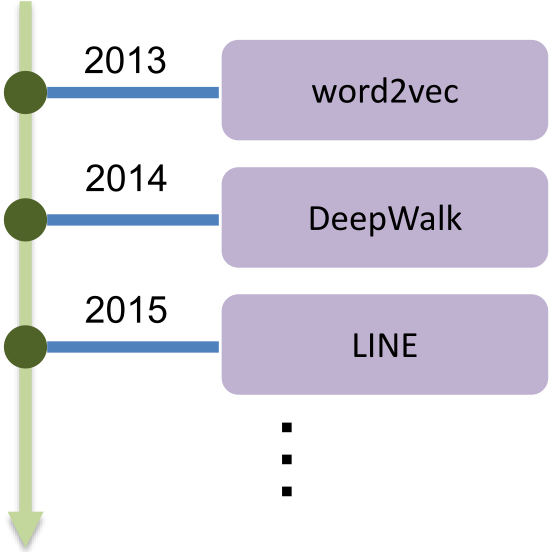
图深度学习

DeepWalk




(a) Random walk generation. (b) Representation mapping.

Image credit



# 图表示的学习

数据降维

网络嵌入

图深度学习

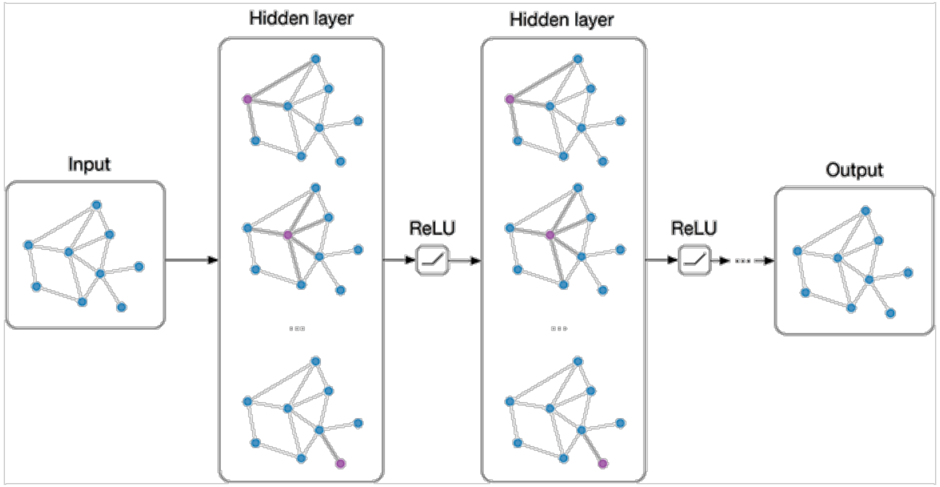
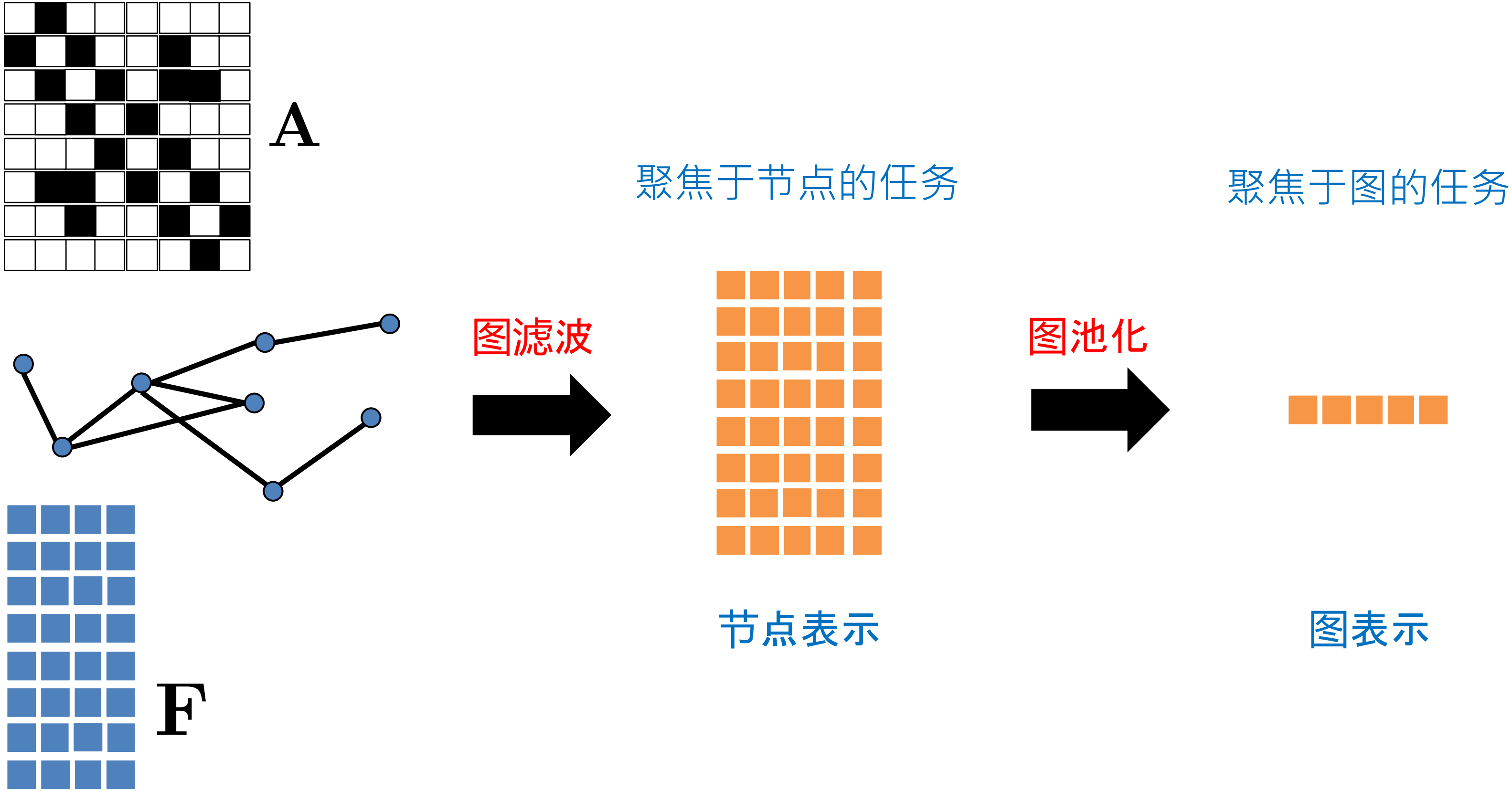



