Graph Convolutional Network

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}\,, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2 = f(X)^\top \Delta f(X)$$

刘朝阳 2019.3.21

Learning Convolutional Neural Networks for Graphs ICML2016

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering NIPS 2016

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS ICLR2017

Semi-Supervised Learning with Graphs

Graph Neural Networks: A Review of Methods and Applications arXiv 2019.3.7

https://github.com/thunlp/GNNPapers

Survey papers

1. **Graph Neural Networks: A Review of Methods and Applications.** *Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Maosong Sun.* 2018. paper

2. **A Comprehensive Survey on Graph Neural Networks.** *Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, Philip S. Yu.* 2019. paper

3. **Deep Learning on Graphs: A Survey.** *Ziwei Zhang, Peng Cui, Wenwu Zhu.* 2018. paper

4. **Relational Inductive Biases, Deep Learning, and Graph Networks.** *Battaglia, Peter W and Hamrick, Jessica B and Bapst, Victor and Sanchez-Gonzalez, Alvaro and Zambaldi, Vinicius and Malinowski, Mateusz and Tacchetti, Andrea and Raposo, David and Santoro, Adam and Faulkner, Ryan and others.* 2018. paper

5. **Geometric Deep Learning: Going beyond Euclidean data.** *Bronstein, Michael M and Bruna, Joan and LeCun, Yann and Szlam, Arthur and Vandergheynst, Pierre.* IEEE SPM 2017. paper

6. **Computational Capabilities of Graph Neural Networks.** *Scarselli, Franco and Gori, Marco and Tsoi, Ah Chung and Hagenbuchner, Markus and Monfardini, Gabriele.* IEEE TNN 2009. paper

7. **Neural Message Passing for Quantum Chemistry.** *Gilmer, Justin and Schoenholz, Samuel S and Riley, Patrick F and Vinyals, Oriol and Dahl, George E.* 2017. paper

8. **Non-local Neural Networks.** *Wang, Xiaolong and Girshick, Ross and Gupta, Abhinav and He, Kaiming.* CVPR 2018. paper

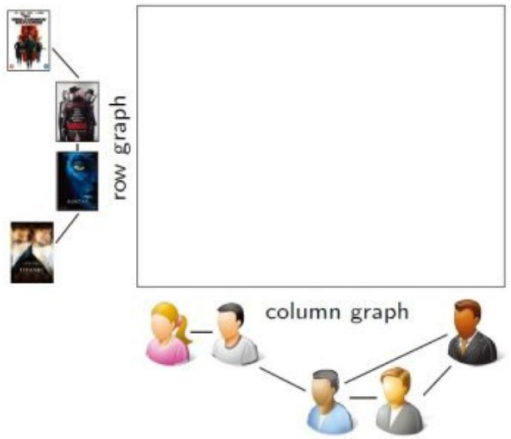
9. **The Graph Neural Network Model.** *Scarselli, Franco and Gori, Marco and Tsoi, Ah Chung and Hagenbuchner, Markus and Monfardini, Gabriele.* IEEE TNN 2009. paper

Learning Convolutional Neural Networks for Graphs ICML2016

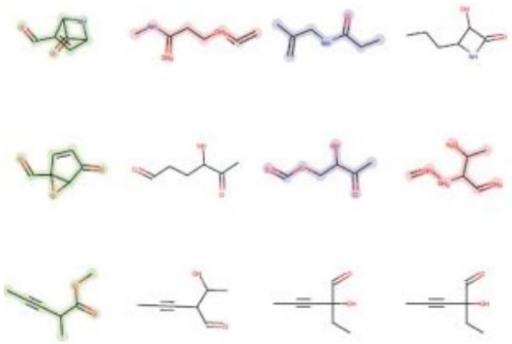
Importance

Numerous important problems can be framed as learning from graph data.

Social Network



Chemical Compound



Protein

Knowledge Graphs

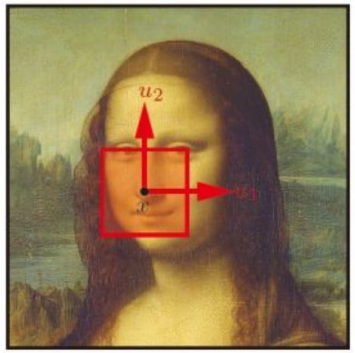
脑网络分析指标

指标名称	常用符号	指标描述
节点属性		
度	k_i	直接连接在一个节点的边的个数
节点效率	e_i	衡量一个节点与其他节点通信的效率
中间中心度	b_i	一个节点对网络中其他节点的信息流的影响
∘ ∘ ∘ ∘ ∘	∘ ∘ ∘	∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘ ∘

