Connected Undirected Weighted Graph

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

Def:

$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}, N = |\mathcal{V}|$

如果有边 $e = (i, j)$，则对称阵 W （在无权图中也相当于邻接矩阵）中 $W_{i,j}$ 表示其权重（记住是无向图哦），否则就为 0。

度的对角矩阵 D 有 $D_{i,i} = \sum_j W_{i,j}$。

Laplacian (∇^2 或 Δ) 是一个标量算子，定义为 divergence of the gradient of function:

$\nabla^2 f = \nabla \cdot \nabla f = \sum_{i=1}^n \frac{\partial^2 f}{\partial x_i^2}, \nabla \cdot f = \mathbf{div} f = \sum_{i=1}^n \frac{\partial f}{\partial x_i}, f : \mathbb{R}^n \Rightarrow \mathbb{R}^n.$

Graph Laplacian

- non-normalized （/combinatorial) graph Laplacian: $L = D - W$。

该 operator 在 Graph 顶点卷积操作 f 为：$(Lf)(i) = \sum_{j \in \mathcal{N}_i} W_{ij}[f(i) - f(j)]$，用于将 vertex domain 转换到 Fourier domain 便于卷积，其中 $\mathcal{N}_i$ 表示顶点 i 的某个邻域内的邻居顶点，结合 D, W 的意义很好理解。

> graph Lapacian 可理解为 Laplacian 的 standard stencil approximation, Ref: Wavelets on graphs via spectral graph theory, formula 13.

- normalized graph Laplacian: $L^{norm} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$，也就是 $(L^{norm} f)(i) = \frac{1}{\sqrt{D(i)}} \sum_{j \in \mathcal{N}_i} W_{ij} [\frac{f(i)}{\sqrt{D(i)}} - \frac{f(j)}{\sqrt{D(j)}}]$。

> 注意 L, L^{norm} 是 **半正定对称矩阵**：一定有 N个线性无关的特征向量 （$UU^\top = I$），特征值一定非负。所以可以 **特征分解 (谱分解)** 为 $L = U\mathtt{diag}(\lambda_1, \cdots, \lambda_N)U^\top, U = [u_1, \cdots, u_N] \in \mathbb{R}^{N \times N}$，特征值越大代表信息量越大。特征值均大于等于 0 且对于连通图，有且只有一个 0。

> 后文中适用于 L 的公式一般也适用于 L^{norm}。

Graph Laplacian 与 Laplacian 关系的物理学解释

https://www.zhihu.com/question/54504471

Graph Fourier Transform

对于经典 Fourier Transform，$\mathcal{F}[f(w)] = \langle f, \exp(-iwt) \rangle = \int f(t) \exp(-iwt) dt$ 可为视作 expansion of a functionfin terms of the complex exponentials 或者说 f 和这个指数函数的内积。

而 $\exp(-iwt)$ 是 Laplacian 的特征函数，也就是说 $\nabla^2 \exp(-iwt) = -w^2 \exp(-iwt)$，$-w^2$ 就是对应的特征值。

对应于 Graph Laplacian，类似的，有 $\mathcal{F}[f(\lambda_l)] = \langle f, u_l \rangle = \sum_{i=1}^N f(i) u_l^*(i)$，其中 $u_l^*(i)$ 为第 l 个特征向量的共轭的第 i 个元素，$\sum$ 可理解为离散积分。最后得到 $\mathcal{F}[f] = U^\top f$。

同理对于逆变换，我们有 $f(i) = \mathcal{F}^{-1}[\tilde{f}(w)] = \sum_{l=1} \tilde{f}(\lambda_l) U_l(i)$。

> TODO: Fourier transform 和 Laplacian 以及 Graph Laplacian 的关系的推导

函数卷积的 Fourier Transform 就是函数 Fourier Transform 的乘积：$f * h = \mathcal{F}^{-1}[\tilde{f}(w)\tilde{h}(w)]$，推广到 Graph 中可以表达为：$(f * h)_G = U((U^\top f) \odot (U^\top h)) = Uf(\Lambda)U^\top h$，其中 $\odot$ 是 Hadamard product（逐位乘），$f(\Lambda) = \mathtt{diag}(f(\lambda_1), \cdots, f(\lambda_n))$ 为可训练 filter。