Image credit

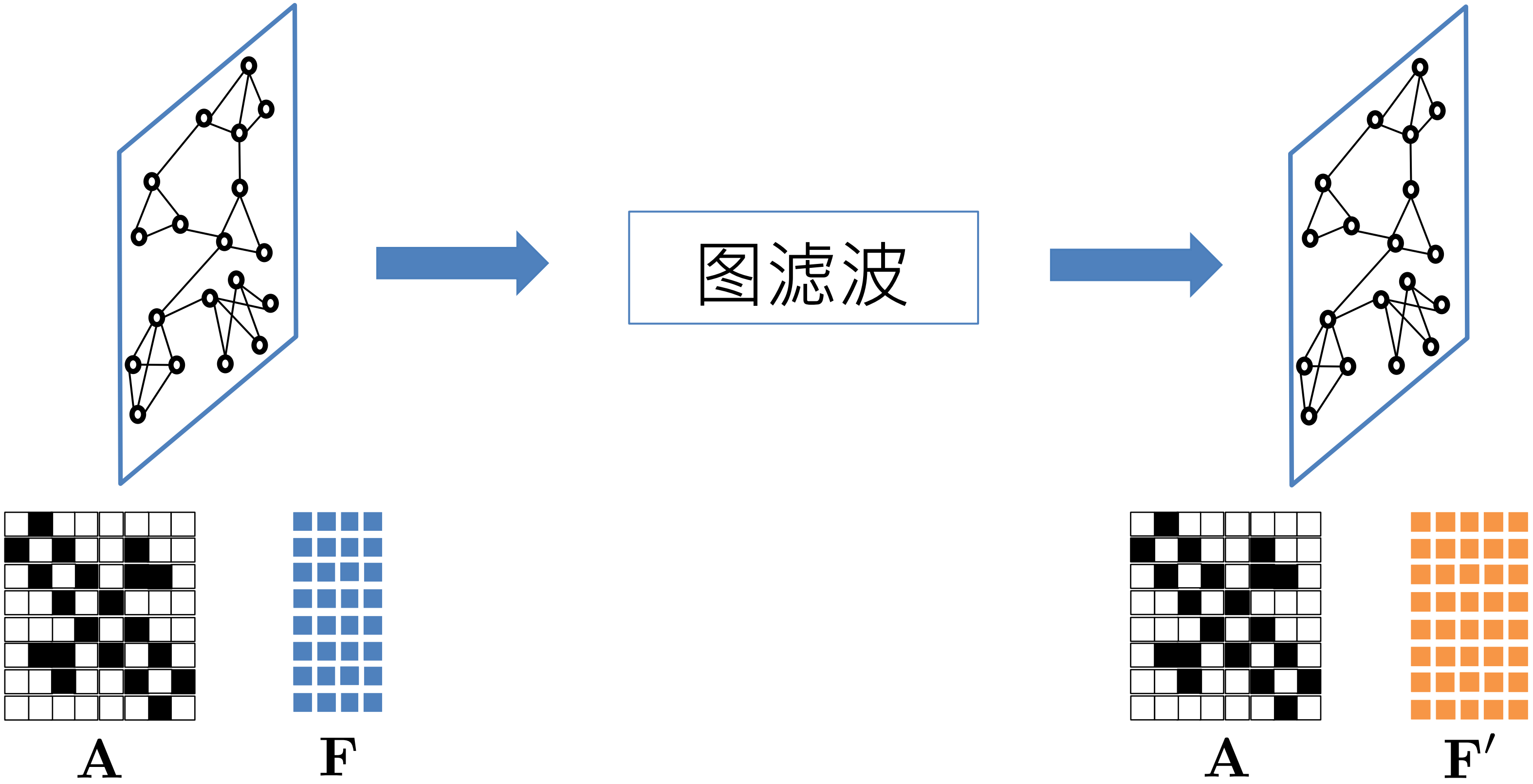
今天的主题

构建适合于图的深度神经网络！

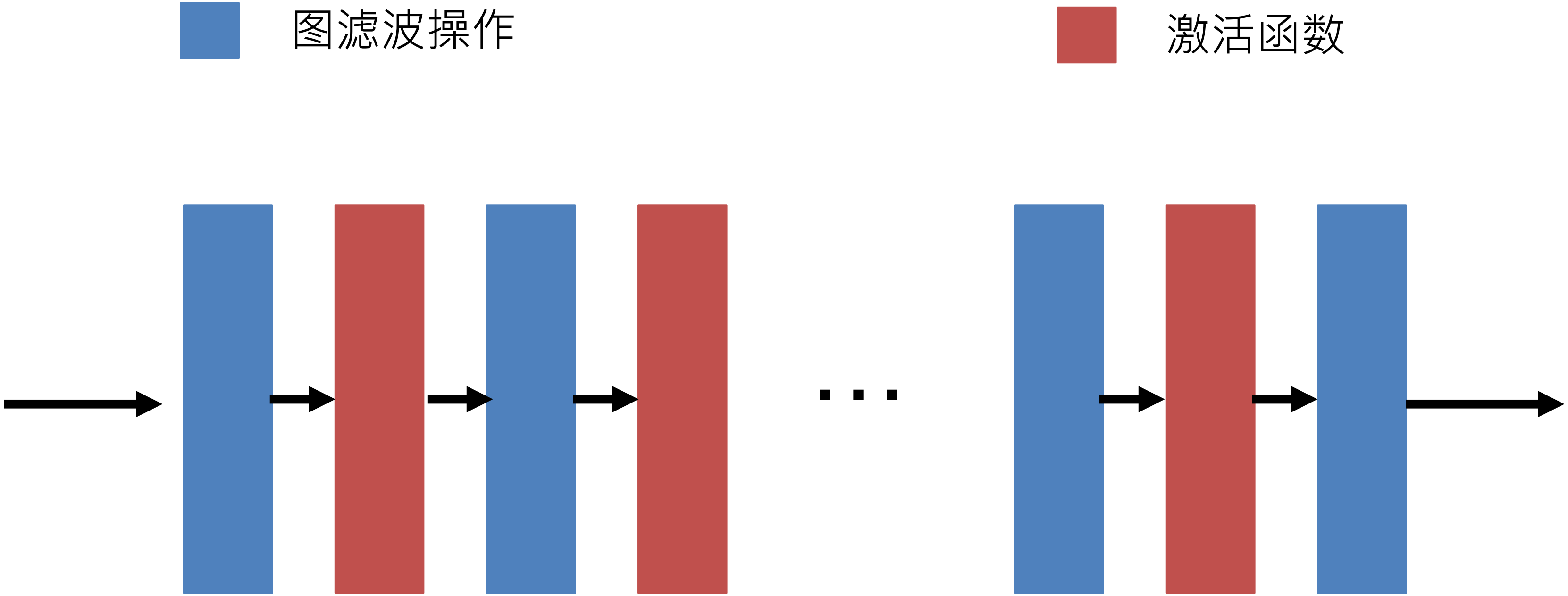
# 图神经网络（GNNs）



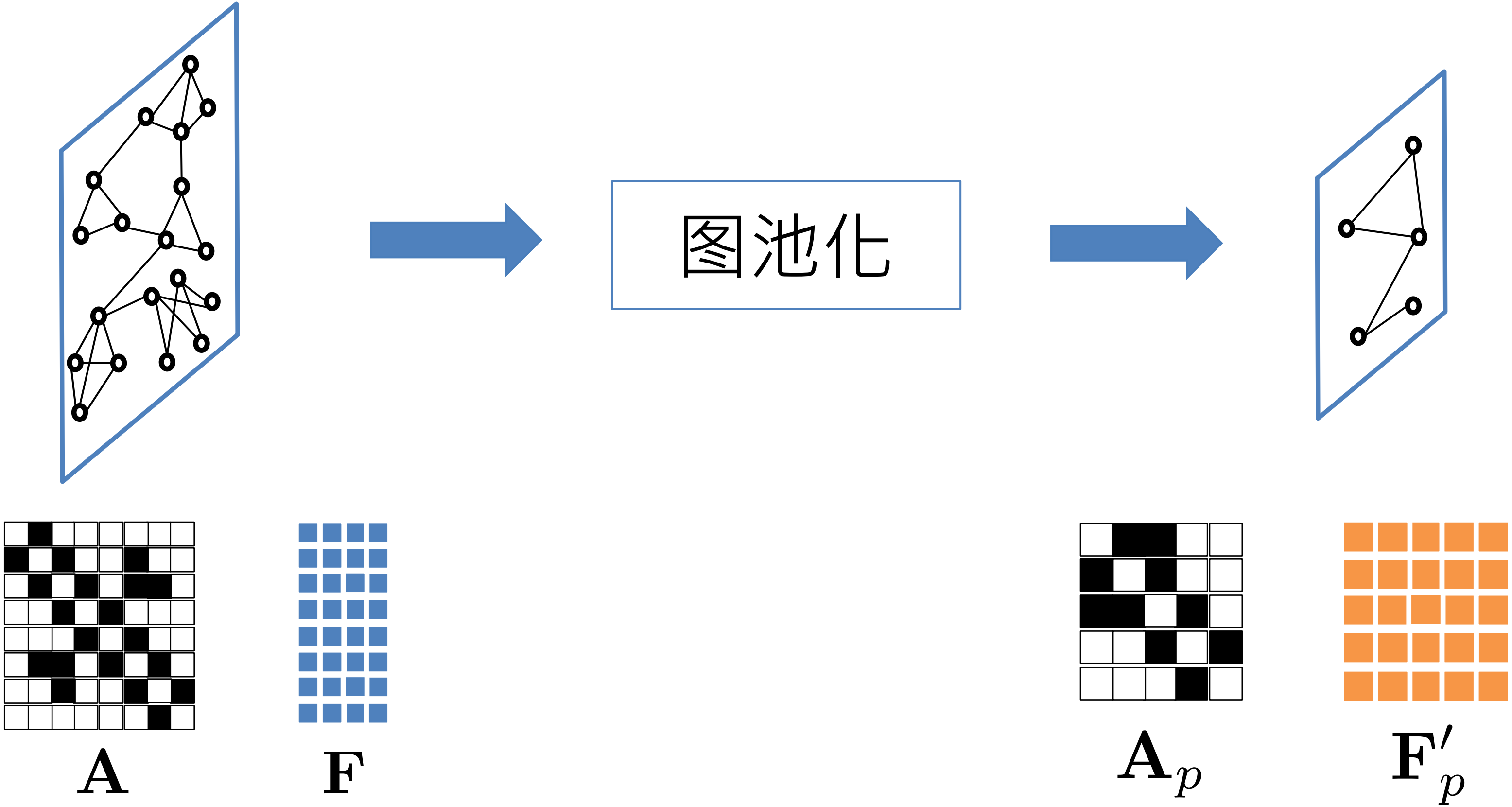
# 图滤波操作



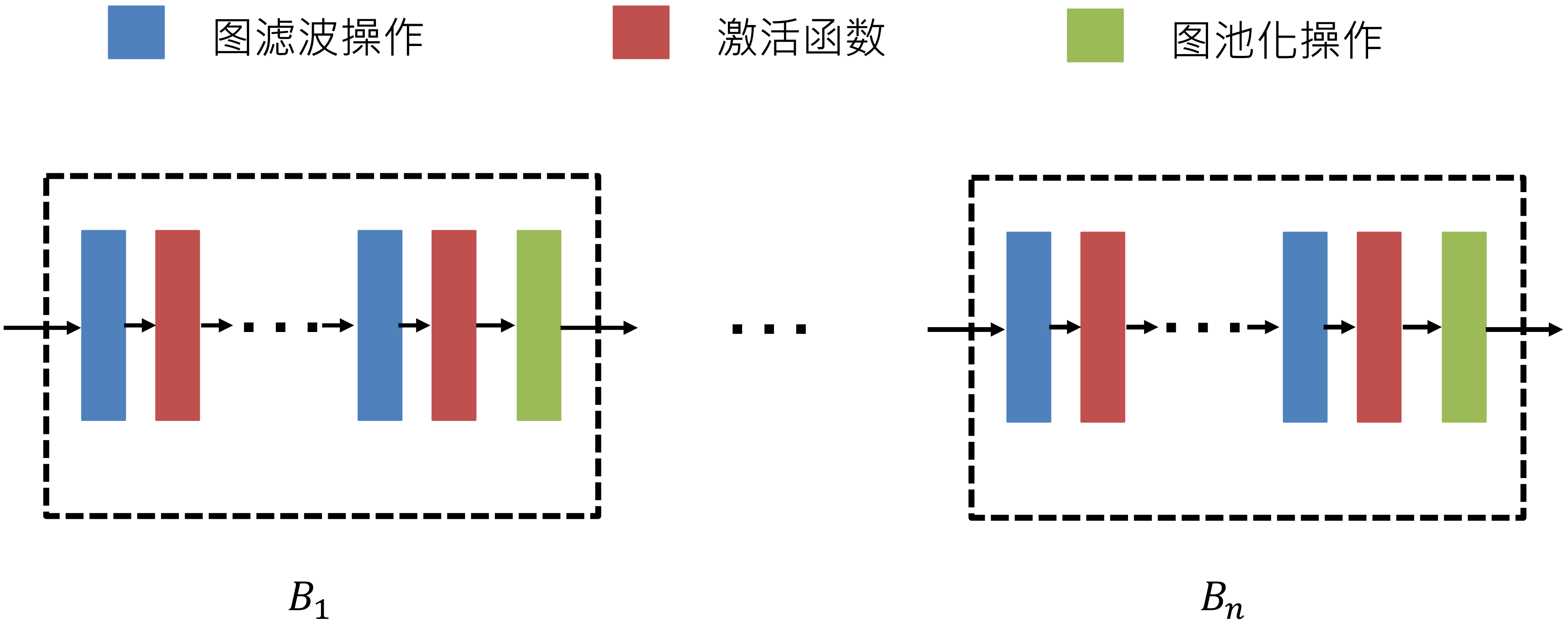
# 通用的GNN框架（节点任务）



# 图池化操作



# 通用的GNN框架（图任务）



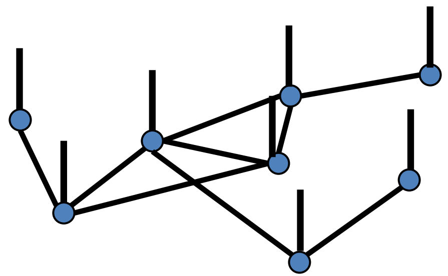
# 目录

# 拉普拉斯矩阵作为算子

拉普拉斯矩阵可以作为一个差分算子

$$\mathbf{h} = \mathbf{L}\mathbf{f} = (\mathbf{D} - \mathbf{A})\mathbf{f} = \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}$$

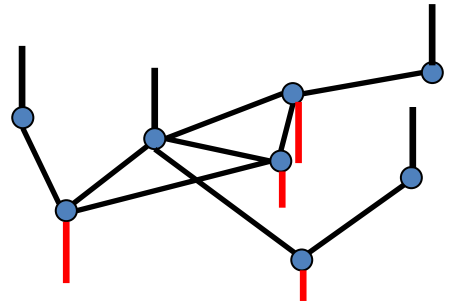
$$\mathbf{h}(i) = \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}(i) - \mathbf{f}(j))$$



“光滑”的“低频”信号

拉普拉斯二次型

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{N} \mathbf{A}[i,j](\mathbf{f}(i) - \mathbf{f}(j))^2$$



“不光滑”的“高频”信号

- 衡量了图信号的“光滑度”或者“频率”
- 拉普拉斯矩阵为半正定矩阵

# 拉普拉斯矩阵的特征分解

拉普拉斯矩阵有一套完整的标准正交的特征向量：

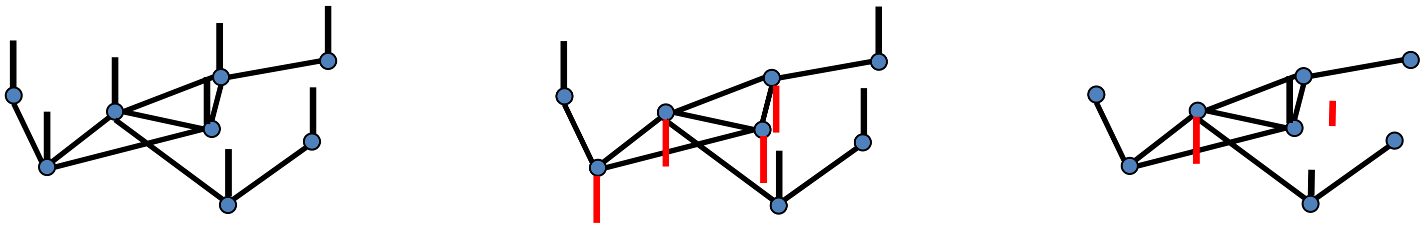
$$\mathbf{L} = \underbrace{\begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix}}_{\boldsymbol{U}} \underbrace{\begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}}_{\boldsymbol{\Lambda}} \underbrace{\begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}}_{\boldsymbol{U}^T}$$

通常我们将这些特征向量按照特征值从小到大排列，

$$0 = \lambda_0 < \lambda_1 \leq \cdots \lambda_{N-1}$$

# 特征向量作为图上的信号

这些特征向量是图信号的一组基



$\boldsymbol{u}_0$ $\boldsymbol{u}_1$ $\boldsymbol{u}_7$

低频 高频

$\boldsymbol{u}_0^T \boldsymbol{L} \boldsymbol{u}_0 = \boldsymbol{\lambda}_0 = \boldsymbol{0}$

$\boldsymbol{u}_1^T \boldsymbol{L} \boldsymbol{u}_1 = \boldsymbol{\lambda}_1$

$\boldsymbol{u}_7^T \boldsymbol{L} \boldsymbol{u}_7 = \boldsymbol{\lambda}_7$

频率： $\mathbf{u}_i^T \mathbf{L} \mathbf{u}_i$

# 图傅立叶变换（GFT）

任意的的图信号$f$可以用图傅立叶级数表示

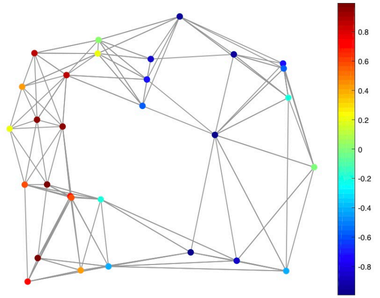
$$\mathbf{f} = \sum_{i=0}^{N-1} \hat{f}_i \cdot \mathbf{u}_i$$

$$\hat{f}_i = \boldsymbol{f}^\top \boldsymbol{u}_i$$

$\boldsymbol{u}_i$: 基（特征向量）

$\lambda_i$: 基的频率（特征值）

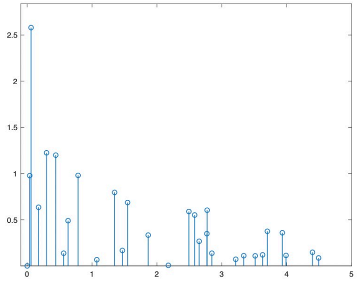
$\hat{f}_i$:傅立叶系数




空间域: $\boldsymbol{f}$

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

分解图信号




谱域: $\hat{\boldsymbol{f}}$

# 图傅立叶变换（GFT）

任意的的图信号$f$可以用图傅立叶级数表示

$$\mathbf{f} = \sum_{i=0}^{N-1} \boxed{\hat{f}_i} \cdot \mathbf{u}_i$$

$$\hat{f}_i = \boldsymbol{f}^\top \boldsymbol{u}_i$$

$\boldsymbol{u}_i$: 基（特征向量）

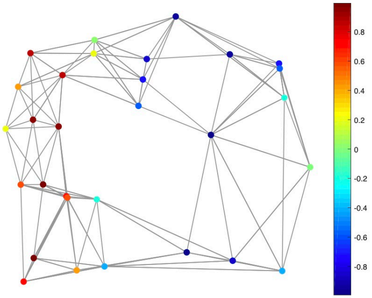
$\lambda_i$: 基的频率（特征值）

$\hat{f}_i$:傅立叶系数

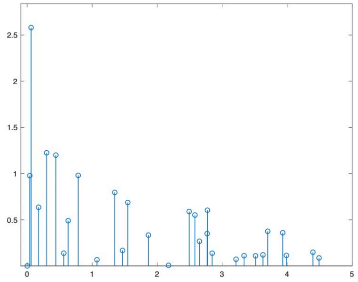
矩阵形式

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

重建图信号




空间域: $\boldsymbol{f}$




谱域: $\hat{\boldsymbol{f}}$

# 目录

# 两大类图滤波操作

基于空间的图滤波操作

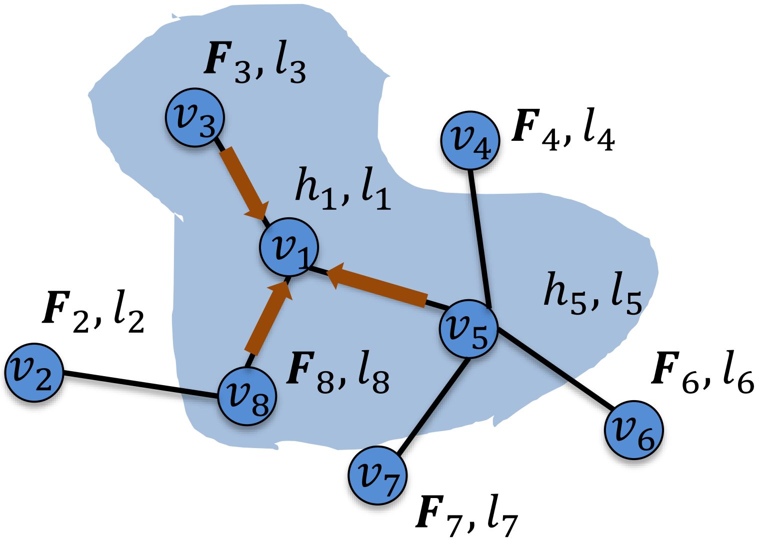
在空间域上设计

基于谱的图滤波操作

在谱域上设计

最早的图滤波操作 → 基于谱的图滤波操作 → 基于空间的图滤波操作

# 最早的图滤波操作




通常是上一个操作的输出

$\boldsymbol{F}_i$: 图滤波操作的输入特征

$l_i$: 初始的节点特征

$\boldsymbol{F}'_i$: 图滤波操作的输出特征

$$\boldsymbol{F}'_i = \sum_{v_j \in N(v_i)} f(l_i, \boldsymbol{F}_j, l_j), \qquad \forall\, v_i \in V.$$

$f(\cdot)$:通常是前馈神经网络

# 图谱滤波

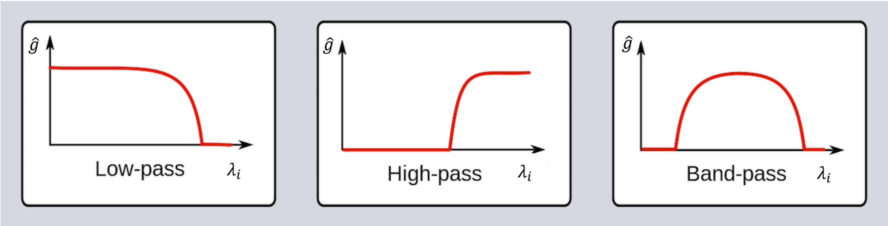
## 图傅立叶变换

$$GFT: \hat{\boldsymbol{f}} = \boldsymbol{U}^\top \boldsymbol{f} \qquad\qquad IGFT: \boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$$

## 图谱滤波

$$\boldsymbol{f} \xrightarrow[\text{分解}]{GFT} \underbrace{\boldsymbol{u}_i^\top \boldsymbol{f}}_{\text{系数}} \xrightarrow[\text{过滤}]{\hat{g}(\lambda_i)} \underbrace{\hat{g}(\lambda_i)\cdot \boldsymbol{u}_i^\top \boldsymbol{f}}_{\text{过滤后的系数}} \xrightarrow[\text{重构}]{IGFT} \sum_{i=0}^{N-1} \hat{g}(\lambda_i)\cdot \boldsymbol{u}_i^\top \boldsymbol{f}\cdot \boldsymbol{u}_i$$

Filter $\hat{g}(\lambda_i)$: 调制信号中不同频率的部分



# 图谱滤波

## 图傅立叶变换

$$GFT: \hat{\boldsymbol{f}} = \boldsymbol{U}^{\top}\boldsymbol{f} \qquad IGFT: \boldsymbol{f} = \boldsymbol{U}\hat{\boldsymbol{f}}$$