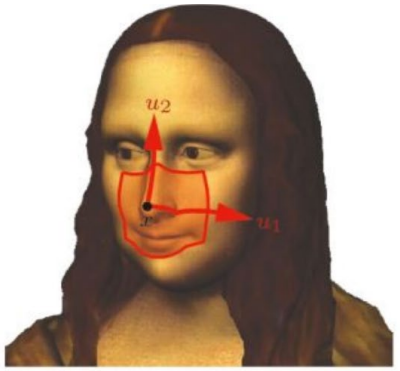
CV

NLP

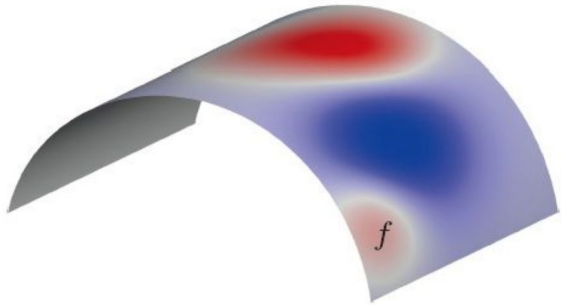
RL

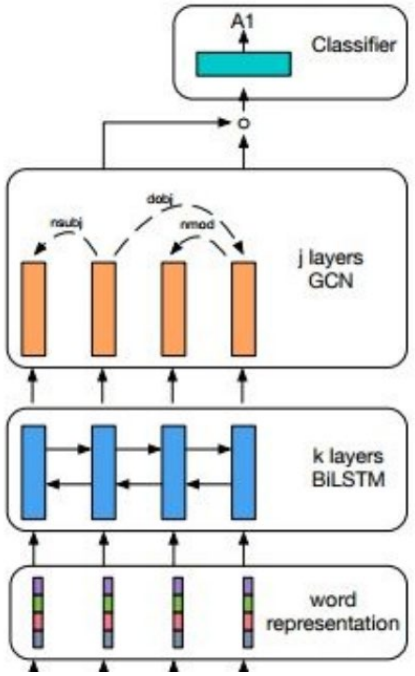


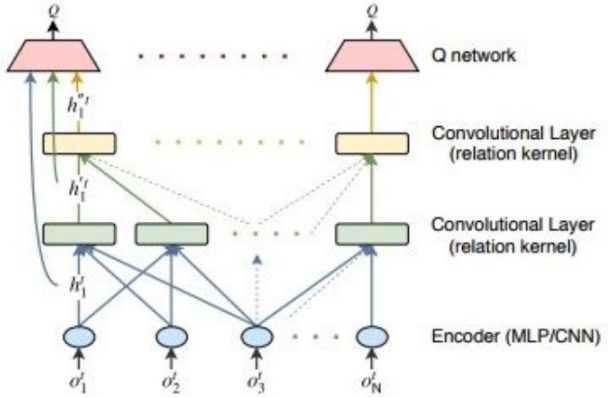
Image



Manifold







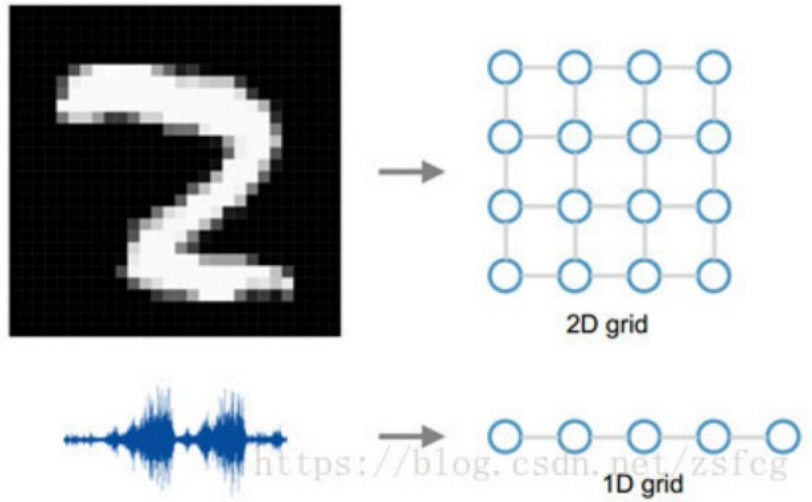
Task

- Given a collection of graphs, learn a function that can be used for classification and regression problems on unseen graphs.

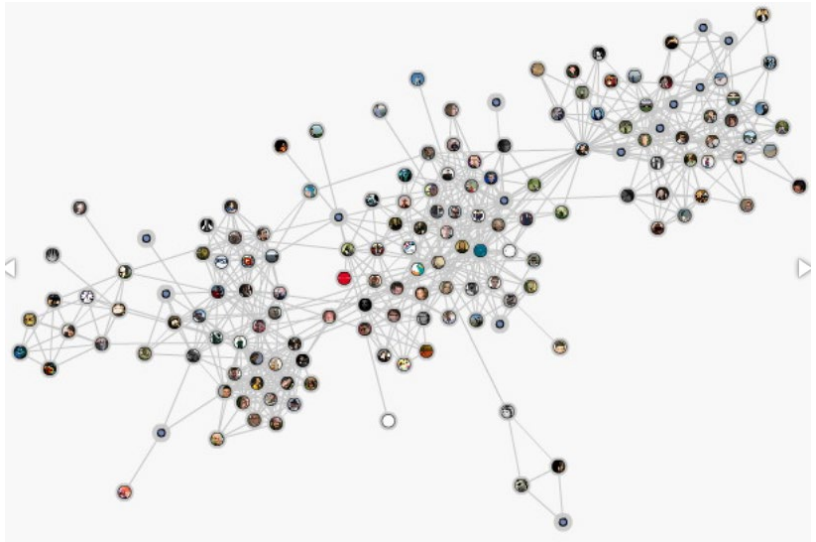
- Given a large graph, learn graph representations that can be used to infer unseen graph properties such as node types and missing edges.

Euclidean Data

Implicit spatial order of the pixels, the same holds for NLP problems where each sentence (and its parse-tree) determines a sequence of words.



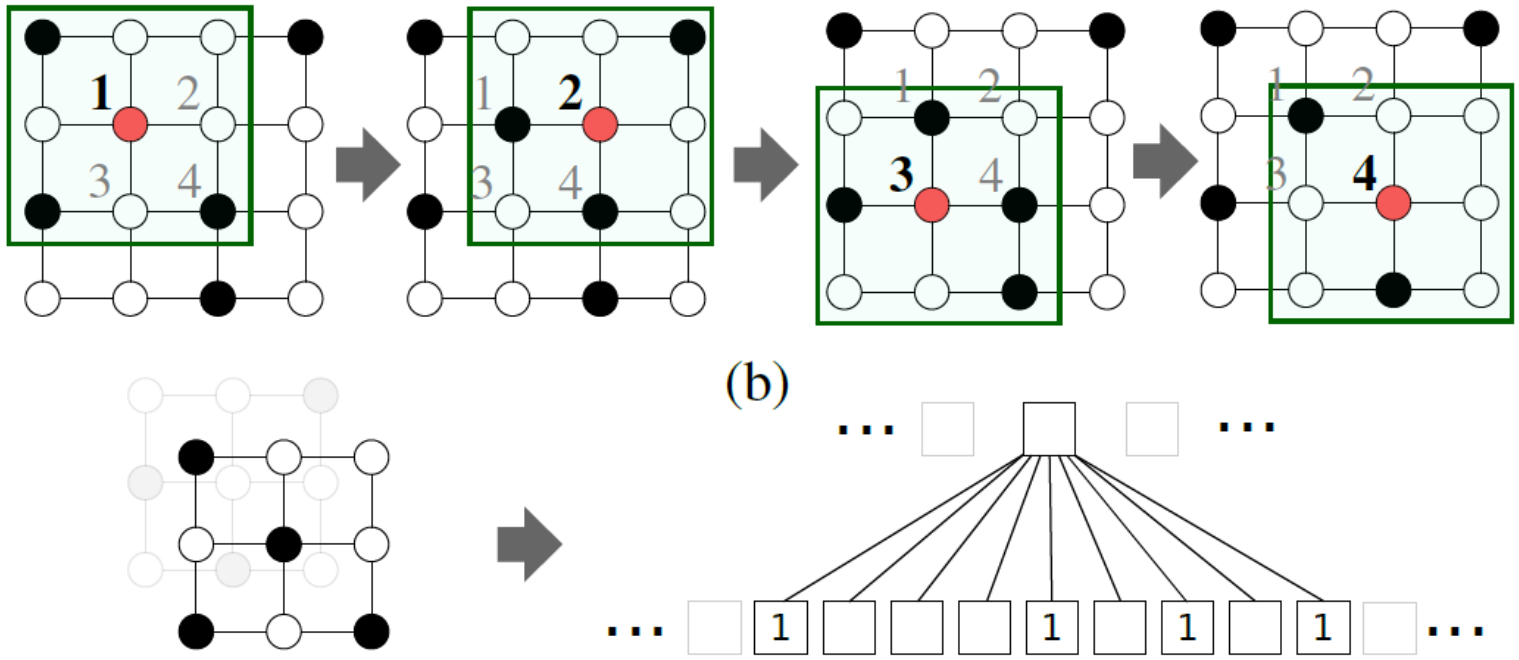
However, for numerous graph collections a **problem-specific ordering** (spatial, temporal, or otherwise) is **missing** and **the nodes of the graphs are not in correspondence**.



