南京大学
NANJING UNIVERSITY

周　报

汇报人：王云攀

2019.7.27-2019.8.3

目录

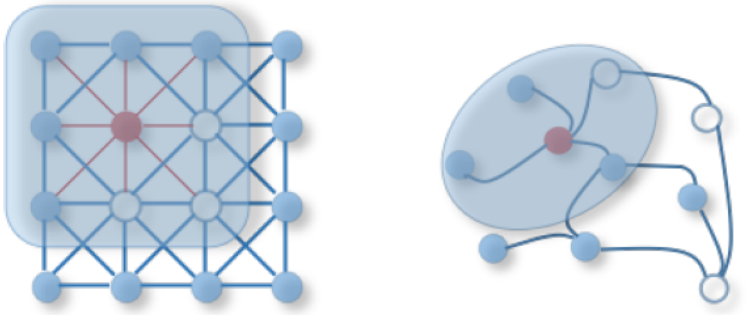
- 本周完成工作
  - 论文阅读1：A Comprehensive Survey on Graph Neural Networks
  - 论文阅读2：Semi-Supervised Classification With Graph Convolutional Networks
- 接下来的工作计划
  - 尝试实现第二篇论文，并进行测试验证

论文阅读1

- Abstruct
  - 序言
    - 深度学习在应用广泛，数据却都表示在欧式空间
    - 对于非欧空间的需求，希望能建模为图（节点之间有着复杂的关系和依赖）
    - 问题：图的复杂性太高！！！
  - 文章
    - 提出了框架和分类New taxonomy
    - 对每个类别进行了介绍Comprehensive review
    - 在现实的应用和数据集Abundant resources
    - 未来的方向Future directions

论文阅读1

- Introduction
  - 如何降低图的复杂性？
    - 受CNN的启发，试图构建图的卷积graph convolution
    - 难点：图的每个节点是无序的而且大小不确定
    - 重点：
      - 结构属性
      - 节点的邻接点



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes a weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.

(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of graph convolution operation takes the average value of node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

Fig. 1: 2D Convolution vs. Graph Convolution.

论文阅读1

- Introduction
  - 发展历史
    - 基于卷积的简单表示->spectral based->spatial-based->…
  - 一些技术
    - Laplacian matrix， Attention，Random Walk
  - 辨析
    - graph neural network VS graph embedding

      Graph embedding目标是得到一个低维的表示，保留拓扑和本省内容信息；但是，实际表示具有什么含义无从得知

      Graph neural network是多种神经网络的总称，有多种目标

论文阅读1

- Definition

TABLE 1: Commonly used notations.

Notations	Descriptions
$\|\cdot\|$	The length of a set
$\odot$	Element-wise product.
$\mathbf{A}^T$	Transpose of vector/matrix $\mathbf{A}$.
$[\mathbf{A}, \mathbf{B}]$	Concatenation of $\mathbf{A}$ and $\mathbf{B}$.
$\mathcal{G}$	A graph
$\mathbf{V}$	The set of nodes in a graph
v_i	A node $v_i \in \mathbf{V}$
$N(v)$	The neighbors of node v
$\mathbf{E}$	The set of edges in a graph
e_{ij}	An edge $e_{ij} \in \mathbf{E}$
$\mathbf{X} \in \mathbf{R}^{N \times D}$	The feature matrix of a graph.
$\mathbf{x} \in \mathbf{R}^N$	The feature vector of a graph in the case of $D = 1$.
$\mathbf{X}_i \in \mathbf{R}^D$	The feature vector of the node v_i.
N	The number of nodes, $N = \|\mathbf{V}\|$.
M	The number of edges, $M = \|\mathbf{E}\|$.
D	The dimension of a node vector.
T	The total number of time steps in time series.

论文阅读1

- Categorization and frameworks
  - Categorization
    - Graph Convolution Networks(GCN)
      考虑自己和邻居的特征信息，学习到该节点的函数
    - Graph Attention Networks(GAN)
      引入了Attention的概念，重点关注更重要的信息
    - Graph Auto-encoder(GAE)
      基于半监督学习，试图学习低维表示，能够转换回去
    - Graph Generative Networks: 从数据中生成图的结构
    - Graph Spatial-Temporal Networks(GSE)
      从随空间结构随时间变化的图中学习