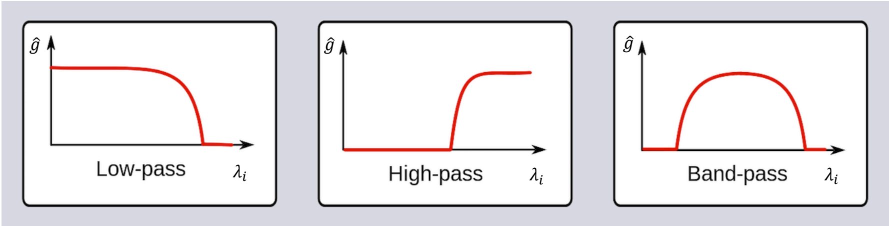
## 图谱滤波

$$\boldsymbol{f} \xrightarrow[\text{分解}]{GFT} \underbrace{\boldsymbol{U}^{\top}\boldsymbol{f}}_{\text{系数}} \xrightarrow[\text{过滤}]{\hat{g}(\Lambda)} \underbrace{\hat{g}(\Lambda)\boldsymbol{U}^{\top}f}_{\text{过滤后的系数}} \xrightarrow[\text{重构}]{IGFT} \boxed{\boldsymbol{U}\hat{g}(\Lambda)\boldsymbol{U}^{\top}}\boldsymbol{f}$$

$$f \xrightarrow[\text{过滤}]{} \hat{g}(\boldsymbol{L})f \qquad \boxed{\hat{g}(\boldsymbol{L})}$$

Filter $\hat{g}(\Lambda)$: 调制信号中不同频率的部分

$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

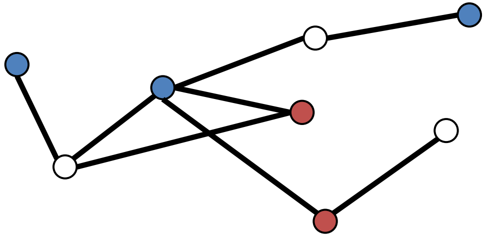


# 将图谱滤波运用到GNN上

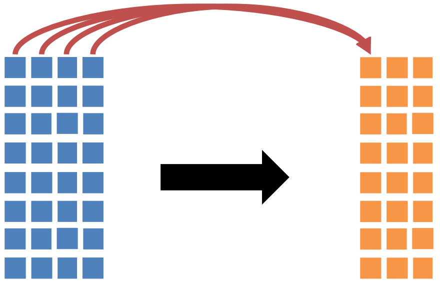
如何设计滤波器

$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

从数据中学习



如何处理多通道信号



$$\boldsymbol{F} \in \mathbb{R}^{N\times d_1} \rightarrow \boldsymbol{F}' \in \mathbb{R}^{N\times d_2}.$$

每个输出通道都与所有输入通道相关

$$\boldsymbol{F}'_{:,i} = \sum_{j=1}^{d_1} \boxed{\hat{g}_{ij}(\mathbf{L})} \boldsymbol{F}_{:,j} \quad i = 1, \ldots d_2$$

需要学习 $d_2 \times d_1$ 个滤波器

# $\hat{g}(\Lambda)$：非参数方法

$$\hat{g}(\Lambda) = \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{N-1}) \end{bmatrix}$$

## $\hat{g}(\Lambda)$：非参数方法

$$\hat{g}(\Lambda) = \begin{bmatrix} \theta_1 & & & \\ & \theta_2 & & \\ & & \cdots & \\ & & & \theta_N \end{bmatrix}$$

$\boldsymbol{U}\hat{g}(\Lambda)\boldsymbol{U}^T\boldsymbol{f}$

$d_2 \times d_1 \times N$个参数

需要做矩阵分解，通常比较耗时

# $\hat{g}(\Lambda)$：利用多项式拟合

$$\hat{g}(\Lambda) = \begin{bmatrix} \sum_{k=0}^{K} \theta_k \lambda_1^k & & & \\ & \sum_{k=0}^{K} \theta_k \lambda_2^k & & \\ & & \cdots & \\ & & & \sum_{k=0}^{K} \theta_k \lambda_N^k \end{bmatrix}$$

$$\boldsymbol{U}\hat{g}(\Lambda)\boldsymbol{U}^T\boldsymbol{f} = \sum_{k=0}^{K} \theta_k \boldsymbol{L}^k f$$

$d_2 \times d_1 \times (K+1)$个参数

不需要做特征分解

# Chebyshev多项式

$$g(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots$$

多项式的基：$1, x, x^2, x^3, \ldots$

非正交

导致学习参数的过程不稳定

Chebyshev 多项式

以递归的形式定义:

- $T_0(x) = 1; T_1(x) = x$
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

这样定义的Chebyshev 多项式 $\{T_k\}$ 是希尔伯特空间 $L^2([-1,1], \frac{dy}{\sqrt{1-y^2}})$ 中的一组正交基。

$$g(x) = \theta_0 T_0(x) + \theta_1 T_1(x) + \theta_2 T_2(x) + \cdots$$

# Cheb-Filter

利用 Chebyshev多项式来拟合 $\hat{g}(\Lambda)$

- 拉普拉斯矩阵的最大特征值
- 调整$\tilde{\Lambda}$大小到[-1,1]的区间

$$\hat{g}(\Lambda) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), with\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - 1$$

$$\mathbf{U}\hat{g}(\Lambda)\boldsymbol{U}^{\top}\boldsymbol{f} = \sum_{k=0}^{K} \theta_k T_k(\tilde{\boldsymbol{L}})\,\boldsymbol{f}, \text{ with } \tilde{\boldsymbol{L}} = \frac{2\boldsymbol{L}}{\lambda_{max}} - \boldsymbol{I}$$

$d_2 \times d_1 \times (K+1)$个参数

不需要做特征分解

学习参数的过程相对来说更稳定

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

# GCN-Filter：简化的Cheb-Filter

把$Chebshev$多项式的阶数设为一，$K = 1$。同时假设$\lambda_{max} = 2$

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

进一步假设 $\theta = \theta_0 = -\theta_1$

$$\hat{g}(\Lambda) = \theta(2I - \Lambda)$$

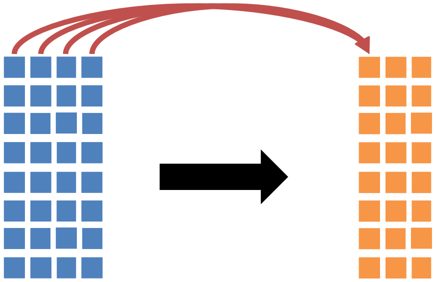
$$U\hat{g}(\Lambda)U^T f = \theta(2I - L)f = \theta\left(I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)f$$

使用renormalization

$$U\hat{g}(\Lambda)U^\top f = \theta\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)f, \text{ with } \tilde{A} = A + I$$

Semi-supervised Classification with Graph Convolutional Networks

# 处理多通道图信号的GCN-Filter



$\boldsymbol{F} \in \mathbb{R}^{N\times d_1} \rightarrow \boldsymbol{F}' \in \mathbb{R}^{N\times d_2}.$

每个输出通道都与所有输入通道相关

$$\boldsymbol{F}'_{:,i} = \sum_{j=1}^{d_1} \hat{g}_{ij}(\mathbf{L})\boldsymbol{F}_{:,j} \quad i = 1, \dots d_2$$

GCN-Filter

$$\boldsymbol{F}'_{:,i} = \sum_{j=1}^{d_1} \theta_{ji}(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}})\boldsymbol{F}_{:,j} \quad i = 1, \dots d_2$$

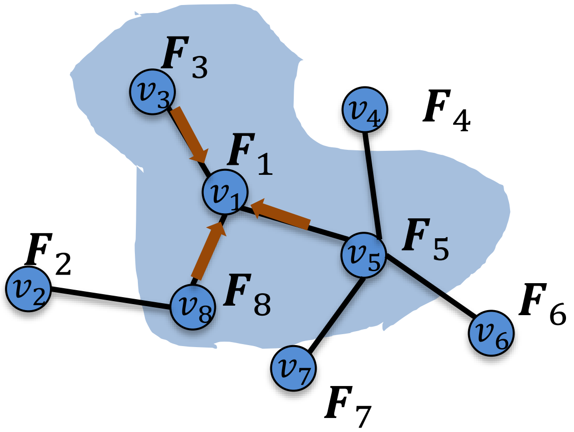
矩阵形式

$$\boldsymbol{F}' = (\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}})\boldsymbol{F}\boldsymbol{\Theta} \text{ with } \boldsymbol{\Theta} \in \mathbb{R}^{d_1\times d_2} \text{ and } \boldsymbol{\Theta}[j,i] = \theta_{ji}$$

# 从空间域理解GCN-Filter

令 $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$

- 那么$\mathbf{F}' = \widehat{\boldsymbol{A}}\boldsymbol{F}\boldsymbol{\Theta}$
- 对于节点 $v_i$, $\boldsymbol{F}_i' = \sum_j \widehat{\boldsymbol{A}}[i,j]\boldsymbol{F}_j\boldsymbol{\Theta}$



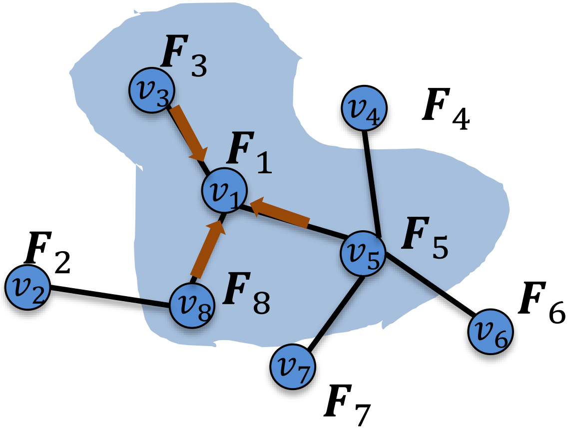

可以观察到

$$\widehat{\boldsymbol{A}}[i,j] = 0 \text{ for } v_j \notin N(v_i) \cup \{v_i\}$$

从空间域理解

$$\boldsymbol{F}_i' = \sum_j \widehat{\boldsymbol{A}}[i,j]\boldsymbol{F}_j\boldsymbol{\Theta} = \sum_{v_j \in N(v_i)\cup\{v_i\}} \widehat{\boldsymbol{A}}[i,j]\boldsymbol{F}_j\boldsymbol{\Theta}$$

# GraphSage-Filter




采样

$$\mathcal{N}_S(v_i) = \text{SAMPLE}(\mathcal{N}(v_i), S)$$

采样数量

聚合

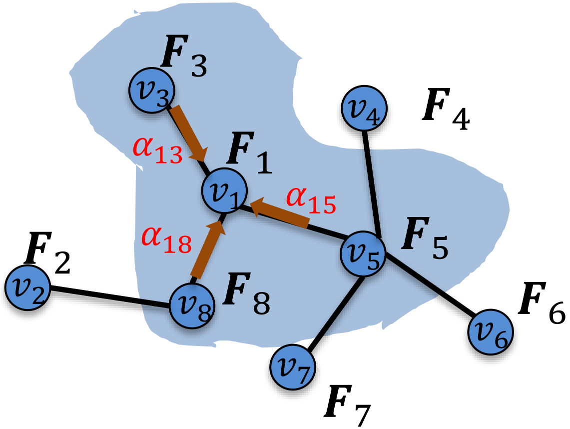
$$f'_{\mathcal{N}_S(v_i)} = \text{AGG}\left(\{\boldsymbol{F}_j, \forall v_j \in \mathcal{N}_S(v_i)\}\right)$$

$$\boldsymbol{F}'_i = \sigma\left(\left[\boldsymbol{F}_i, f'_{\mathcal{N}_S(v_i)}\right]\boldsymbol{\Theta}\right)$$

聚合函数：如求和，求平均，LSTM等

Inductive Representation Learning on Large graphs

# GAT-Filter



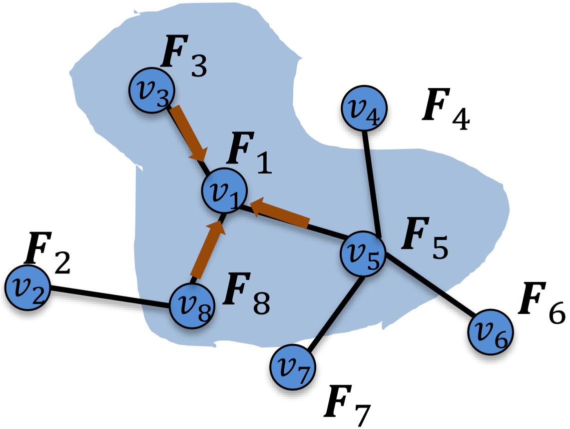

GCN-Filter: $\boldsymbol{F}'_i = \sum_{v_j \in N(v_i) \cup \{v_i\}} \boxed{\widehat{\boldsymbol{A}}[i,j]} \boldsymbol{F}_j \boldsymbol{\Theta}$

GAT-Filter: $\boldsymbol{F}'_i = \sum_{v_j \in N(v_i) \cup \{v_i\}} \boxed{\alpha_{ij}} \boldsymbol{F}_j \boldsymbol{\Theta}$

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{\top}\left[\boldsymbol{\Theta}\vec{\mathbf{F}}_i \| \boldsymbol{\Theta}\vec{\mathbf{F}}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}(v_i) \cup \{v_i\}} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^{\top}\left[\boldsymbol{\Theta}\vec{\mathbf{F}}_i \| \boldsymbol{\Theta}\vec{\mathbf{F}}_k\right]\right)\right)}$$

Graph Attention Networks

# ECC-Filter



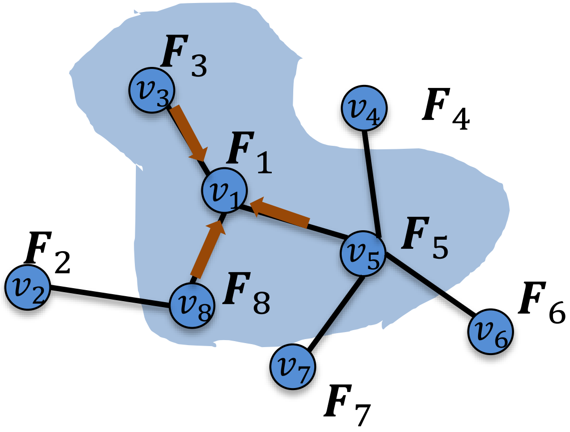

边有不同的类型

$$\boldsymbol{F}_i' = \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} \boldsymbol{F}_j \boldsymbol{\Theta}_{\mathrm{tp}(v_i, v_j)}$$

针对某种特定边的类型的参数矩阵

Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs

# GGNN-Filter



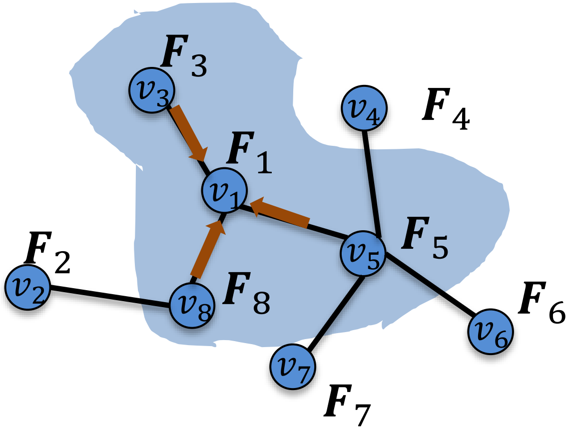

针对某种特定边的类型的参数矩阵

$$\boldsymbol{m}_i = \sum_{(v_j, v_i) \in \mathcal{E}} \Theta^{e}_{\mathrm{tp}(v_j, v_i)} \boldsymbol{F}_j$$

$$\boldsymbol{F}'_i = \mathrm{GRU}\left(\boldsymbol{m}_i, \boldsymbol{F}_i\right)$$

Gated Graph Sequence Neural Networks

# MPNN




信息传递

$$\boldsymbol{m}_i = \sum_{v_j \in \mathcal{N}(v_i)} M\left(\boldsymbol{F}_i, \boldsymbol{F}_j, \boldsymbol{e}_{(v_i, v_j)}\right)$$