Has to solve two problems

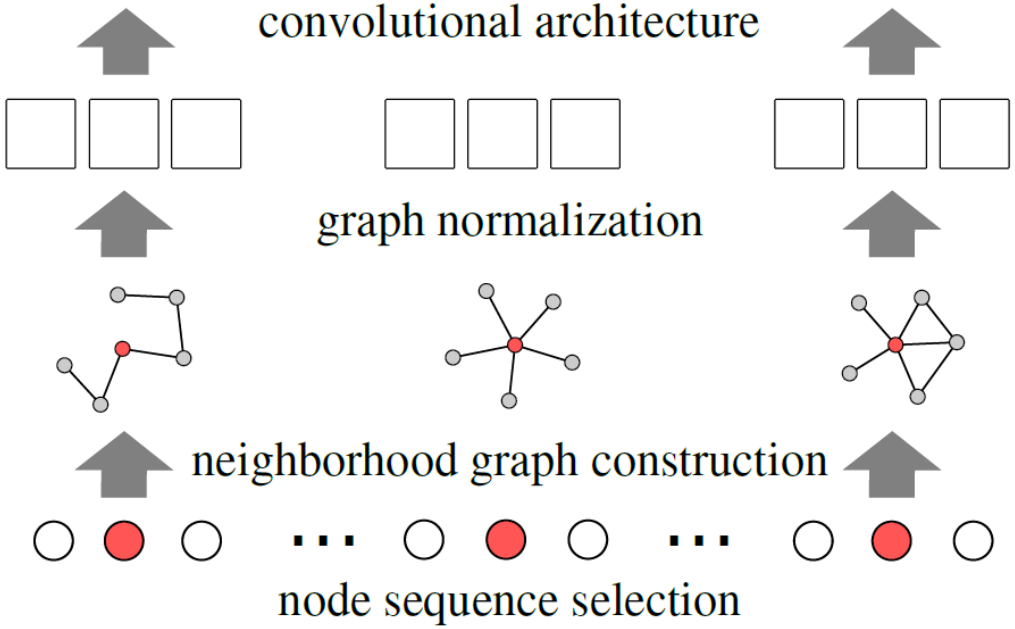
Determining the node sequences for which neighborhood graphs are created.

Computing a normalization of neighborhood graphs.



First, it is highly efficient, naively parallelizable, and applicable to large graphs.

Second, supports feature visualizations, providing insights into the structural properties of graphs.



An illustration of the proposed architecture

Third, instead of crafting yet another graph kernel, PATCHY-SAN learns application dependent features without the need to feature engineering.

PATCHY-SAN utilizes graph labelings to impose an order on nodes.

A graph labeling procedure computes a graph labeling for an input graph.

$$\ell: V \to S$$

A ranking is a function $\mathbf{r}: V \to \{1, \ldots, |V|\}$, Every labeling induces a ranking $\mathbf{r}$ with $\mathbf{r}(u) < \mathbf{r}(v)$ if and only if $\ell(u) > \ell(v)$

Examples of graph labeling

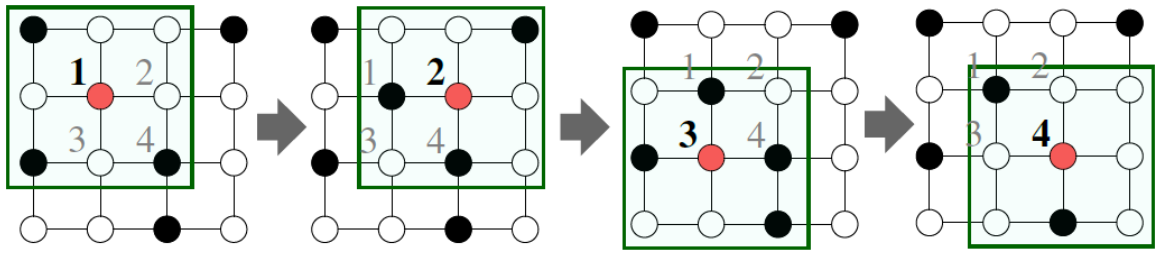
Node degree and other measures of centrality(degree, page-rank, eigenvector centrality, etc) commonly used in the analysis of networks.

The **betweeness centrality** of a vertex v computes the fractions of shortest paths that pass through v.