> 上式 Fourier transform 复杂度 $\mathcal{O}(n^2)$ 太高

对于 filter f 的选择：

- 最简单的为 $f_\theta(\Lambda) = \mathtt{diag}(\theta), \theta \in \mathbb{R}^N$ [3] 是一个可学习向量参数（与 Λ 无关），用于作为 Fourier coefficients, 所以学习复杂度为 $\mathcal{N}(n)$ 而且这是个 global filter 不能学习 spatial localization（a.k.a. 局部空间特征）。
- Polynomial parametrization for localized filters：$f_\theta(\Lambda) = \sum_{k=1}^{K} \theta_k \Lambda^k, f_\theta * h = \sum_k \theta_k U \Lambda^k U^\top = \sum_k \theta_k L^k$, parameter $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients. 这样参数的学习复杂度就跟经典 CNN 一样是 $\mathcal{O}(K)$。并且根据 $d_\mathcal{G}(i,j) > K \Leftrightarrow (L^K)_{i,j} = 0$ （$d_\mathcal{G}(i,j)$ 表示 shortest path distance between vertex i and j，K^{th}-order polynomials of Laplacian 就是 K-localized，也就是 receptive field 的大小。
- Chebyshev polynomial as approximate of polynomial parameterization [1]：$f_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), f_\theta * h = \sum_k \theta_k T_k(\tilde{L})h, \tilde{\Lambda} = 2\Lambda/\lambda_{max} - I, \tilde{L} = 2L/\lambda_{max} - I$，$\lambda_{max}$ 可以由 power iteration 求出，其计算复杂度为 $\mathcal{O}(K|\mathcal{E}|) \ll \mathcal{O}(n^2)$ 远小于前面两个（因为这里采用了 K 个 **sparse** matrix-vector multiplications）。Chebyshev polynomial 定义为 $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x), T_0 = 1, T_1 = x, x \in [-1,1]$。

  > $2L/\lambda_{max} - I$ 的目的是为了满足 Chebyshev 对 $x \in [-1,2]$ 的要求

  > Chebyshev polynomial 的误差分析参考 [2]

> Ref:
>
> 1. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering
> 2. Wavelets on graphs via spectral graph theory
> 3. Spectral Networks and Deep Locally Connected Networks on Graphs

Undirected Weighted Hypergraph

HGNN (AAAI-19)

Ref: `Hypergraph Neural Networks`（abbr. HGNN, 19-AAAI）

Hypergraph 就是指同一条边连接的结点数大于 2，也就是边的度大于 2。

$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$，其中对角矩阵 $W \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{E}|}$ 表示 $W_{i,i}$ 为边 i 的权重，并且 $\mathcal{G}$ 表示为 incidence matrix $H \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$：

$H(v,e) = \mathbb{I}(v \in e), v \in \mathcal{V}, e \in \mathcal{E}$

- 顶点的度 D_v：$d(v) = \sum_{e \in \mathcal{E}} w(e)h(v,e)$
- 边的度 D_e：$\delta(e) = \sum_{v \in \mathcal{V}} h(v,e)$

hypergraph Laplacian: $\Delta = I - D_v^{-1/2} HWD_e^{-1} H^\top D_v^{-1/2}$

> hypergraph Laplacian 由 Learning with Hypergraphs: Clustering,Classification, and Embedding (NIPS-07) 中 formula 2 的优化问题推导出来的，该组合优化问题很像 graph Laplacian 的定义。对于普通无向图，$D_e = 2I, HWH^\top - D_v = A$ 带入上式得 $\Delta = I - 0.5D_v^{-1/2} HWH^\top D_v^{-1/2} = I - 0.5D_v^{-1/2}(D_v + A)D_v^{-1/2} = 0.5(I - D_v^{-1/2} A D_v^{-1/2})$，$A$ 就是邻接矩阵。

> TODO: formula 2 到 real-valued optimization problem 的推导过程

> $HWD_e^{-1}H^\top$ 主要是为了构造出类似于 Graph Laplacian 中的 W 的那种半正定性质

类似 Chebyshev polynomial as approximate of polynomial parameterization，可以得到：

$g * x \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Delta})x$

> $x \in \mathbb{R}^{n\times 1}$ 表示一个 n 个顶点，每个顶点为一维特征的简单样本

限定 $K = 1$（也就是卷积的 receptive field 为 2-hop neighbor），并且通过 [1] 中估计 $\lambda_{max} \approx 2$ 也就是 $\tilde{\Delta} \approx \Delta - I$:

$g * x \approx \theta_0 x + \theta_1 \tilde{\Delta} x = \theta_0 x - \theta_1 D_v^{-1/2} HWD_e^{-1}H^\top D_v^{-1/2} x$

> Ref:
>
> [1] Semi-Supervised Classification with Graph Convolutional Networks (abbr. GCN, ICLR-17)