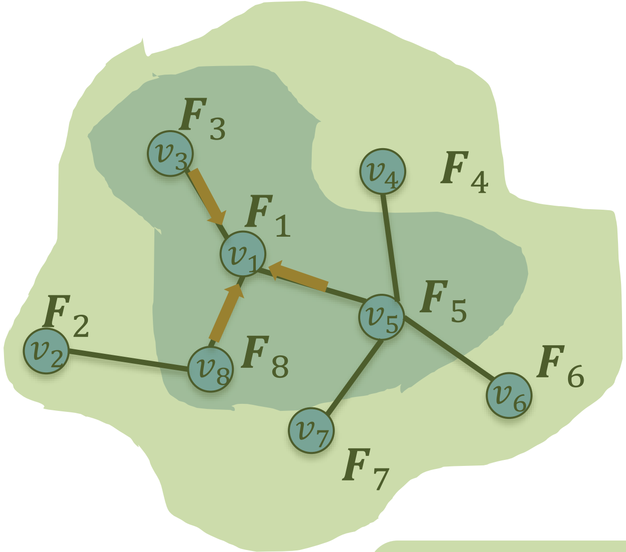
更新函数

$$\mathbf{F}'_i = U\left(\boldsymbol{F}_i, \boldsymbol{m}_i\right)$$

$M_k()$ 和 $U_k()$是需要进一步设计的函数

Neural Message Passing for Quantum Chemistry

# PPNP




GCN-Filter: $F_i' = \sum_{v_j \in N(v_i) \cup \{v_i\}} \hat{A}[i,j] F_j \Theta$

从周围节点聚合

PPNP: $F_i' = \sum_{v_j \in \mathcal{V}} P[i,j] F_j \Theta$

可以换成MLP

$$\mathbf{P} = \beta(\mathbf{I} - (1-\beta)\hat{\mathbf{A}})^{-1}$$

Personalized PageRank

Predict then Propagate: Graph Neural Networks meet Personalized PageRank

# APPNP

PPNP: $\boldsymbol{F}_i' = \sum_{v_j \in \mathcal{V}} P[i,j] \boldsymbol{F}_{\mathrm{j}} \boldsymbol{\Theta}$ $\xrightarrow{\text{矩阵形式}}$ $\boldsymbol{F}' = \boldsymbol{P}\boldsymbol{F}\boldsymbol{\Theta}$

$\boldsymbol{F_{tr}}$

$$\mathbf{P} = \beta(\mathbf{I} - (1-\beta)\hat{\mathbf{A}})^{-1}$$

计算复杂

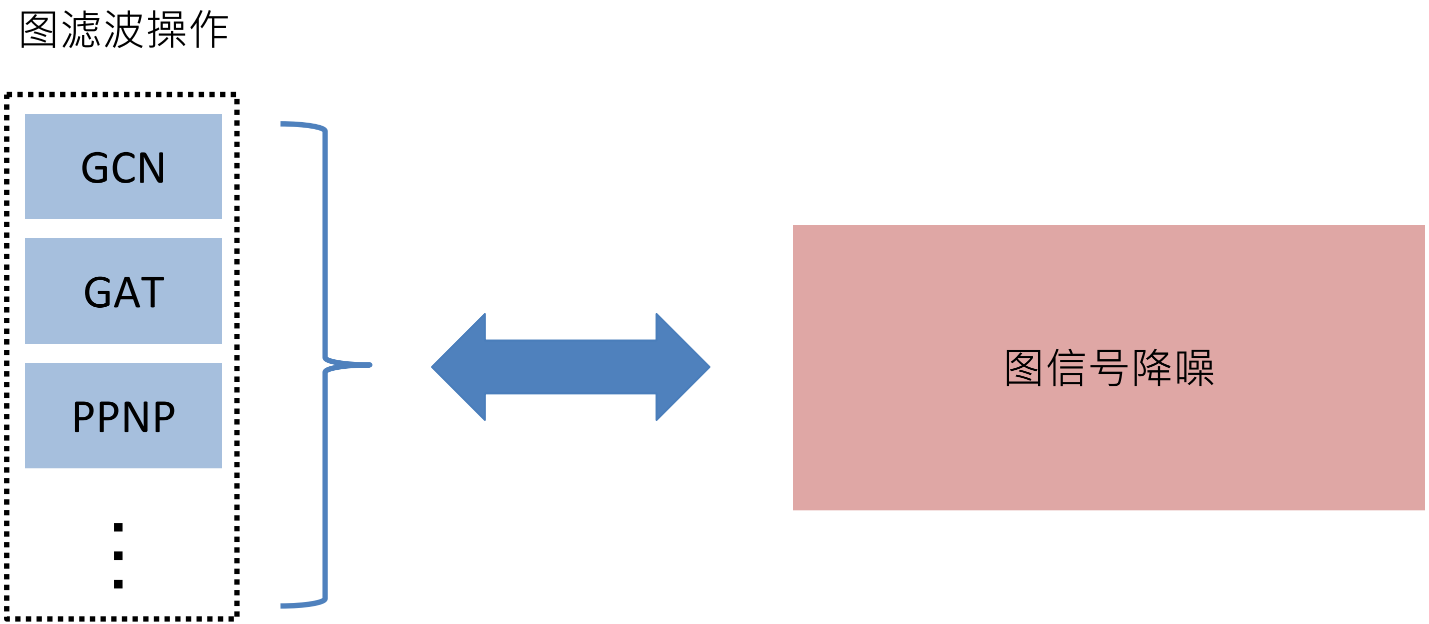
可拓展性差

APPNP:用迭代的方式逼近$\boldsymbol{F}'$

$$\mathbf{F}_{out}^{(k)} = (1-\beta)\hat{\mathbf{A}}\mathbf{F}_{out}^{(k-1)} + \beta\mathbf{F}_{tr} \quad k = 1, \ldots K$$

$$\mathbf{F}' = \mathbf{F}_{out}^{(K)} \xrightarrow[K \to \infty]{\text{收敛}} \boldsymbol{F}' = \boldsymbol{P}\boldsymbol{F_{tr}}$$

# 从图信号降噪的角度理解图滤波




A Unified View on Graph Neural Networks as Graph Signal Denoising

# 一些滤波操作的简单回顾

特征转换 $\boldsymbol{F_{tr}} = \boldsymbol{F\Theta}$

GCN

$$\mathbf{F}' = \hat{\mathbf{A}}\mathbf{F}_{tr}$$

GAT

$$\mathbf{F}'_i = \sum_{v_j \in \mathcal{N}(v_i) \cup \{v_i\}} \alpha_{ij}\mathbf{F}_{tr}$$

PPNP

$$\mathbf{F}' = \mathbf{P}\mathbf{F}_{tr}$$

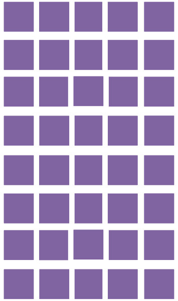
APPNP

$$\mathbf{F}_{out}^{(k)} = (1-\beta)\hat{\mathbf{A}}\mathbf{F}_{out}^{(k-1)} + \beta\mathbf{F}_{tr}$$

$$\mathbf{F}' = \mathbf{F}_{out}^{(K)}$$

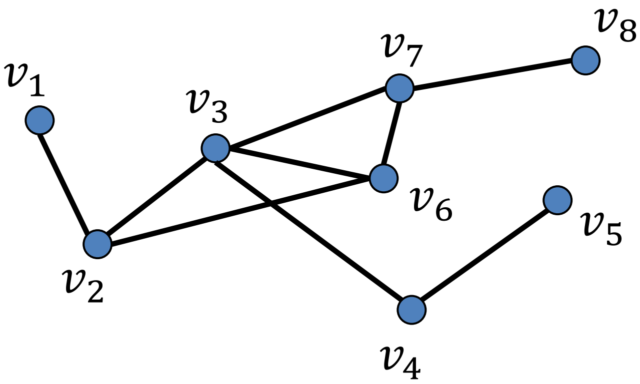
# 图信号降噪

“Noisy Signal”



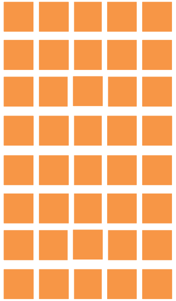
$\mathbf{F}_{tr}$

图




“节点和它的邻居相似”

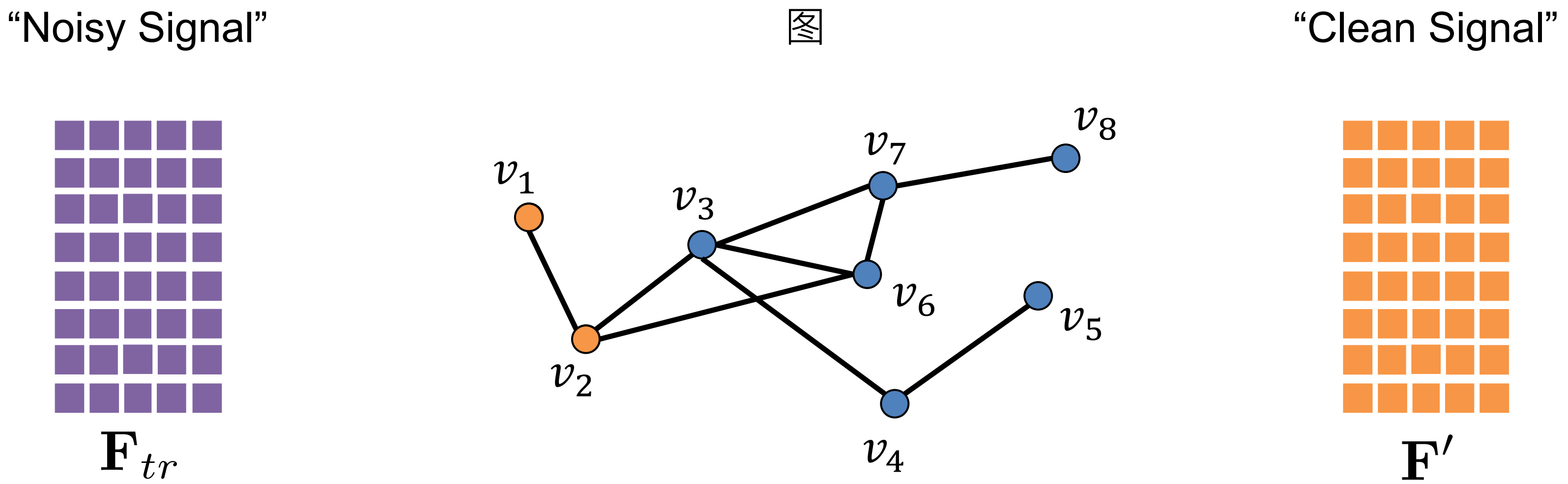
“Clean Signal”



$\mathbf{F}'$

$$\arg\min_{\mathbf{F}'} \mathcal{L}(\mathbf{F}') = \underbrace{\|\mathbf{F}' - \mathbf{F}_{tr}\|_F^2}_{\text{Close to the input}} + \underbrace{c \cdot \frac{1}{2}\sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|\mathbf{F}'_i - \mathbf{F}'_j\|_2^2}_{\text{Enforce the prior}}$$

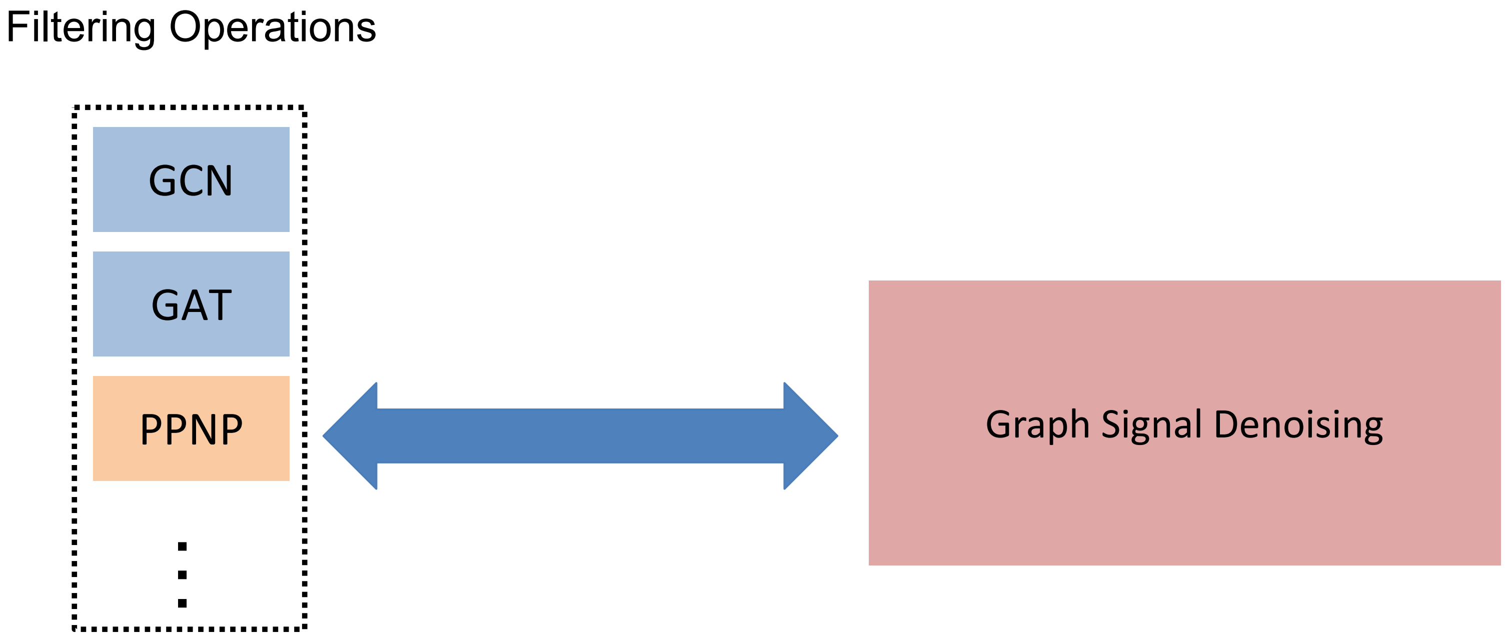
# 图信号降噪




“节点和它的邻居相似，相似程度可能不同”

$$\arg\min_{\mathbf{F}'} \mathcal{L}\left(\mathbf{F}'\right) = \|\mathbf{F}' - \mathbf{F}_{tr}\|_F^2 + \frac{1}{2}\sum_{i\in\mathcal{N}(i)} c_i \cdot \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}} \|\mathbf{F}'_i - \mathbf{F}'_j\|_2^2$$

# 从图降噪角度理解PPNP



# 从图降噪角度理解PPNP

$$\arg\min_{\mathbf{F}'} \mathcal{L}(\mathbf{F}') = \|\mathbf{F}' - \mathbf{F}_{tr}\|_F^2 + c \cdot \frac{1}{2} \sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \sum_{i \in \mathcal{N}(i)} \|\mathbf{F}'_i - \mathbf{F}'_j\|_2^2$$

精确解:

$$\mathbf{F}' = \frac{1}{1+c}\left(\mathbf{I} - \frac{c}{1+c}\hat{\mathbf{A}}\right)^{-1}\mathbf{F}_{tr}$$