Node sequence selection

Algorithm 1 SELNODESEQ: Select Node Sequence

input: graph labeling procedure ℓ, graph $G = (V, E)$, stride s, width w, receptive field size k
$V_{\texttt{sort}}$ = top w elements of V according to ℓ
$i = 1, j = 1$
while $j < w$ **do**
  if $i \leq |V_{\texttt{sort}}|$ **then**
    $\mathsf{f} = \textsc{ReceptiveField}(V_{\texttt{sort}}[i])$
  else
    $\mathsf{f} = \textsc{ZeroReceptiveField}()$
  apply f to each input channel
  $i = i + s, j = j + 1$

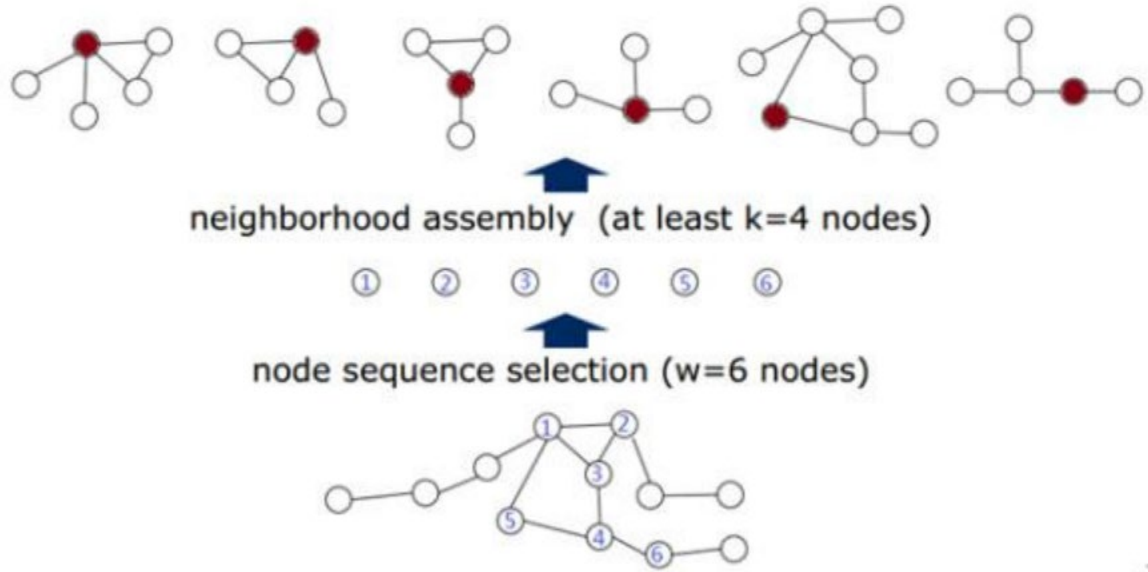


Algorithm 3 RECEPTIVEFIELD: Create Receptive Field

input: vertex v, graph labeling ℓ, receptive field size k
$N = \textsc{NeighAssemb}(v, k)$
$G_{\texttt{norm}} = \textsc{NormalizeGraph}(N, v, \ell, k)$
return $G_{\texttt{norm}}$

Algorithm 2 NEIGHASSEMB: Neighborhood Assembly

input: vertex v, receptive field size k
output: set of neighborhood nodes N for v
$N = [v]$
$L = [v]$
while $|N| < k$ and $|L| > 0$ **do**
  $L = \bigcup_{v \in L} N_1(v)$
  $N = N \cup L$
return the set of vertices N



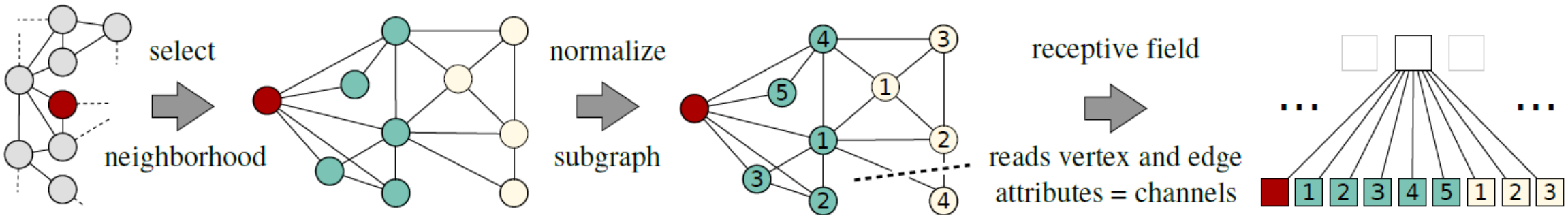
adjacency matrices under labeling

$$\arg\min_{\ell} \mathbb{E}_{\mathcal{G}} \left[\left| \mathbf{d}_{\mathbf{A}} \left(\mathbf{A}^{\ell}(G), \mathbf{A}^{\ell}(G') \right) - \mathbf{d}_{\mathbf{G}}(G, G') \right| \right]$$

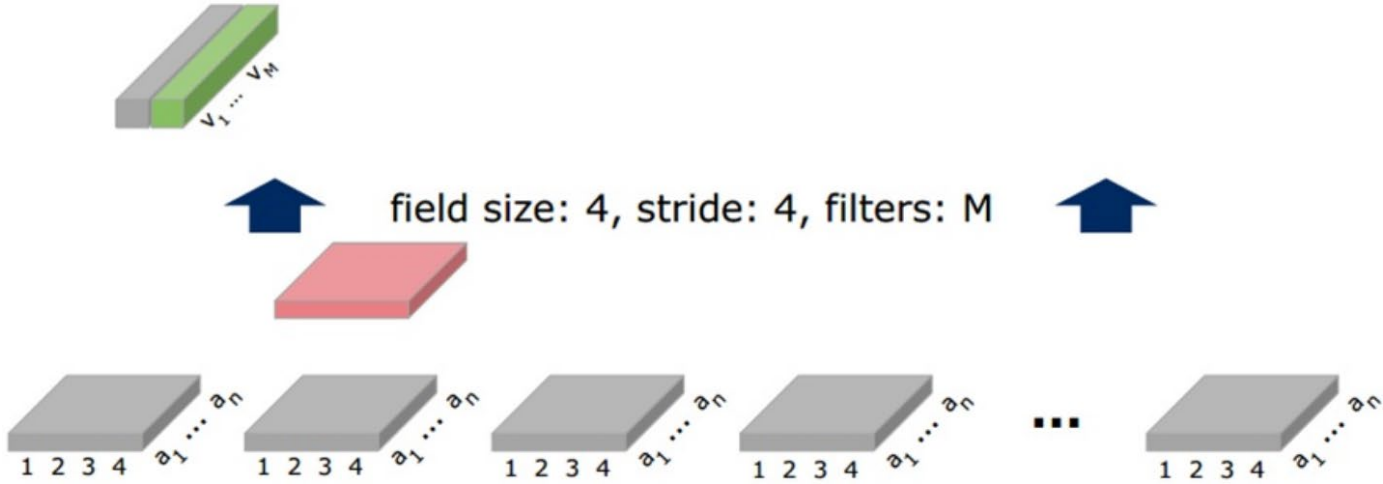
labeling method (centrality etc.)