论文阅读1

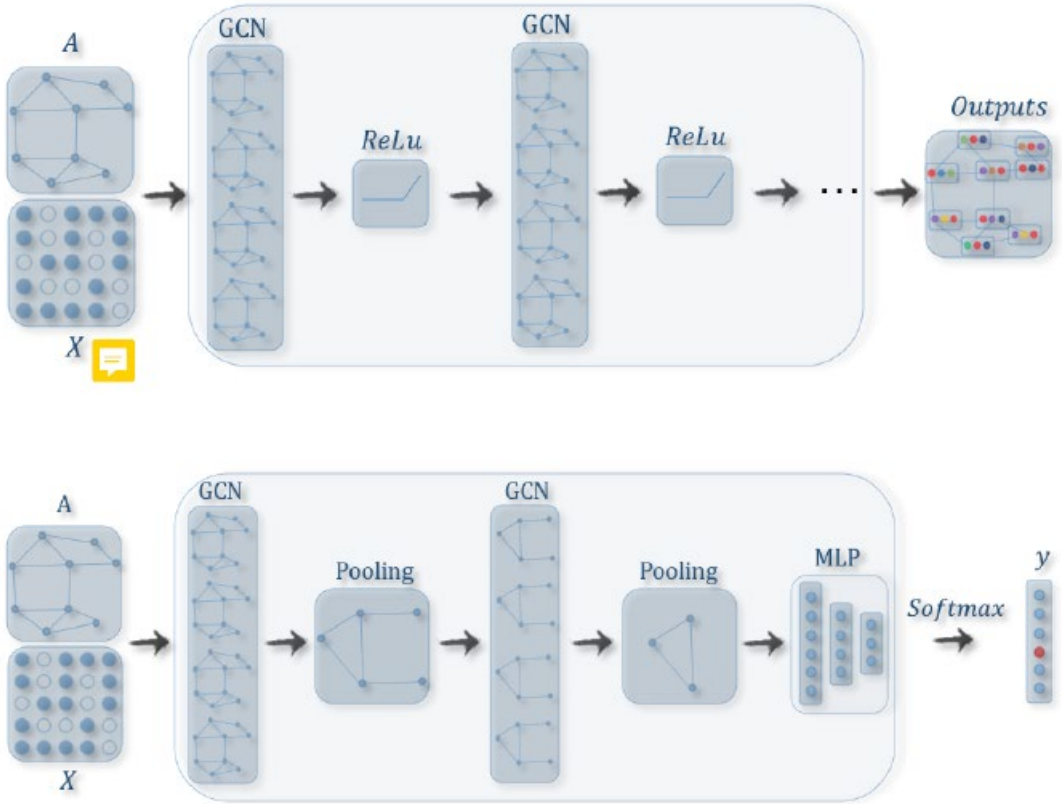
- Categorization and frameworks
  - Frameworks
    - 角度1
      - 输入: 图的结构和节点的内容信息
      - 输出：对应于不同的图的分析任务
        - Node-level：关于节点的回归和分类任务，最后一层为softmax函数或者感知机
        - Edge-level: 关于边的分类或者边的预测任务，一个额外的函数用作嫁女两个节点的表示
        - Graph-level: 图的分类任务，最后一层一般为pooling层

论文阅读1

- Categorization and frameworks
  - Frameworks
    - End-to-end 框架
      - 半监督学习 node-level classification
        先学习有标记数据，再对未标记数据进行分类
      - 监督学习 graph-level classification
        给出整个图数据集，进行图的分类任务
      - 无监督学习 graph embedding
        没有具体的分类标准
        - auto-encoder + decoder

论文阅读1

- GCN
  - Spectral-based
  - Spatial-based
  - Pooling modules



论文阅读1

- Spectual-based
  - 直观理解

    从图信号的角度，将卷积的操作看作是消除噪声
  - 操作
    - 构建Laplacian矩阵
    - 矩阵特征值分解
    - 关键设计filter

$$\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}},$$

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

$$\begin{aligned}\mathbf{x} *_G \mathbf{g} &= \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g})) \\ &= \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g})\end{aligned} \tag{1}$$

where $\odot$ denotes the Hadamard product. If we denote a filter as $\mathbf{g}_\theta = diag(\mathbf{U}^T\mathbf{g})$, then the graph convolution is simplified as

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x} \tag{2}$$

论文阅读1

- Spectual-based
  - 主要方法
    - Spectral CNN: 直接对参数进行学习
    - ChebySheve: 改进了公式，从数学的角度推导，使得目标计算不再需要特征值分解
    - 1阶Cheby: 沟通了Spaitial-method, 但是在batch training中，随着层数的增加，计算开销会指数级扩张
    - Adaptive Graph Convolution Networks

      Laplacian矩阵只可以用于无向图，试图用一个剩余矩阵来代替邻接举证，衡量两个节点的距离

$$g_\theta = \sum_{i=0}^{K} \theta_i T_k(\widehat{\Lambda})$$

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U}(\sum_{i=0}^{K-1} \theta_i T_k(\tilde{\mathbf{\Lambda}}))\mathbf{U}^T\mathbf{x}$$

$$= \sum_{i=0}^{K-1} \theta_i T_i(\tilde{\mathbf{L}})\mathbf{x}$$

$$\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{max} - \mathbf{I_N}.$$

$$\mathbf{x} *_G \mathbf{g}_\theta = \theta_0\mathbf{x} - \theta_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{x}$$