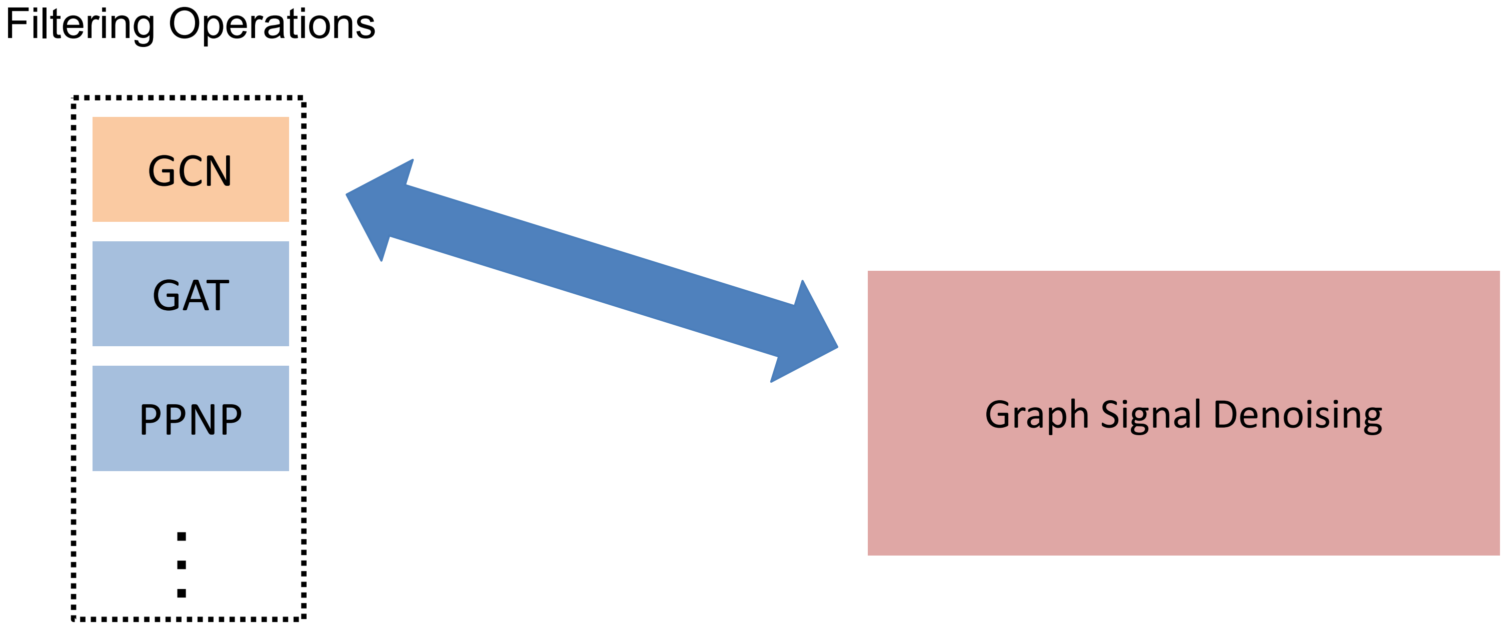
$$Set\ \beta = \frac{1}{1+c}$$

PPNP:

$$\mathbf{F}' = \mathbf{P}\mathbf{F}_{tr}$$

$$\mathbf{P} = \beta(\mathbf{I} - (1-\beta)\hat{\mathbf{A}})^{-1}$$

# 从图降噪角度理解GCN



# 从图降噪角度理解GCN

$$\arg\min_{\mathbf{F}'} \mathcal{L}(\mathbf{F}') = \|\mathbf{F}' - \mathbf{F}_{tr}\|_F^2 + c \cdot \frac{1}{2} \sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \sum_{i \in \mathcal{N}(i)} \|\mathbf{F}_i' - \mathbf{F}_j'\|_2^2$$

Gradient Descent:

$$\mathbf{F}' \leftarrow \mathbf{F}_{tr} - b \cdot \left.\frac{\partial \mathcal{L}}{\partial \mathbf{F}'}\right|_{\mathbf{F}'=\mathbf{F}_{tr}} = (1 - 2bc)\mathbf{F}_{tr} + 2bc\hat{\mathbf{A}}\mathbf{F}_{tr}$$

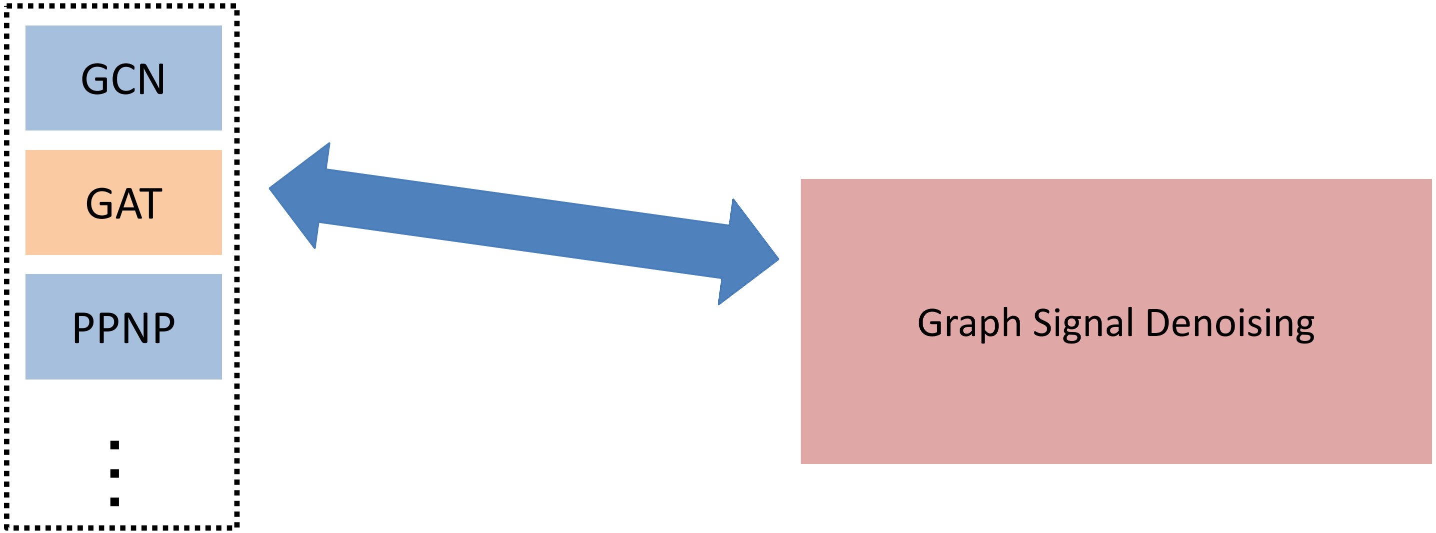
$$Set\ b = \frac{1}{2c}$$

$\Updownarrow$

GCN: $\mathbf{F}' = \hat{\mathbf{A}}\mathbf{F}_{tr}$

# 从图降噪角度理解GAT

Filtering Operations



# 从图降噪角度理解GAT

$$\arg\min_{\mathbf{F}'} \mathcal{L}\left(\mathbf{F}'\right) = \left\|\mathbf{F}' - \mathbf{F}_{tr}\right\|_F^2 + \frac{1}{2}\sum_{i\in\mathcal{N}(i)} c_i \cdot \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}}\left\|\mathbf{F}'_i - \mathbf{F}'_j\right\|_2^2$$

Gradient Descent（针对第$i$个节点）:

$$\mathbf{F}'_i \leftarrow \mathbf{F}_{tr}[i,:] - b_i \cdot \left.\frac{\partial\mathcal{L}}{\partial\mathbf{F}'_i}\right|_{\mathbf{F}'_i=\mathbf{F}_{tr}[i,:]}$$

$$= \boxed{\cancel{\left(1 - b_i \sum_{j\in\mathcal{N}(i)}\left(c_i + c_j\right)\right)}}\mathbf{F}_{tr}[i,:] + \sum_{j\in\mathcal{N}(i)} b_i\left(c_i + c_j\right)\mathbf{F}_{tr}[j,:]$$

$$\text{Set} \sum_{j\in\mathcal{N}(i)} b_i\left(c_i + c_j\right) = 1$$

# 从图降噪角度理解GAT

## Gradient Descent Solution to Graph Signal Denoising:

$$\mathbf{F}'_i \leftarrow \sum_{j \in \mathcal{N}(i)} \boxed{b_i\left(c_i + c_j\right)} \mathbf{F}_{tr}[j,:] \quad \text{where} \quad \sum_{j \in \mathcal{N}(i)} b_i\left(c_i + c_j\right) = 1$$

$$Let\ \alpha_{ij} = b_i(c_i + c_j)$$

GAT:

$$\mathbf{F}'_i = \sum_{j \in \mathcal{N}(i)} \boxed{\alpha_{ij}} \mathbf{F}_{tr}[j,:]$$

where

$$e_{ij} = f\left(\mathbf{F}_{tr}[i,:], \mathbf{F}_{tr}[j,:]\right)$$

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

# 一个统一的理解

Graph Convolutional Networks (GCN)

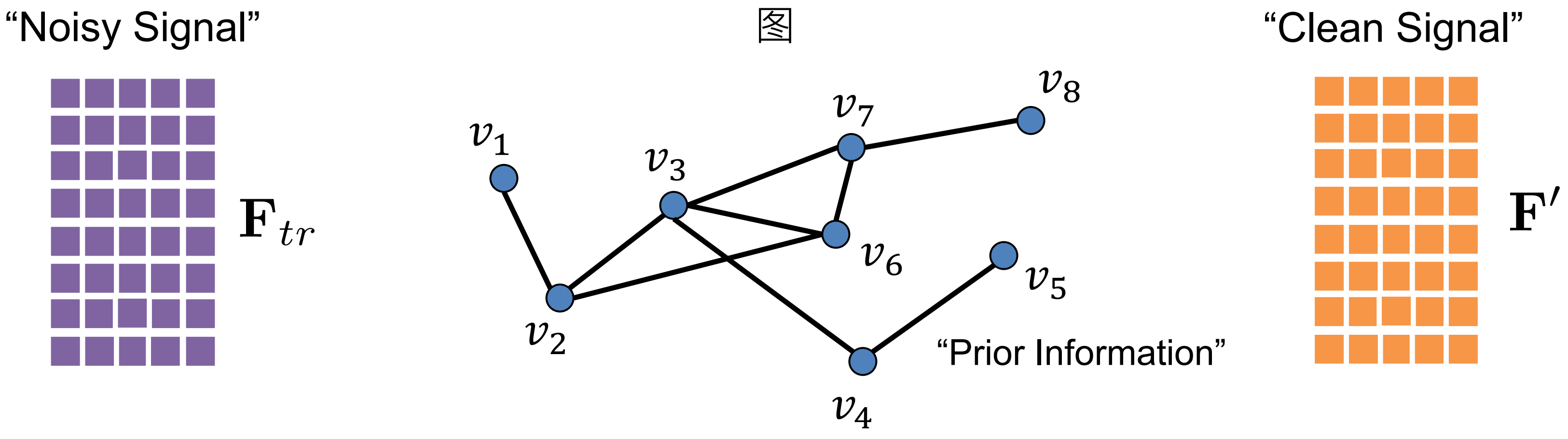
Graph Attention Networks (GAT)

Personalized Propagation of Neural Predictions (PPNP)

⋮

图信号降噪

# 一个统一的框架：UGNN




$$\arg\min_{\mathbf{F}'} \mathcal{L}(\mathbf{F}') = \|\mathbf{F}' - \mathbf{F}_{tr}\|_F^2 + \underline{\mathcal{R}(\mathbf{F}')} \qquad (1)$$

Enforce the prior

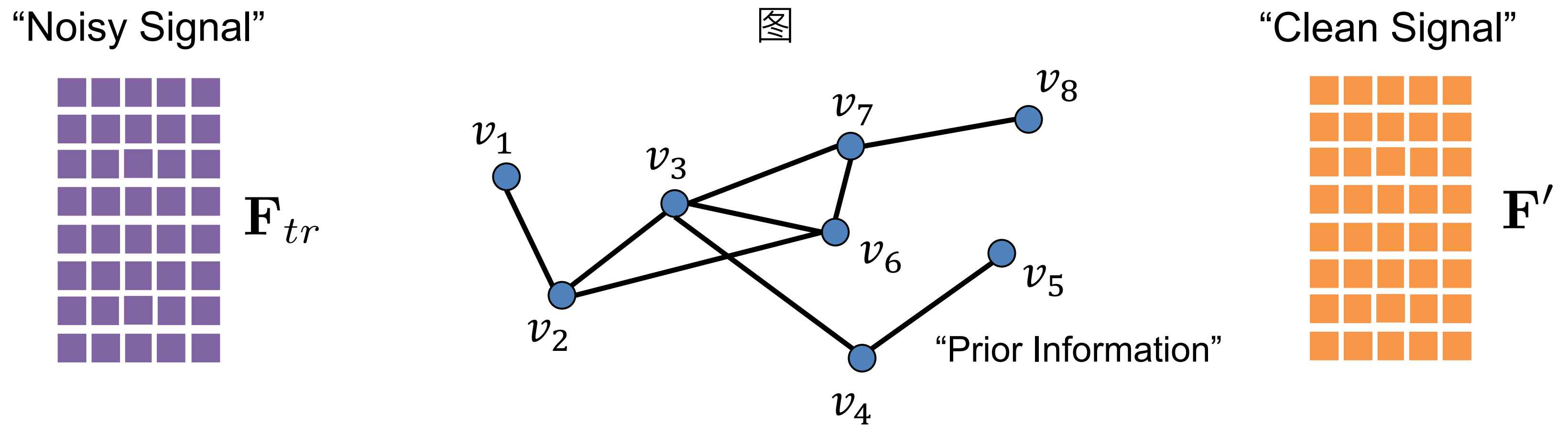
GCN, PPNP:

$$\mathcal{R}(\mathbf{F}') = c \cdot \frac{1}{2} \sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \sum_{i \in \mathcal{N}(i)} \|\mathbf{F}'_i - \mathbf{F}'_j\|_2^2$$

GAT:

$$\mathcal{R}(\mathbf{F}') = \frac{1}{2} \sum_{i \in \mathcal{N}(i)} c_i \cdot \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \|\mathbf{F}'_i - \mathbf{F}'_j\|_2^2$$

# 利用UGNN设计新的图滤波操作

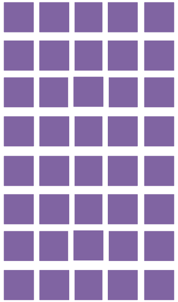



$$\arg\min_{\mathbf{F}'} \mathcal{L}(\mathbf{F}') = \|\mathbf{F}' - \mathbf{F}_{tr}\|_F^2 + \mathcal{R}(\mathbf{F}') \qquad (1)$$

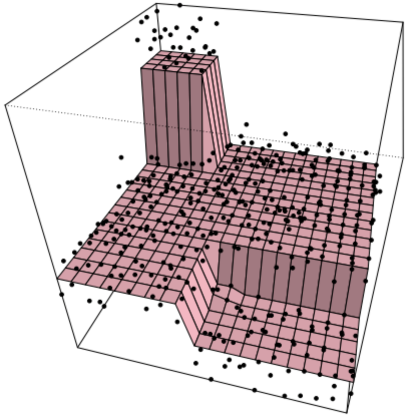
Design new $\mathcal{R}(\mathbf{F}')$ ➡ Solve (1) ➡ Novel graph filtering

# Graph Trend Filtering

"Noisy Signal"

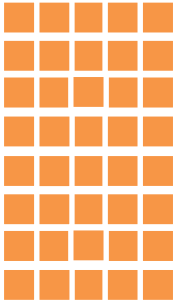


$\mathbf{F}_{tr}$



"Prior Information"

"Clean Signal"