distance measures in matrix and graph space

The normalization is performed for each of the graphs induced on the neighborhood of a root node v.



a_v the number of vertex attributes, a_e the number of edge attributes. For each graph G,it applies normalized receptive fields and results in one (w, k, a_v) and one (w, k, k, a_e) tensor. These can be reshaped to a (wk, a_v) and a (w, k^2, a_e) tensors.



提取图空间特征的两种方式　　spatial domain　　**spectral domain**

Spectral graph theory　　借助于图的拉普拉斯矩阵的特征值和特征向量来研究图的性质

GSP(graph signal processing)

Graph Fourier Transformation

Graph convolution

Graph Convolutional Network

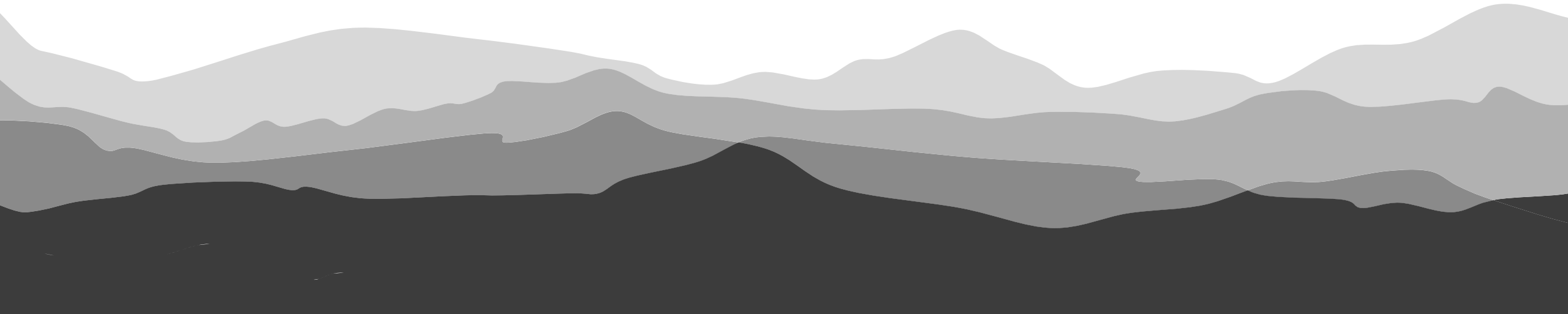
CNN无法处理Non Euclidean Structure的数据，学术上的表达是传统的离散卷积在Non Euclidean Structure的数据上无法保持平移不变性。

由于CNN无法处理Non Euclidean Structure的数据，又希望在这样的数据结构（拓扑图）上有效地提取空间特征来进行机器学习。

广义上来讲任何数据在赋范空间内都可以建立拓扑关联，谱聚类就是应用了这样的思想。所以说拓扑连接是一种广义的数据结构，GCN有很大的应用空间。

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering NIPS 2016

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS ICLR2017



什么是(离散)卷积？CNN中卷积发挥什么作用？

什么是拉普拉斯矩阵？为什么GCN要用拉普拉斯矩阵？

拉普拉斯矩阵的谱分解（特征分解）及其性质？

如何从传统的傅里叶变换、卷积类比到Graph上的傅里叶变换及卷积？

为什么拉普拉斯矩阵的特征向量可以作为傅里叶变换的基？特征值表示频率？

如何通过反向传播优化GCN中的卷积核参数？

很多graph neural network models存在共性，如卷积过滤器参数在图的所有位置共享，暂且都称作Graph Convolutional Networks(GCNs)

The goal is then to learn a function of signals/features on a graph $G(V, E)$

Input

A feature description x_i for every node i,X: $N \times D$.

A representative description of the graph structure in matrix form.

Output

Z: $N \times F$.

$$H^{(l+1)} = f(H^{(l)}, A) \qquad f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

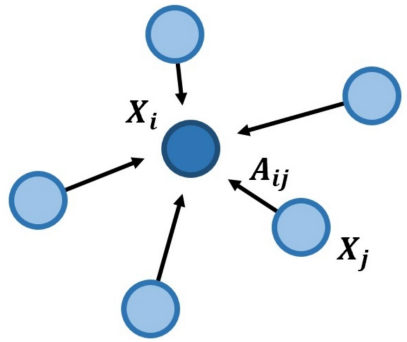
Despite its simplicity this model is already quite powerful.

图卷积的核心思想是利用边的信息对节点进行聚合，从而生成新的节点表示。

加权平均法

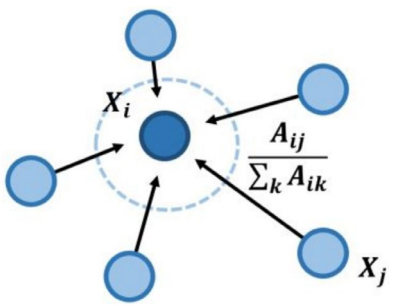
$$aggregate(X_i) = AX = \Sigma_{j=1}^{N} A_{ij} X_j$$

$$aggregate(X_i) = (A+I)X = \Sigma_{j=1}^{N} A_{ij} X_j + 1 \times X_i$$



归一化的加权平均法

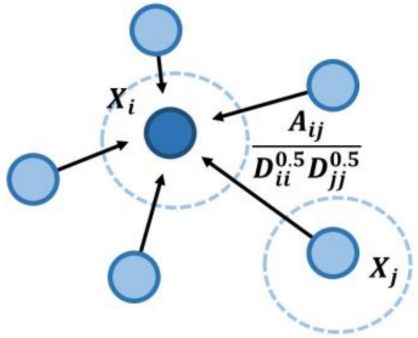
$$\begin{aligned} aggregate(X_i) &= D^{-1}(A+I)X \\ &= \Sigma_{k=1}^{N} D_{ik}^{-1} \Sigma_{j=1}^{N} A_{ij} X_j + \Sigma_{k=1}^{N} D_{ik}^{-1} \times X_i \\ &= \Sigma_{j=1}^{N} D_{ii}^{-1} A_{ij} X_j + D_{ii}^{-1} \times X_i \\ &= \Sigma_{j=1}^{N} \frac{A_{ij}}{D_{ii}} X_j + \frac{1}{D_{ii}} X_i \\ &= \Sigma_{j=1}^{N} \frac{A_{ij}}{\Sigma_{k=1}^{N} A_{ik}} X_j + \frac{1}{D_{ii}} X_i \end{aligned}$$