论文阅读1

- Spectual-based
  - 评价
    - Spectral CNN: 直接对参数进行学习
    - ChebySheve和1阶Cheby
      - 使得公式的更新复杂度从O(N^3)降低到O(E)
      - 只需用到空间中的局部信息
    - AGCN扩展了适用范围，使得有向图也可做
  - 总体评价
    - 依赖于特征分解, 需要N^3的时间，N^2的空间
    - 受数据扰动影响大
    - Filter是domin dependent
    - 需要将图的所有信息加载到内存中，扩展性差

论文阅读1

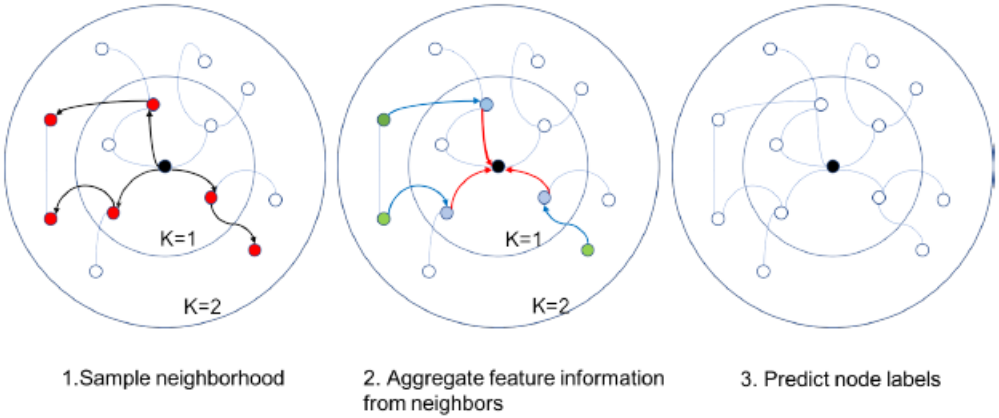
- GCN
  - Spatial-based

    想法: 综合考虑到节点自身和邻居信息，以及空间信息
    - Recurrent-based:

      基于循环神经网络，多个相同的GCN堆叠
      - Graph Neural Networks:关于节点表示，节点边的表示，上一步邻居的表示，邻居的属性的函数，要求该函数是收缩映射的，凸函数，能够收敛
      - Gate Graph Neural Networks： 对于GNN的循环的步长的改进，加入GRU门控单元，可以根据隐藏状态和当前出现的值来决定是否进一步计算还是改变当前值进行另一种计算。
      - Stochasitc steady-state embedding: 每个GCN，随机选择一些点进行更新，随机选取几个已更新的点来更新梯度（这里考虑历史状态和新状态的平均权重）

论文阅读1

- GCN
  - Spatial-based
    - Composition-base



每一步的GCN均不同

  - Message Passing Neural Networks

    Messgae passing phase: 每一步的信息函数综合上一步信息和所有邻居节点更新后的操作的集合

    Readout phase： 最后一步，相当于池化，用来产生整个图的一个表示

  - GraphSage

    1. 对一个节点的k-hop邻居进行固定大小的采样
    2. 通过聚集邻居节点的所有信息，最终产生该节点的最终状态
    3. 用最终状态来预测和反馈梯度

    注意到这里不是更新所有接地那，而是进行batch-learning，感觉可以扩展到很大的数据集

论文阅读1

- GCN
  - Spatial-based
    - 杂项
      - DCNN扩散卷积神经网络 $Z^m_{i,j,:} = f(W_{j,:} \odot P^m_{i,j,:} X^m_{i,:})$

        模拟压缩图的扩散过程，P概率转移矩阵？（具体？应该是分布），问题在于需要在内存中存储转移矩阵 $O(N_m^2 H)$
      - PATCHY-SAN

        预处理对节点进行排序，将图结构数据转化为网格数据，取特定的前k个，然后用标准的CNN进行学习。CNN：移动不变性。
      - LGCN

        node-level, 与前一个算法不同的是，得到特征矩阵，选取特征矩阵的前k行来做。！！！子图训练
      - Mixture Model Network

        考虑了空间位置关系，即边的权重，建议使用标准的高斯核来学习参数调节边的权重

论文阅读1

- GCN
  - Spatial-based评价
    - Recurrent能获得节点的稳定状态
    - Composition可能获得高级别有用的信息
    - 两种结构中，每一层在训练中都需要查看和更新所有节点的隐藏状态；如何破？
      - 基于子图训练的方法？？？
        - GraphSage？
        - 随机异步训练的方法SSE？
    - 近几年提出的方法，建立更复杂的网络结构
      - 门控机制控制节点的深度和宽度 GeniePath
      - 设计两个图来维持本地的一致性和全局一致性 DuraGCN
      - 超参数来影响节点接收域的大小
  - Spectral vs Spatial
    - 效率上，Spatial可以用batch和sampling提高效率
    - 泛化性，Spatial可以共享权重
    - 灵活性，Spatial可以处理多源输入