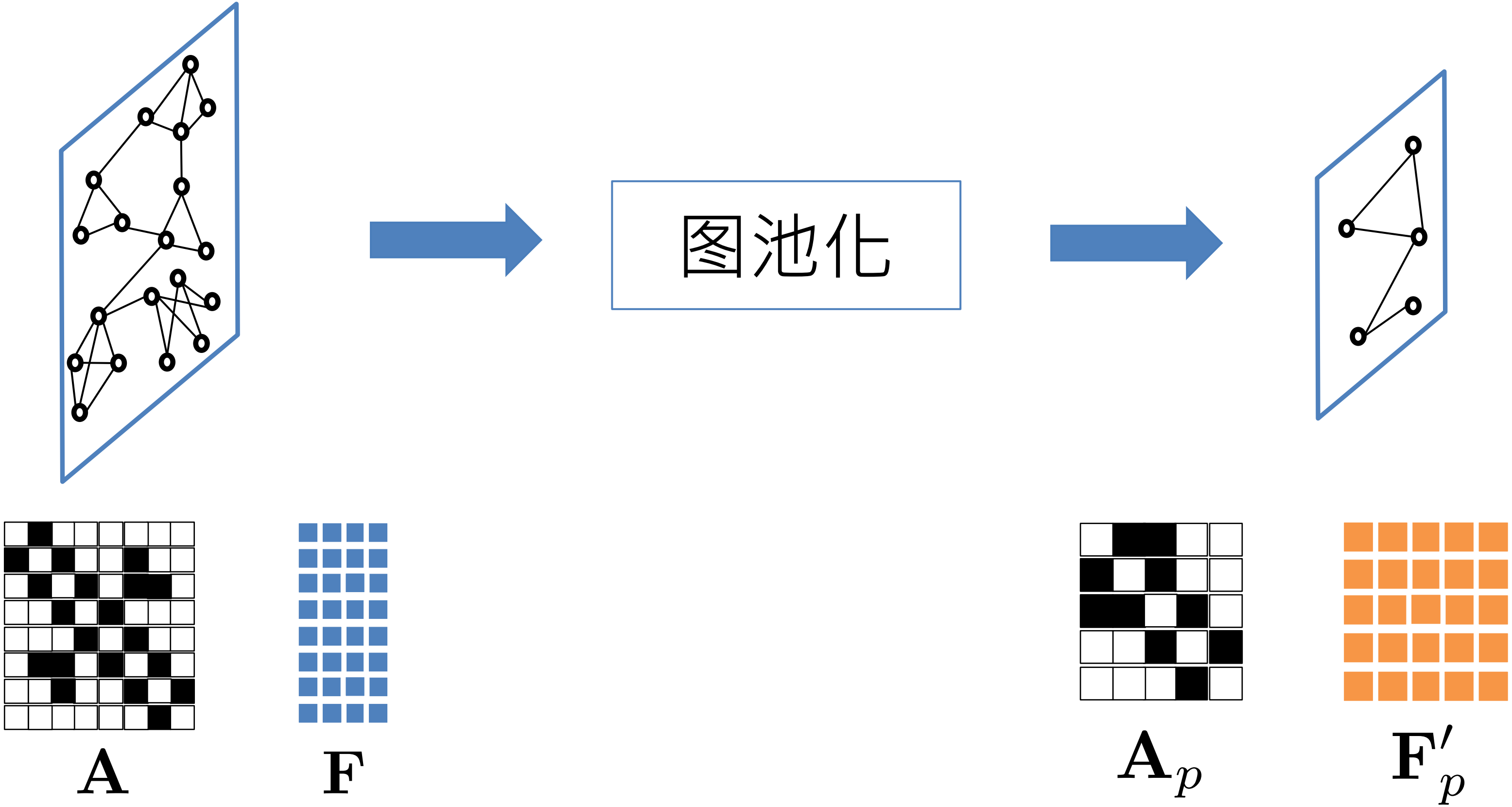
$\mathbf{F}'$

$$\arg\min_{\mathbf{F}'} \mathcal{L}\left(\mathbf{F}'\right) = \left\|\mathbf{F}' - \mathbf{F}_{tr}\right\|_F^2 + \mathcal{R}\left(\mathbf{F}'\right) \qquad (1)$$

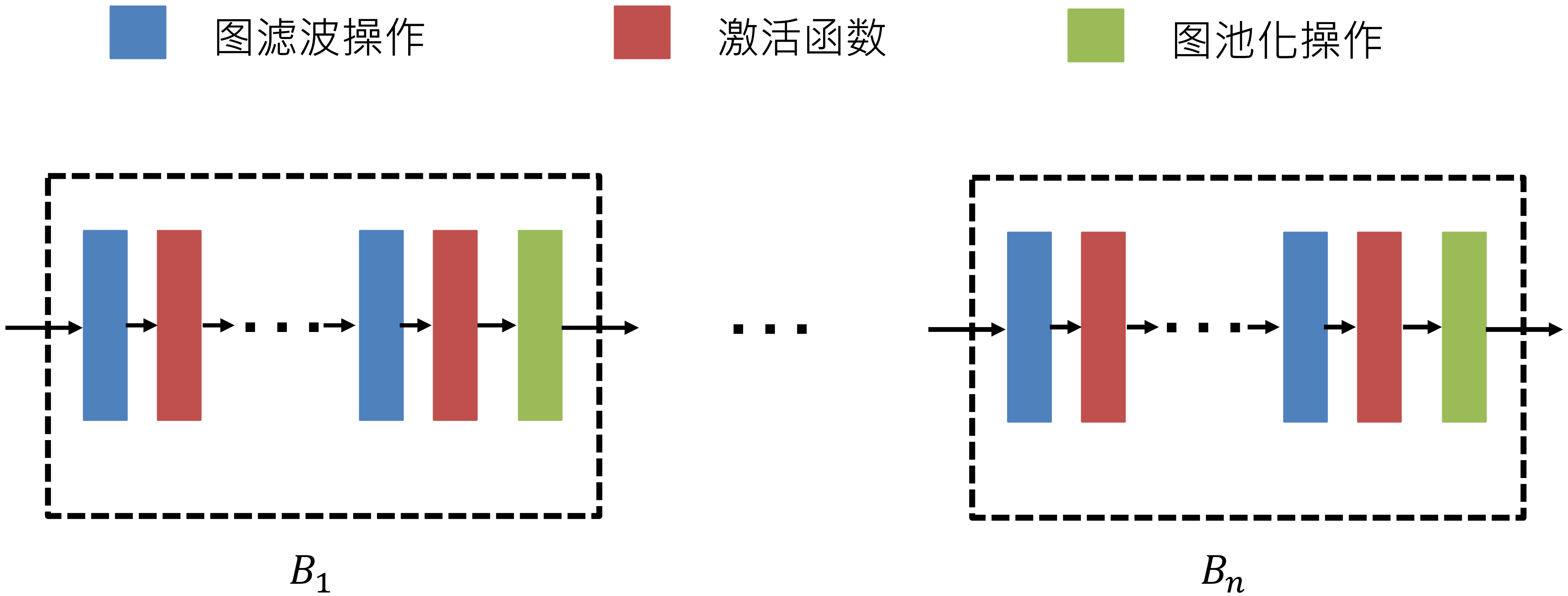
$$\mathcal{R}\left(\mathbf{F}'\right) = c \cdot \frac{1}{2} \sum_{i \in \mathcal{V}} \frac{1}{|\mathcal{N}(i)|} \underline{\sum_{j \in \mathcal{N}(i)} \left\|\mathbf{F}'_i - \mathbf{F}'_j\right\|_1}$$

# 目录

# 图池化操作

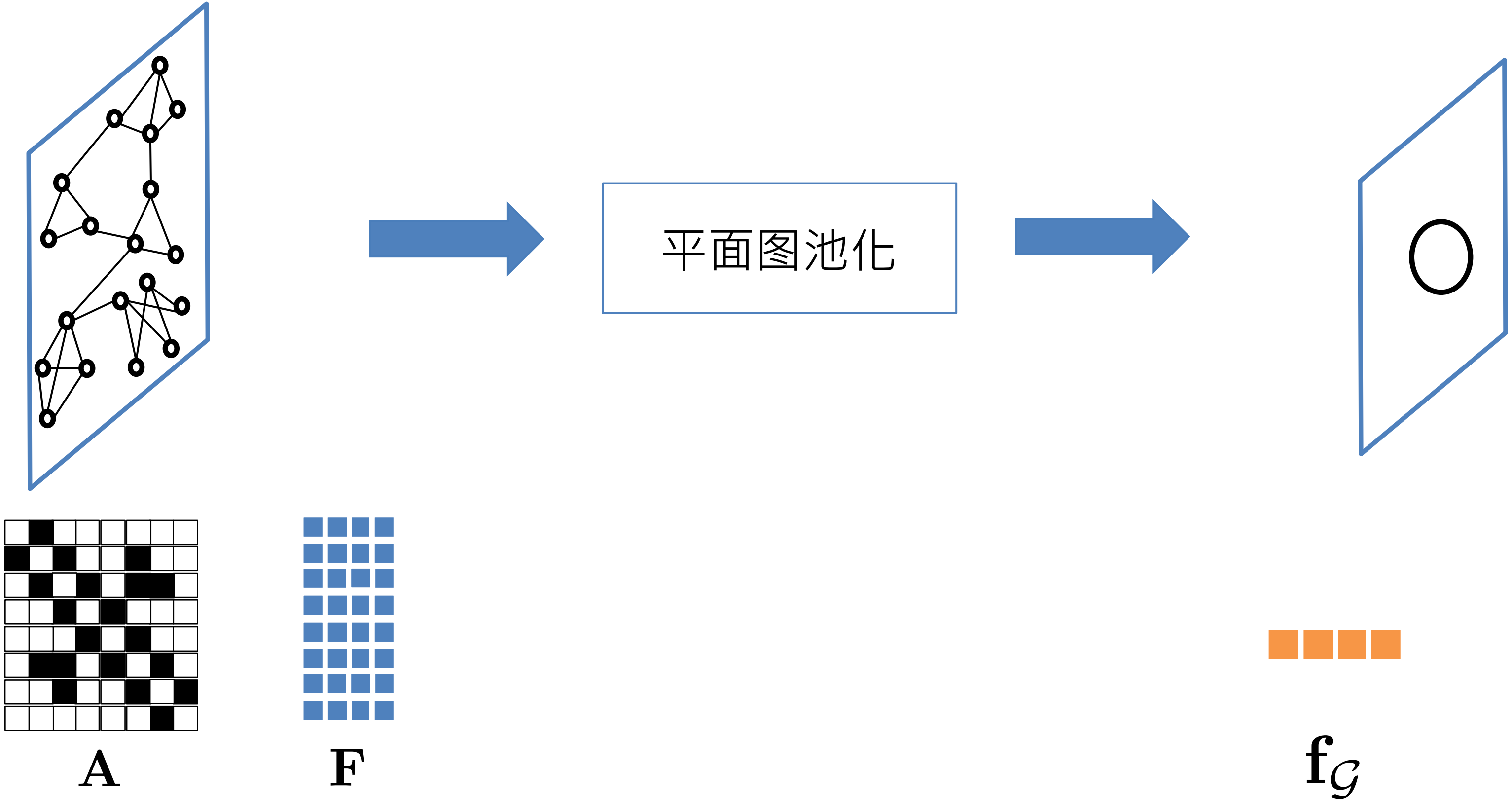


# 通用的GNN框架（图任务）

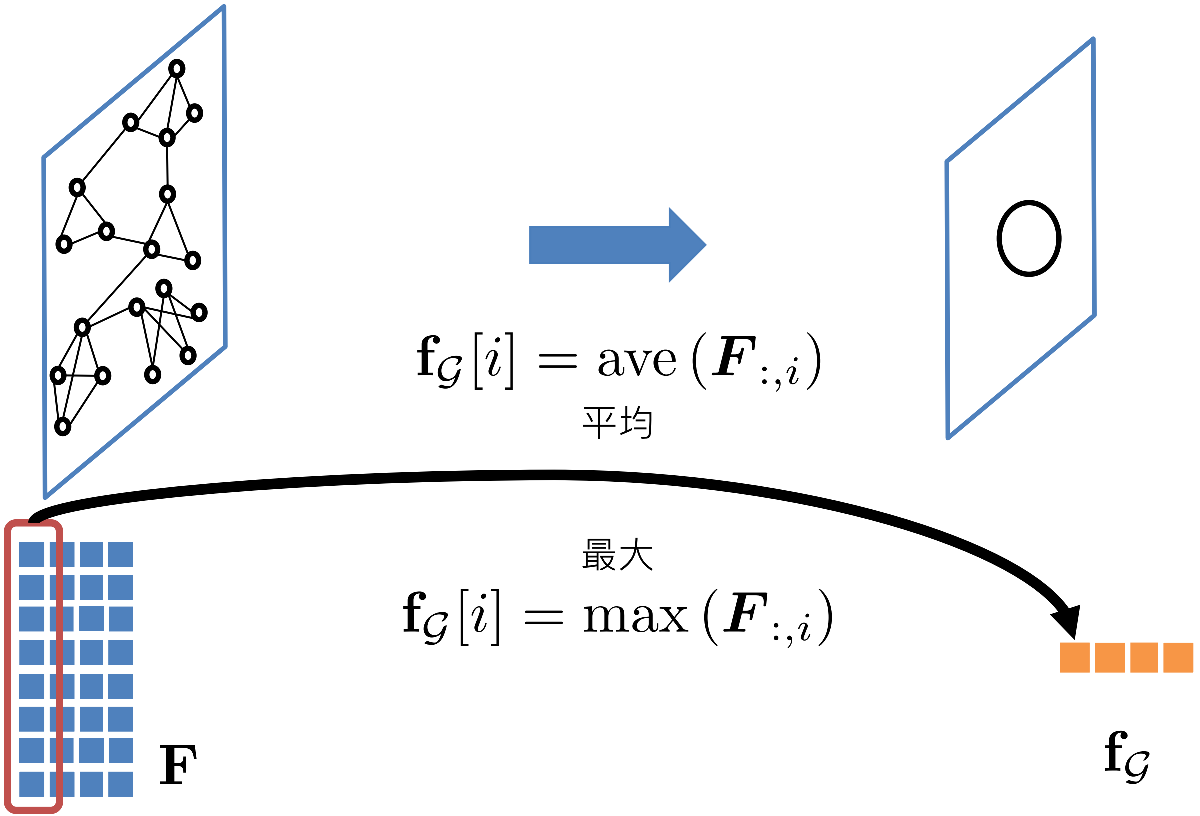



这个过程是层次化的

# 平面池化

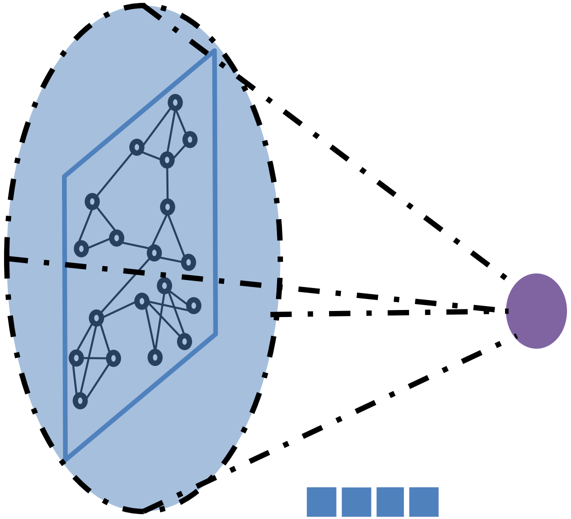


# 平均池化和最大池化

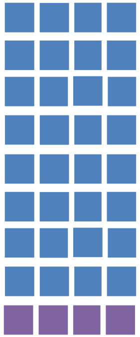


$$\mathbf{f}_{\mathcal{G}}[i] = \text{ave}\left(\boldsymbol{F}_{:,i}\right)$$
平均
最大
$$\mathbf{f}_{\mathcal{G}}[i] = \max\left(\boldsymbol{F}_{:,i}\right)$$
$\mathbf{F}$
$\mathbf{f}_{\mathcal{G}}$