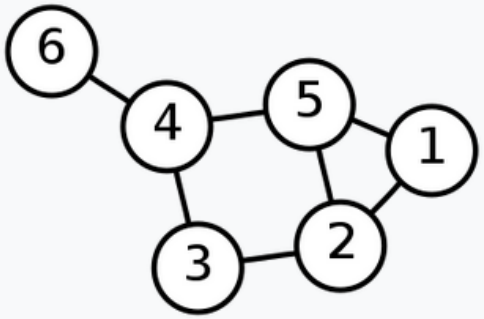


离群较远或者度较小的节点在聚合后特征较小

对称归一化的加权平均法

$$
\begin{aligned}
aggregate(X_i) =& D^{-0.5}\tilde{A}D^{-0.5}X \\
=& \Sigma_{k=1}^{N} D_{ik}^{-0.5} \Sigma_{j=1}^{N} \tilde{A}_{ij} X_j \Sigma_{l=1}^{N} D_{il}^{-0.5} \\
=& \Sigma_{j=1}^{N} D_{ii}^{-0.5} A_{ij} X_j D_{jj}^{-0.5} \\
=& \Sigma_{j=1}^{N} \frac{1}{{D_{ii}}^{0.5}} A_{ij} \frac{1}{{D_{jj}}^{0.5}} X_j
\end{aligned}
$$



Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Combinatorial Laplacian $L = D - A$

Symmetric normalized Laplacian $L^{sys} = D^{-1/2} L D^{-1/2}$

拉普拉斯矩阵可以进行特征分解（谱分解）

不是所有的矩阵都可以特征分解，其充要条件为**n阶方阵存在n个线性无关的特征向量**

拉普拉斯矩阵是半正定实对称矩阵

- 对称矩阵一定n个线性无关的特征向量
- 半正定矩阵的特征值一定非负
- 对称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。

$$L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^{-1} \quad U = (\overrightarrow{u_1}, \overrightarrow{u_2}, \cdots, \overrightarrow{u_n}) \qquad L = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^{T}$$

Spectral filtering of graph signals

We cannot express a meaningful translation operator in the vertex domain, the convolution operator on graph G is defined in the Fourier domain such that

$$x *_{\mathcal{G}} y = U((U^T x) \odot (U^T y))$$

It follows that a signal x is filtered by g_θ as $y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = U g_\theta(\Lambda)U^T x$

where the parameter $\theta \in R^n$ is a vector of Fourier coefficients. $g_\theta(\Lambda) = \mathrm{diag}(\theta)$

SPECTRAL GRAPH CONVOLUTIONS

We consider spectral convolutions on graphs defined as the multiplication of a signal $x \in R^N$(a scalar for every node) with a filter $g_\theta = \mathrm{diag}(\theta)$ parameterized by $\theta \in R^N$ in the Fourier domain.

$$g_\theta \star x = U g_\theta U^\top x$$

什么是(离散)卷积？CNN中卷积发挥什么作用

https://www.zhihu.com/people/matongxue/answers

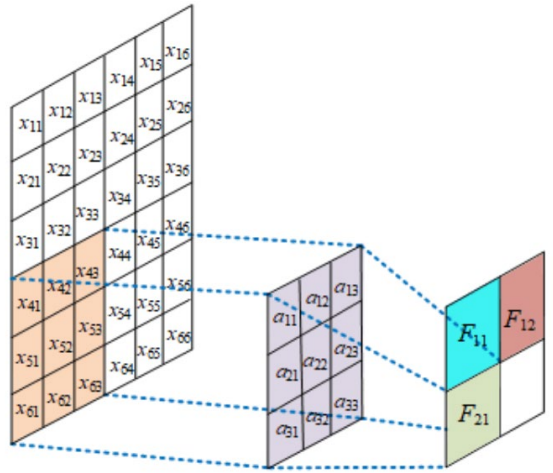
$$(f*g)(n)=\int_{-\infty}^{\infty}f(\tau)g(n-\tau)d\tau \qquad (f*g)(n)=\sum_{\tau=-\infty}^{\infty}f(\tau)g(n-\tau)$$

$$f(1)g(3)+f(2)g(2)+f(3)g(1) \qquad (f*g)(4)=\sum_{m=1}^{3}f(4-m)g(m)$$

离散卷积本质可以看成一种加权求和

CNN中的卷积

利用一个共享参数的过滤器（kernel），通过计算中心像素点以及相邻像素点的加权和来构成feature map实现空间特征的提取。



卷积核的参数通过优化求出才能实现特征提取的作用，GCN的理论很大一部分工作就是为了引入可以优化的卷积参数。

加权和、卷积及傅立叶变换三者之间的联系

GSP(graph signal processing)

信号处理是着力于改变信号的频谱，也就是说，先对信号进行傅里叶变换，然后在频域进行处理，之和进行傅里叶逆变换得到处理过后的信号。

Graph Fourier Transformation

将两个数列经过傅里叶变换后在频域上做点积后再做逆傅里叶变换回到时域，其结果与两个向量的卷积是相同的。

Graph convolution

Graph Convolutional Network

Graph Fourier Transformation

传统傅里叶变换定义 $$F(\omega) = \mathcal{F}[f(t)] = \int f(t)e^{-i\omega t}dt$$

拉普拉斯矩阵就是离散拉普拉斯算子

广义特征方程定义： $AV = \lambda V$ $$\Delta e^{-i\omega t} = \frac{\partial^2}{\partial t^2}e^{-i\omega t} = -\omega^2 e^{-i\omega t}$$