并且为了避免 overfitting 以及减少计算量，将两个参数用一个参数表示
$\theta_1 = -0.5\theta, \theta_0 = 0.5\theta D_v^{-1/2} H{\color{red}I}D_e^{-1}H^\top D_v^{-1/2}$ (为什么要这样呢？为了凑后面的结果呀)，于是有：

$g * x \approx 0.5\theta D_v^{-1/2} H(I + W)D_e^{-1}H^\top D_v^{-1/2} x \approx \theta D_v^{-1/2} HWD_e^{-1}H^\top D_v^{-1/2} x$

> 后面这个转换是因为 W 初始为 I 表示所有 hyperedge 都是一样的权重。

对于输入图信号 $X \in \mathbb{R}^{n\times C_1}$，可训练参数 $\Theta \in \mathbb{R}^{C_1\times C_2}, W \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$（对角矩阵），hypergraph 卷积输出为：

$Y = \sigma(D_v^{-1/2} HWD_e^{-1}H^\top D_v^{-1/2} X\Theta)$

node-edge-node transform 用于学习 higher order：

例如顶点特征图卷积之后为 $n \times C_2$，接着使用左乘 $H^\top$ 按照边对顶点进行特征融合变为 $|\mathcal{E}| \times C_2$，接着再左乘 H 按照顶点对边进行特征融合为 $n \times C_2$。

> 这个 node-edge-node 在代码里面没有找到。。

> 该文差不多就是把 Learning with Hypergraphs: Clustering,Classification, and Embedding 和 GCN 结合了一下，并且它的两个应用：Citation network（把原来 graph network 改为 hypergraph，不过两者网络结构差别不大所以提升也不大）和点云分类（每个 node 为一个 object， 特征为 Multi-view CNN 和 Group-view CNN 提取出来的特征向量，强行把 K 近邻的 objects 连为一条 hyperedge，不过效果比较好）都不是很有代表性的 hypergrpah

HyperGCN (NIPS-2019)

Ref: `HyperGCN: A New Method of Training GraphConvolutional Networks on Hypergraphs` (abbr. HyperGCN, NIPS-2019)

> Code: https://github.com/malllabiisc/HyperGCN/blob/master/model/utils.py

HyperGCN 认为 HGNN 一类方法采用的是 clique expansion of a hypergraph (converting each hyperedge to a clique subgraph), 认为会导致 distortion, fails to utilise higher-order relationships in the data and leadsto unreliable learning performance for clustering。这主要是为了解释本文提出的将 hypergraph 转化为 graph 来处理的思想。

Defs:

- $\mathcal{H} = (V, E), n = |V|$
- Semi-supervised learning (SSL) : learning a small set V_L of labeled hypernodes to predict other unlabeled hypernodes in V/V_L.

- Basic assumption: hypernodes in the same hyperedge aresimilar and hence are likely to share the same labe. Hypergraph Laplacianas is an implicit regulariser which achieves this objective.

Hypergraph laplacian with mediators:

For graph signal $S \in R^{n\times c}$, construct weighted graph $G_S \in \mathbb{R}^{n\times n}$: For each hyperedge $e \in E$, connect node pair$(i_e, j_e) := \arg\max_{i,j\in e} |S_i^\top r - S_j^\top r|$ with $1/(2|e|-3)$ and $r \in \mathbb{R}^{c\times 1}$ is a **random** vector (this trick is called **breaking ties randomly**). Also connect mediators $K_e := \{k \in e | k \neq i_e, k \neq j_e\}$to above two nodes with $1/(2|e|-3)$ (normalize weights on the edges for each hyperedge to 1). Then the normalised hypergraph Laplacian is: $\mathbb{L} = I - D_S^{-1/2} A_S D_S^{-1/2}$ where D_S, A_S are diagonal degree matrix and adjacency matrix of G_S.

Two-layer GCN:

$$Z = f(X, A) = \mathrm{softmax}(\hat{A}\ \mathrm{ReLU}(\hat{A}X\Theta^{(1)})\Theta^{(2)}) \in \mathbb{R}^{n\times c}$$

where $\hat{A} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}, \tilde{A} = A + I, \tilde{D} = \sum_{j=1}^{n} \tilde{A}_{jj}$.

> The above transform is called *renormalization trick* in GCN: 1st-order Chebyshev polynomial approximation is $g * x = \theta(I + D^{-1/2}AD^{-1/2})x$ where $I + D^{-1/2}AD^{-1/2}$ has eigenvalues in range $[0, 2]$ which will leads exploding/vanishing gradients. Therefore GCN use the trick to fix this problem.

HyperGCN:

$S_i = (\Theta^{(l)})^\top h_i^{(\tau,l-1)}$ where $\Theta^{(l)}$ is the parameters of the layer l, and $h_i^{(\tau,l-1)}$ is the representation of node i from last layer in epoch τ.