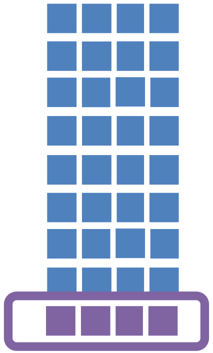
# 添加虚拟节点进行池化



- 往图中添加一个虚拟节点
- 使这个虚拟节点与原图中所有节点连接



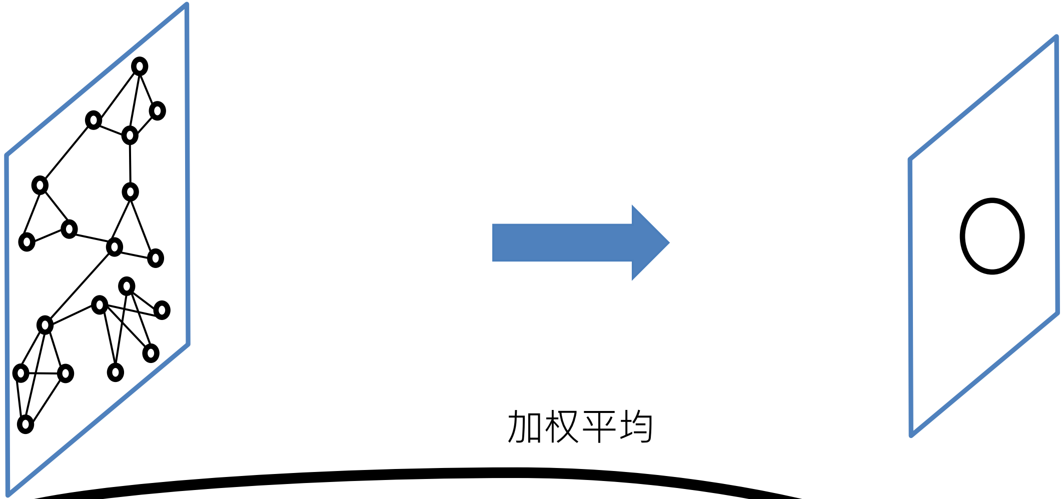
在新的图上更新节点表示



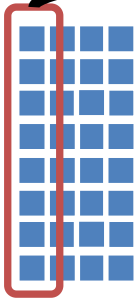
虚拟节点

作为图的表示$f_G$

# 基于注意力机制的平面池化



加权平均



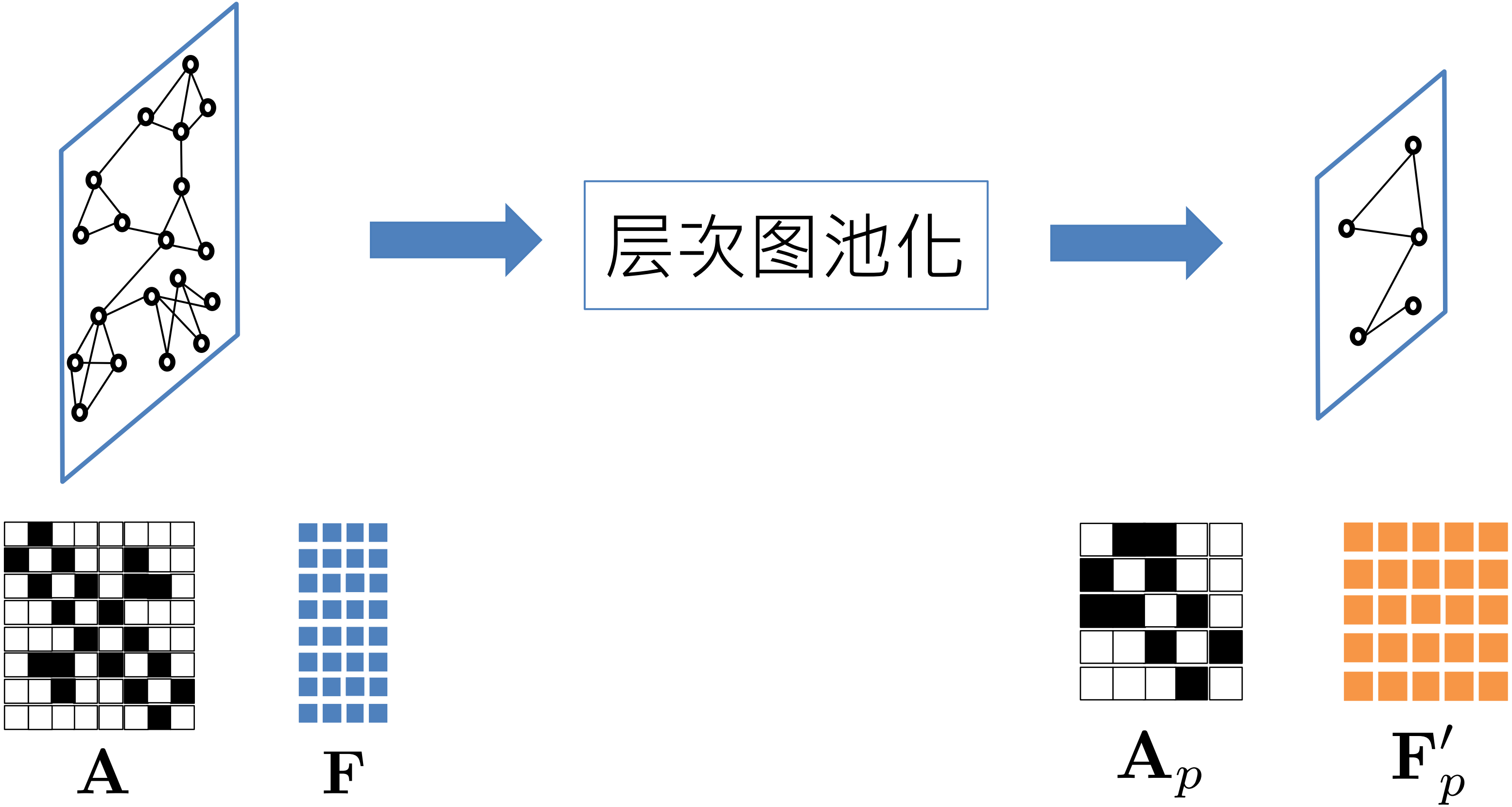
$\mathbf{F}$

$$\boldsymbol{f}_{\mathcal{G}} = \sum_{v_i \in \mathcal{V}} s_i \cdot \tanh\left(\mathbf{F}_i \boldsymbol{\Theta}_{ip}\right)$$

$$s_i = \frac{\exp\left(h\left(\mathbf{F}_i\right)\right)}{\sum_{v_j \in \mathcal{V}} \exp\left(h\left(\mathbf{F}_j\right)\right)}$$

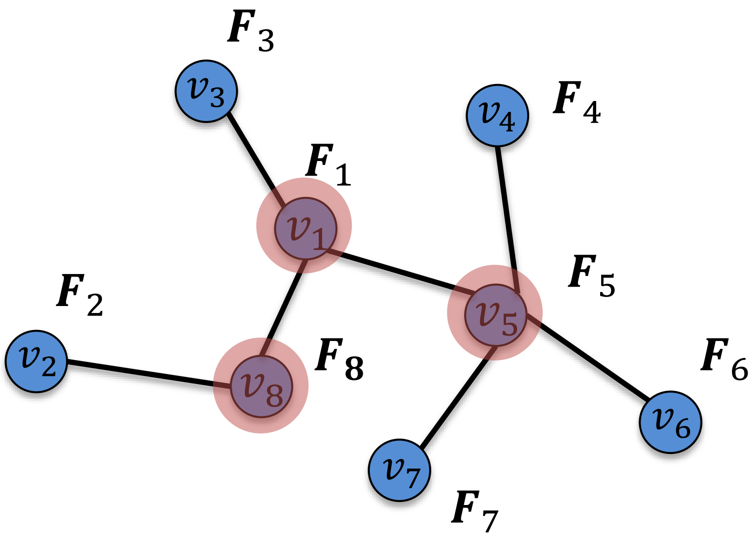
$\mathbf{f}_{\mathcal{G}}$

# 层次图化操作



# gPool

选择最重要的节点来作为粗化图的节点




生成粗化图

$\mathbf{A}_p$ $\mathbf{F}'_p$

重要性的度量

$$v_i \rightarrow y_i \quad y_i = \frac{\boldsymbol{F}_i^{\top} p}{||p||}$$

选择重要性最高的$n_p$个节点

$$idx = rank(\boldsymbol{y}, n_p)$$

生成粗化图的结构以及备用的节点特征

$$\boldsymbol{A}_p = \boldsymbol{A}[idx, idx]$$

$$\boldsymbol{F}_{inter} = \boldsymbol{F}[idx, :]$$

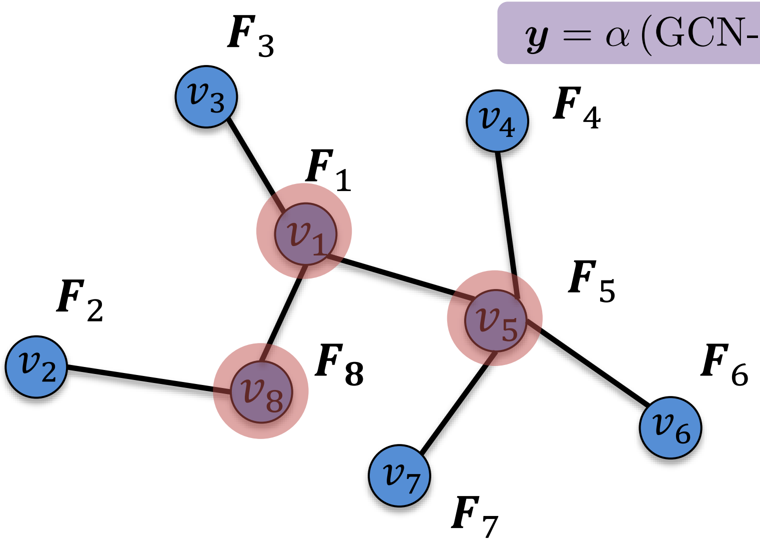
生成粗化图的节点特征

$$\tilde{y} = sigmoid(\boldsymbol{y}[idx])$$

$$\boldsymbol{F}'_p = \boldsymbol{F}_{inter} \odot \tilde{y}$$

# SAGPool

选择最重要的节点来作为粗化图的节点




$\boldsymbol{y} = \alpha\left(\text{GCN-Filter}\left(\boldsymbol{A}, \boldsymbol{F}\right)\right)$

重要性的度量

$v_i \to y_i$ $\quad y_i = \frac{\boldsymbol{F}_i^\top p}{||p||}$

选择重要性最高的$n_p$个节点

$$idx = rank(\boldsymbol{y}, n_p)$$

生成粗化图的结构以及备用的节点特征

$$\boldsymbol{A}_p = \boldsymbol{A}[idx, idx]$$

$$\boldsymbol{F}_{inter} = \boldsymbol{F}[idx, :]$$

生成粗化图

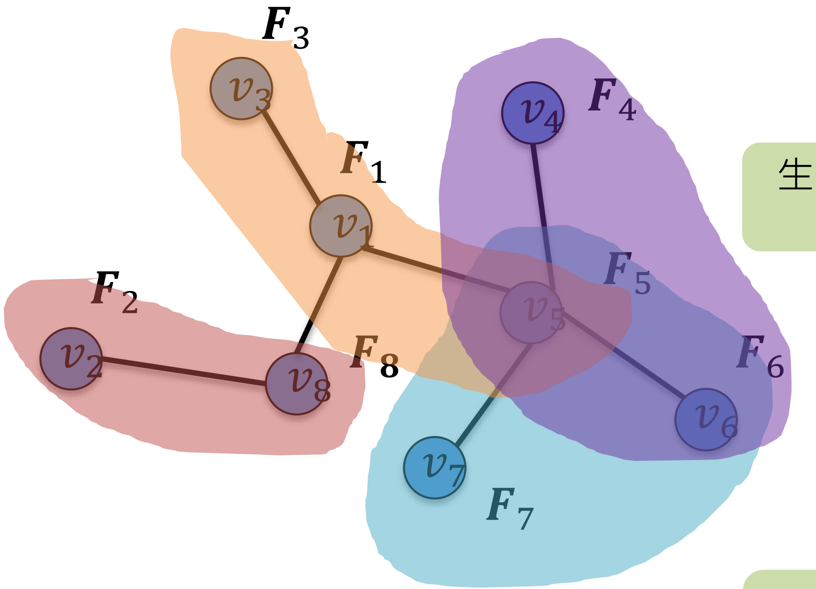
$\mathbf{A}_p$ $\quad \mathbf{F}'_p$

生成粗化图的节点特征

$$\tilde{y} = sigmoid(\boldsymbol{y}[idx])$$

$$\boldsymbol{F}'_p = \boldsymbol{F}_{inter} \odot \tilde{y}$$

利用GNN生成子图来作为粗化图的节点




生成粗化图

$\mathbf{A}_p$ $\mathbf{F}'_p$

两个GCN模型

生成一个分配矩阵

$$\boldsymbol{S} = \mathrm{softmax}\left(\mathrm{GCN}\left(\boldsymbol{A}, \boldsymbol{F}\right)\right)$$

$$\mathbf{S} \in \mathbb{R}^{N \times n_p}$$

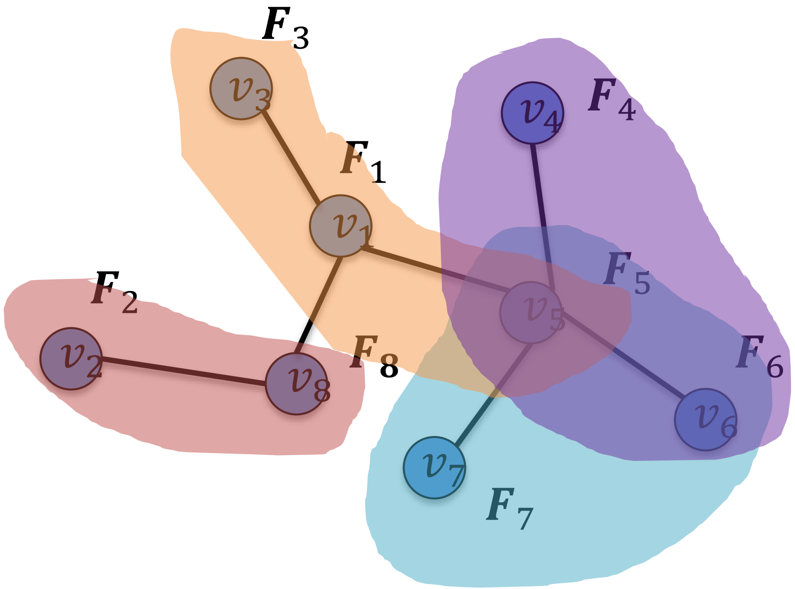
$S_{ij}$表示把节点$v_i$分配给第$j$个子图的部分

生成备用节点特征

$$\boldsymbol{F}_{inter} = \mathrm{GCN}\left(\boldsymbol{A}, \boldsymbol{F}\right)$$

## DiffPool

利用GNN生成子图来作为粗化图的节点




生成粗化图

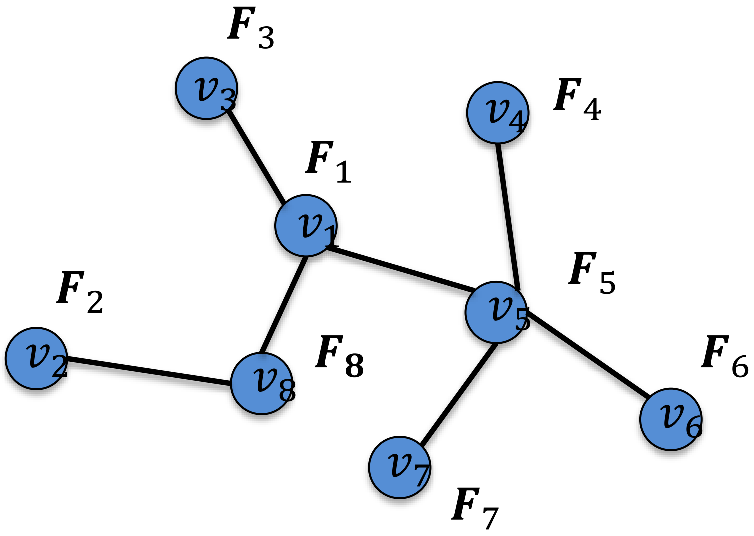
$\mathbf{A}_p$ $\mathbf{F}'_p$

利用学到的$\boldsymbol{S}$和$\boldsymbol{F}_{inter}$生成$\boldsymbol{A}_p$和$\boldsymbol{F}'_p$

$$\mathbf{A}_p = \mathbf{S}^\top \mathbf{A}\mathbf{S}$$

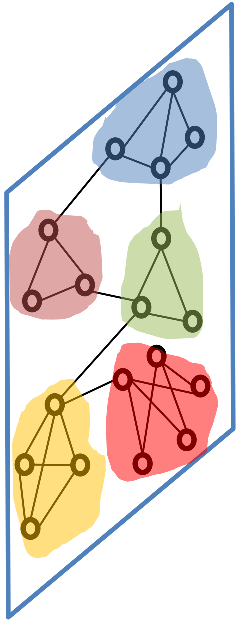
$$\mathbf{F}'_p = \mathbf{S}^\top \mathbf{F}_{inter}$$