在Graph上傅里叶变换的矩阵形式为

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

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

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

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

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，也就是对于函数$f(t)h(t)$两者的卷积是其函数傅里叶变换乘积的逆变换 。

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

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

卷积核h的傅里叶变换写成对角矩阵的形式为：

$$\begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix}$$

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

$$\begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^T f$$

再乘以 U 求两者傅立叶变换乘积的逆变换，则求出卷积

$$(f * h)_G = U \begin{pmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{pmatrix} U^T f$$

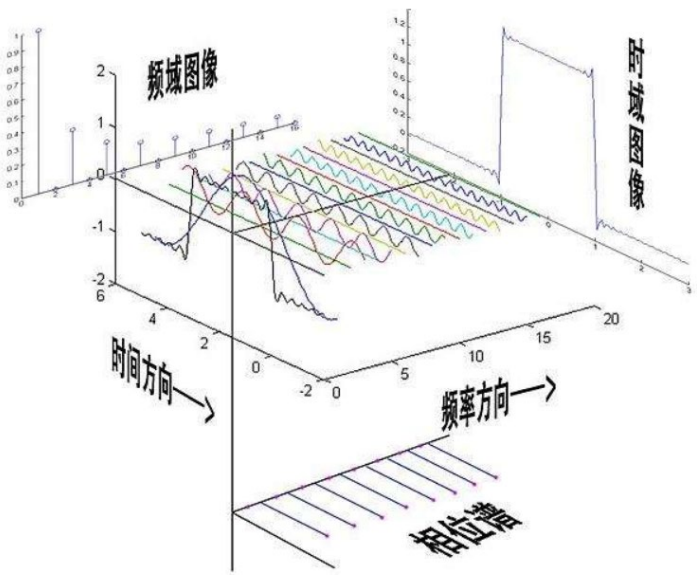
$$(f*h)_G = U\begin{pmatrix}\hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n)\end{pmatrix}U^T f$$

$$(f*h)_G = U((U^T h)\odot(U^T f))$$

小结　　为什么拉普拉斯矩阵的特征向量可以作为傅里叶变换的基？特征值表示频率？

傅里叶变换一个本质理解就是：把任意一个函数表示成了若干个正交函数（由谐波函数构成）的线性组合。

$$f = \hat{f}(\lambda_1)u_1 + \hat{f}(\lambda_2)u_2 + \cdots + \hat{f}(\lambda_n)u_n$$



n 维空间中 n 个线性无关的向量可以构成空间的一组基，而且拉普拉斯矩阵的特征向量还是一组正交基。

Spectral Networks and Locally Connected Networks on Graphs

$$y_{output} = \sigma\left(U \begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{pmatrix} U^T x\right)$$

缺点

每一次前向传播，都要计算$U\ diag(\theta_l)\ U^T$三者的乘积，特别对于大规模的图，计算代价高，计算复杂度为$O(n^2)$；卷积核参数有 n 个；没有spatial localization。

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering NIPS 2016

$$y_{output} = \sigma\left(U \begin{pmatrix} \sum_{j=0}^{K} \alpha_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^{K} \alpha_j \lambda_n^j \end{pmatrix} U^T x\right)$$

$$\begin{pmatrix} \sum_{j=0}^{K} \alpha_j \lambda_1^j & & \\ & \ddots & \\ & & \sum_{j=0}^{K} \alpha_j \lambda_n^j \end{pmatrix} = \sum_{j=0}^{K} \alpha_j \Lambda^j$$

$$U \sum_{j=0}^{K} \alpha_j \Lambda^j U^T = \sum_{j=0}^{K} \alpha_j U \Lambda^j U^T = \sum_{j=0}^{K} \alpha_j L^j$$

$$y_{output} = \sigma\left(\sum_{j=0}^{K} \alpha_j L^j x\right)$$

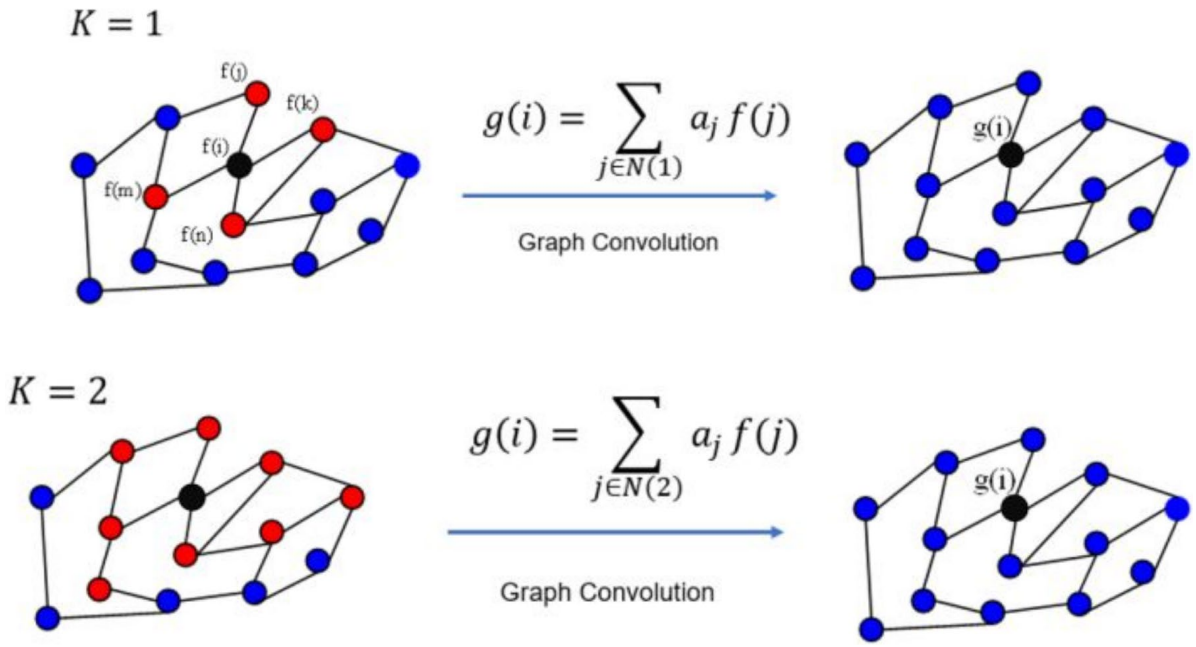
优点

卷积核只有K个参数，一般 K远小于n

卷积核具有很好的spatial localization

矩阵变换后，发现不需要做特征分解了，直接用拉普拉斯矩阵 L进行变换，计算复杂度为$O(n)$

K关系到卷积核的receptive field，每次卷积会将中心定点K-hop neighbor上的feature进行加权求和。



The value at vertex j of the filter g_θ centered at vertex i is given by $\sum_k \theta_k (L^k)_{i,j}$,$d_G(i,j) > K$ implies $(L^k)_{i,j} = 0$, where $d_G(i,j)$ is the shortest path distance.