In each epoch and each layer, HyperGCN will $\hat{A}_S^{(l)}$ according to the input graph signal (X or $H^{(l-1)}$):

> neural message-passing framework 和 $h_v^{(\tau+1)} = \sigma(((\Theta^{(l)})^{(\tau)})^\top \sum_{u\in\mathcal{N}(v)}([\hat{A}_S^{(\tau)}]_{vu} h_u^{(\tau)}))$ 这玩意是强行加上去的吧。。代码里面根本没有, 只有重新计算 $\hat{A}_S^{(l)}$ 然后 $\hat{H}_S^{(l)} H^{(l-1)}\Theta^{(l)}$。

> $H^{(0)} = X$

where $\mathcal{N}(v)$ is the set of neighbours of v.

$$Z = \mathrm{softmax}(\mathrm{ReLU}(\hat{A}_S^{(2)}\ \mathrm{ReLU}(\hat{A}_S^{(1)} X\Theta^{(1)} + b^{(1)})\Theta^{(2)} + b^{(2)}))$$

FastHyperGCN:

Only compute A once before training using X:

$$S_i = X_i, Z = \mathrm{softmax}(\mathrm{ReLU}(\hat{A}_S\ \mathrm{ReLU}(\hat{A}_S X\Theta^{(1)} + b^{(1)})\Theta^{(2)} + b^{(2)}))$$

Experiments:

- Co-authorship: All documents co-authored by an author are in one hyperedge
- Co-citation: All documents cited by a document are connected by a hyperedge. Remove hyperedges with length 1.

Feature of each hypernode (document) is bag-of-words.

Bad-of-words: For a dictionary Σ of the most frequent words in all documents, the 0/1-valued feature vector $x^{(i)} \in \mathbb{R}^{|\Sigma|}$ for document i indicating the absence/presence of the corresponding word from the dictionary.

Objective:

Classifying each document (hypernode) to its corresponding categories.

本文比较好的地方是用理论分析了实验结果。

> 本文主要思想就是 breaking ties randomly (15年 STOC 他们自己的论文) 和 mediators 将 hypergraph 转化为 graph 然后用 GCN

Datasets

1. COLLAB (Scientific Collaboration) [1]

   3-class, If author i co-authoreda paper with author j, the graph contains an undirected edge from i to j.

   Only have node id, no any other features in nodes and edges.

   http://snap.stanford.edu/data/ca-HepPh.html

   http://snap.stanford.edu/data/ca-CondMat.html

   http://snap.stanford.edu/data/ca-AstroPh.html

2. Twitter (not suitable to our target)

   Containing around 950 ego networks (directed) of users from Twitter with a mean of around 130 nodes and 1700 edges per graph. 每个节点代表一个用户， 一个图代表一个以某个用户为中心的关于他的朋友圈，边代表用户关注，节点特征为 user profile（组织成 tree）。原文任务是从图中找出不同的小圈子（类似于 Twitter 的列表功能，也就是一个子图），并且这些小圈子可能会重叠或者内含。

   http://snap.stanford.edu/data/egonets-Twitter.html

3. arXiv

   - co-ciations & co-author
   - 1.35×10^6 篇论文，6.72×10^6 条引用关系
   - Features: co-ciations & co-author relations，论文标题，单位，论文 ID
   - Task: 多分类（多标签）

> Ref:
>
> [1] A New Space for Comparing Graphs
>
> [2] On the use of arxiv as a dataset

Pooling

$m_{ik}^l = \sum_{j \neq i}^{v_j \in e_k} h_j^{l-1}, \forall e_k \in \mathrm{Adj}(v_i)$

把v_i所在边的顶点特征全部聚集起来就是 Pooling。

$z_{ik} = \langle h_i^{l-1}, m_{ik}^l \rangle$

$n_i^l = \sum_k \mathrm{softmax}(z_{ik})_k m_{ik}^l$

h_j^{l-1} 表示顶点 v_j 在第 $l-1$ 层的特征, n_i^l 就是 aggregate 后的顶点特征

> Ref:
>
> [1] Learning Multi-Granular Hypergraphs for Video-Based Person Re-Identification

$Z = \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} X \Theta)$

$\mathrm{idx} = \text{top-rank}(Z, \lceil kN \rceil), k \in (0, 1]$

$X_{\text{out}} = X_{\text{idx}} \odot Z_{\text{mask}}$

把 N 个顶点 Pooling 为 $\lceil kN \rceil$ 个顶点

> Ref:
>
> [1] Self-Attention Graph Pooling