# Eigenpooling




生成粗化图

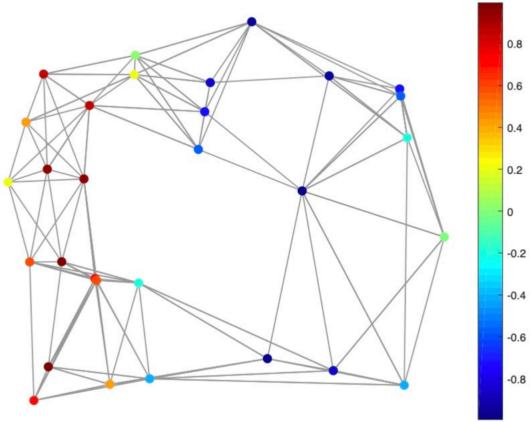
$\mathbf{A}_p$ $\mathbf{F}'_p$



利用聚类方法学习到子图作为粗化图的节点，并生成相应的 $A_p$

- ❑ 主要关注学习$\boldsymbol{F}'_p$的过程
- ❑ 学习 $\boldsymbol{F}'_p$ 的过程中需要捕捉结构和节点特征信息

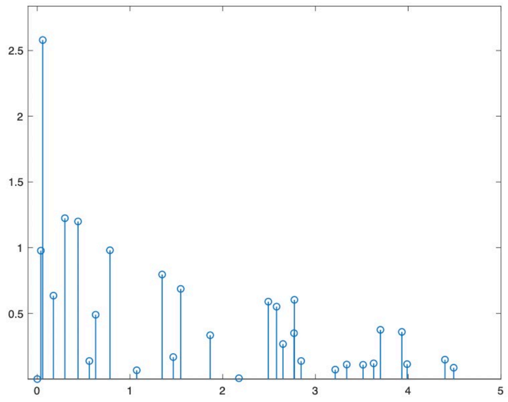
# 傅立叶变换的简要回顾




空间域

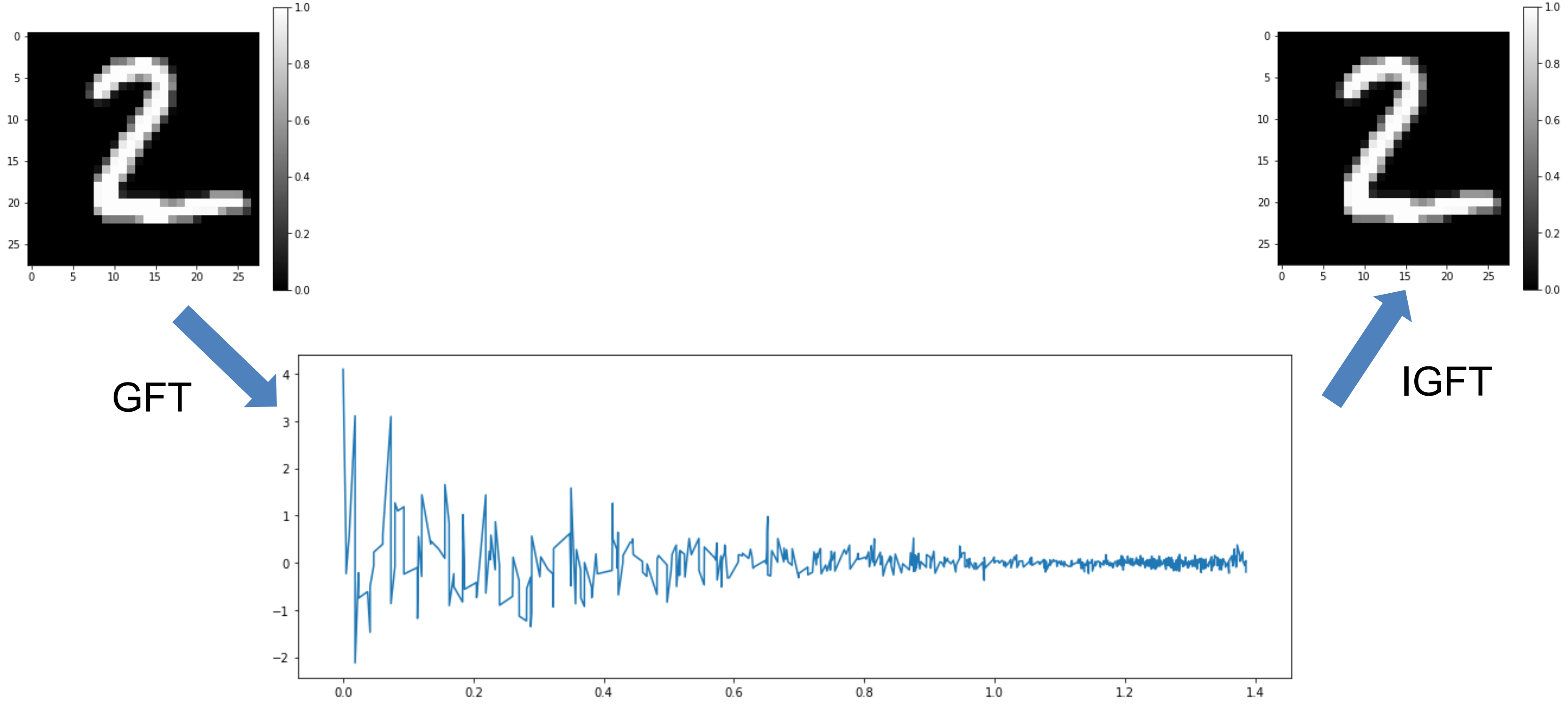
傅立叶变换

傅立叶逆变换



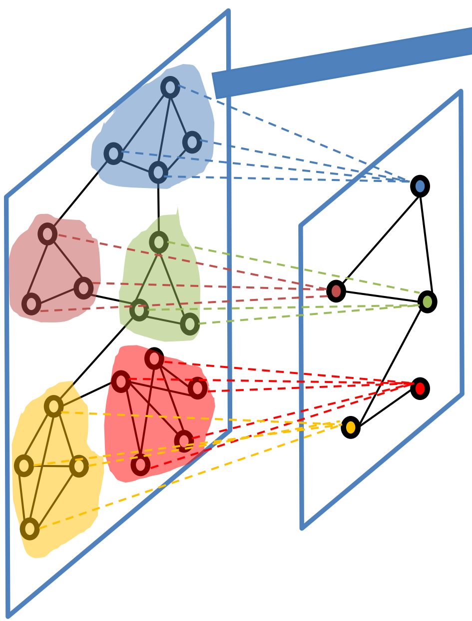

谱域

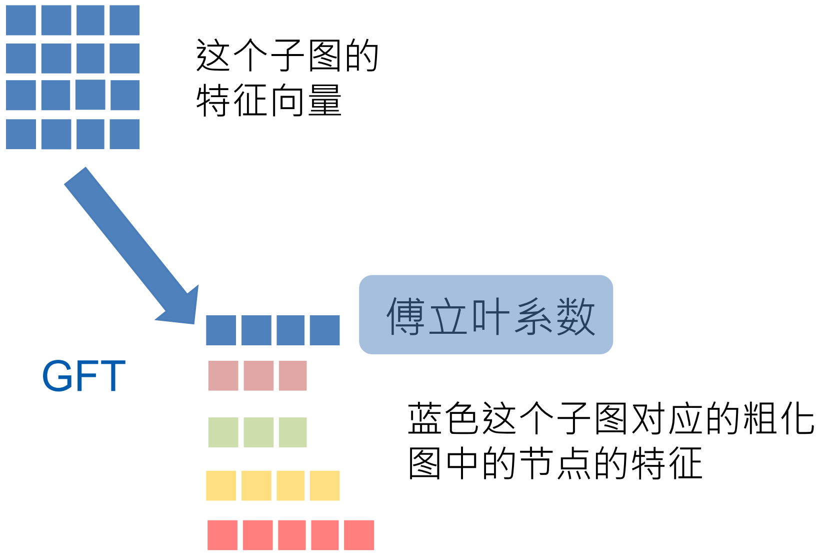
# 信号重建的一个示例




图傅立叶系数

# Eigenpooling：用傅立叶系数作为特征



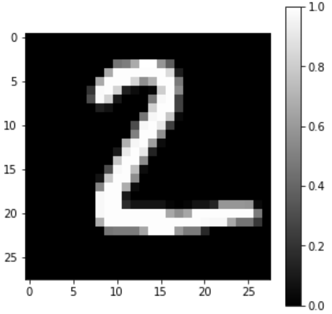



这些子图可能包含不同数量的节点

计算所有的特征向量可能比较费时

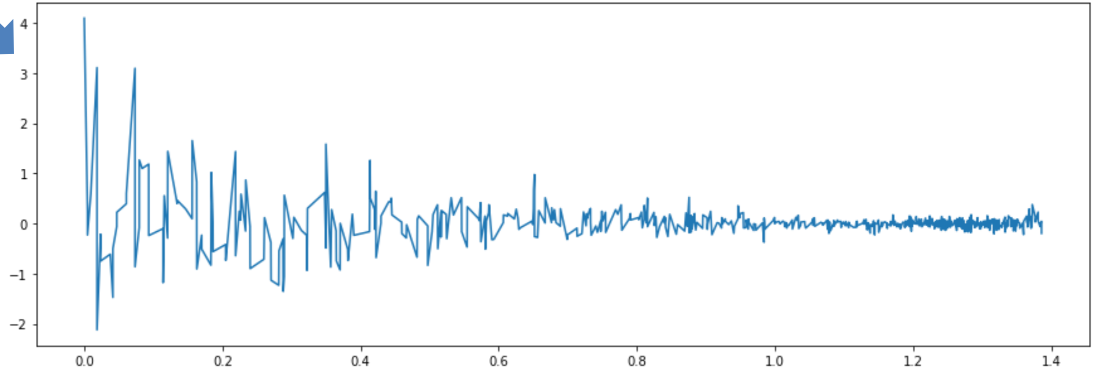
# 信号重建的一个示例

真的需要用到所有的傅立叶系数来重建一个“不错”的信号吗?

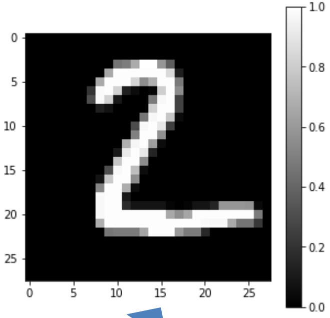


GFT

图傅立叶系数

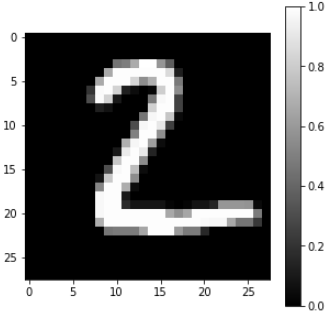


IGFT



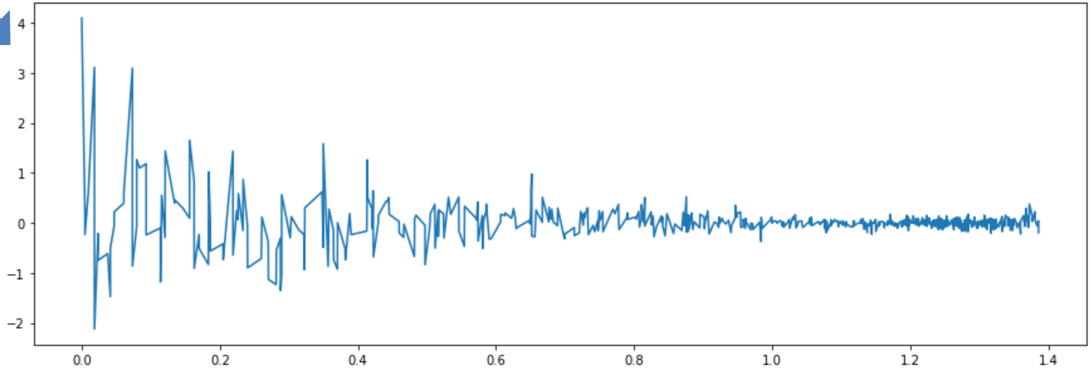
# 信号重建的一个示例

真的需要用到所有的傅立叶系数来重建一个“不错”的信号吗?

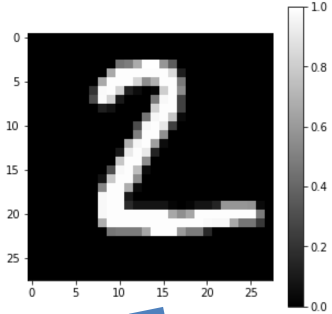



GFT

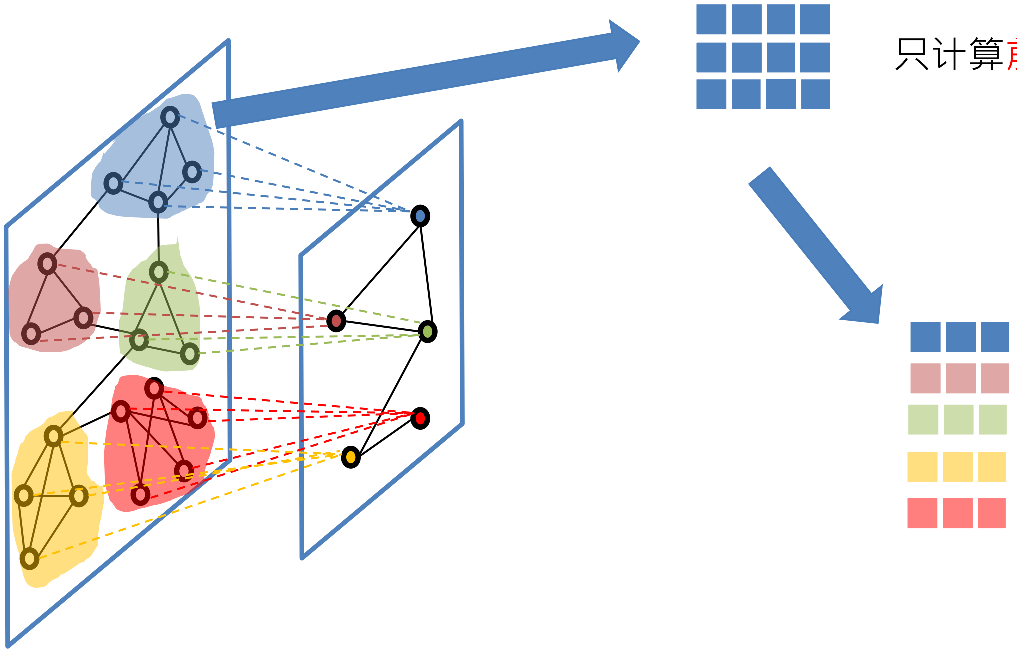
截断的图傅立叶系数




IGFT



# Eigenpooling：用截断的傅立叶系数作为特征



只计算前K个特征向量

截断的傅立叶系数

# 目录