Wavelets on Graphs via Spectral Graph Theory 2011(Lemma 5.2,5.3,5.4)

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = U g_\theta(\Lambda)U^T x$$

A solution to this problem is to parametrize $g_\theta(L)$ as a polynomial function that can be computed recursively from L。

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

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda})$$

$\theta \in R^K$ is a vector of Chebyshev coefficients, $\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n$

定义7.3 设$f(x)$，$g(x) \in C[a,b]$ 若

$$(f, g) = \int_a^b \rho(x) f(x) g(x) dx = 0$$

则称$f(x)$与$g(x)$在$[a,b]$上带权$\rho(x)$正交。

切比雪夫多项式的性质：

(1) 正交性：

由$\{T_n(x)\}$所组成的序列$\{T_n(x)\}$是在区间[-1, 1]上带权

$$\rho(x) = \frac{1}{\sqrt{1-x^2}}$$

的正交多项式序列。且

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

$$y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})\, x \quad T_k(\tilde{L}) \in R^{n \times n} \quad \tilde{L} = \frac{2L}{\lambda_{max}} - I_n$$

$$\bar{x}_k = T_k(\tilde{L})x \in R^n \qquad T_k(\tilde{L}) = 2\tilde{L}T_{k-1}(\tilde{L}) - T_{k-2}(\tilde{L}) \qquad y_{output} = \sigma\left(\sum_{j=0}^{K} \alpha_j L^j x\right)$$

$$\bar{x}_k = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2} \qquad \bar{x}_0 = x \qquad \bar{x}_1 = \tilde{L}x$$

$$y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})\, x \quad \Longrightarrow \quad y = g_\theta(L)x = [\bar{x}_0, \dots, \bar{x}_{K-1}]\boldsymbol{\theta}$$

Learning filters

The j^{th} output feature map of the sample s is $y_{s,j} = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(L)x_{s,i} \in R^n$

$x_{s,i}$ are the input feature maps and $F_{in} \times F_{out}$ vectors of Chenyshev coefficients $\theta_{i,j} \in R^K$ are the layer's trainable parameters.

$$\frac{\partial E}{\partial \theta_{i,j}} = \sum_{s=1}^{S} [\bar{x}_{s,i,0}, \ldots, \bar{x}_{s,i,K-1}]^T \frac{\partial E}{\partial y_{s,j}} \qquad \frac{\partial E}{\partial x_{s,i}} = \sum_{j=1}^{F_{out}} g_{\theta_{i,j}}(L) \frac{\partial E}{\partial y_{s,j}}$$

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS ICLR2017

We consider spectral convolutions on graphs defined as the multiplication of a signal $x \in R^N$(a scalar for every node) with a filter $g_\theta = \mathrm{diag}(\theta)$ parameterized by $\theta \in R^N$ in the Fourier domain.

$$g_\theta \star x = U g_\theta U^\top x \qquad g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x \qquad \tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

LAYER-WISE LINEAR MODEL

K = 1

- Recover a rich class of convolutional filter functions by stacking multiple such layers.
- Not limited to the explicit parameterization given by, e.g., the Chebyshev polynomials.
- Can alleviate the problem of overfitting on local neighborhood structures for graphs with very wide node degree distributions.
- For a fixed computational budget, this layer-wise linear formulation allows us to build deeper models.

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x \qquad \tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

Approximate $\lambda_{max} \approx 2$ as we can expect that neural network parameters will adapt to this change in scale during training.

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 (L - I_N) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

In practice, it can be beneficial **to constrain the number of parameters further to address overfitting** and to minimize the number of operations (such as matrix multiplications) per layer in the neural network model.

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$$

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x \qquad \theta = \theta'_0 = -\theta'_1$$

renormalization trick

$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \qquad \tilde{A} = A + I_N \qquad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

generalize

generalize this definition to a signal $X \in R^{N \times C}$ with F filters or features maps as follows:

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta \qquad \Theta \in \mathbb{R}^{C \times F}$$

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right) + H^{(l)}$$

EXAMPLE OF SEMI-SUPERVISED NODE CLASSIFICATION

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big) \qquad \hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} \qquad \mathcal{L} = -\sum_{l\in\mathcal{Y}_L}\sum_{f=1}^{F} Y_{lf}\ln Z_{lf}$$

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

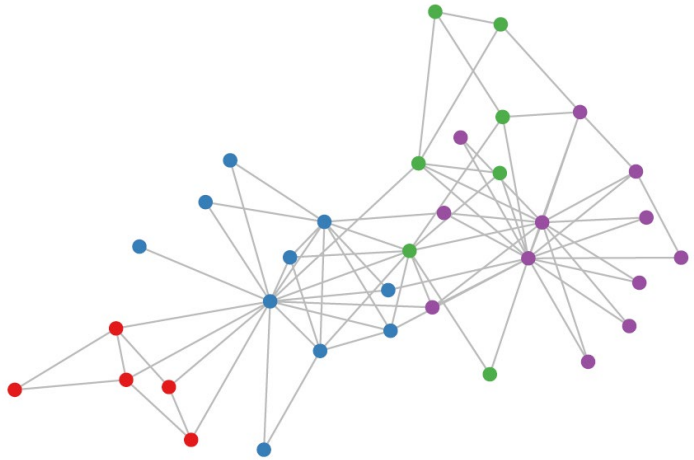
Table 3: Comparison of propagation models.

Description		**Propagation model**	**Citeseer**	**Cora**	**Pubmed**
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1^{st}-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1^{st}-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x$$

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}}AD^{-\frac{1}{2}}x$$

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right)$$



(a) Karate club network