完胜

论文阅读1

- GCN
  - Pooling
    - 类似于CNNS中的down-sampling?
    - 池化可以加快计算
    - ChebNet在输入图形的一开始进行了coresen处理，使其传递到了平衡二叉树的最低级别，产生了一个很好的效果
    - DGCNN SortPooling首先使用SortPooling进行了预处理，先得到顺序，然后再染成WL colors, 再对其去前k个云阿苏
    - DIFFPOOL使用两个GCN, 一个embed, 一个pool,可以整合异质的GCN; 其中GCN-embed得到新的节点表示，GCN-pool得到新的指派矩阵

$$Z^{(l)} = GNN_{l,embed}(A^{(l)}, X^{(l)})$$

$$S^{(l)} = softmax(GNN_{l,pool}(A^{(l)}, X^{(l)}))$$

$$X^{(l+1)} = S^{(l)^T} Z^{(l)} \in R^{n_{l+1} \times d}$$

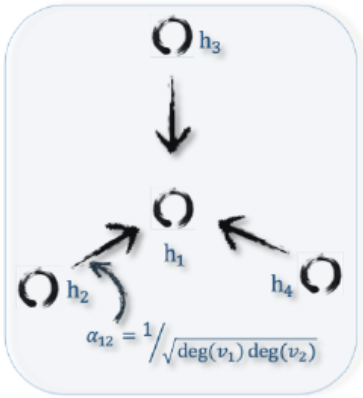
$$A^{l+1} = S^{(l)^T} A^{(l)} S^{(l)} \in R^{n_{l+1} \times n_{l+1}}$$

论文阅读1

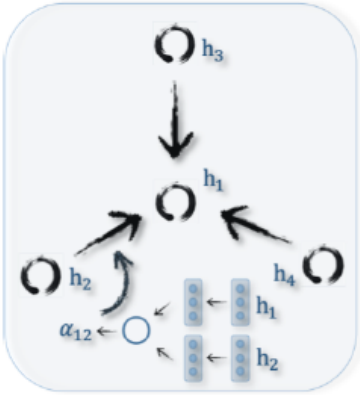
- GAN（Attention）

  试图将注意力集中在重要的东西上，相比于spaital方法在计算权重上加了个GAN函数

  - GAT: 考虑所有邻居节点的表示和邻居节点的权重。并提出了子空间的概念即指邻居节点可能存在着不同的表示？
  - GAANL:也考虑multi-head,不过是分配不同的权重
  - GAM: 使用LSTM网络，考虑某一段的状态
  - AW: node embedding, 学习转移矩阵，并考虑不同的概率。



(a) Graph Convolution Networks [14] explicitly assign a non-parametric weight $a_{ij} = \frac{1}{\sqrt{deg(v_i)deg(v_j)}}$ to the neighbor v_j of v_i during the aggregation process.

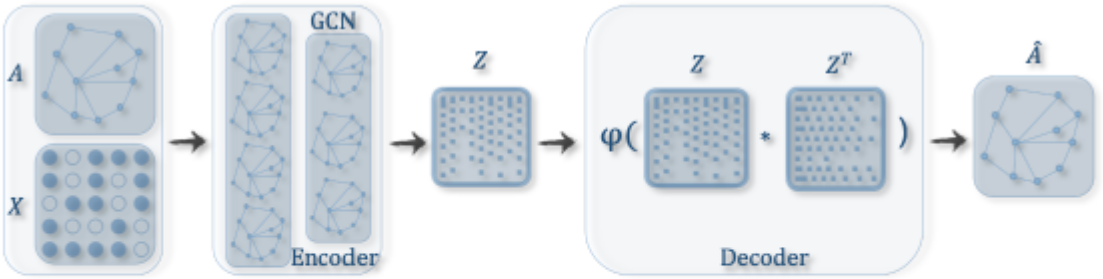


(b) Graph Attention Networks [15] implicitly capture the weight a_{ij} via an end-to-end neural network architecture, so that more important nodes receive larger weights.

论文阅读1

- GAN(Attention)
  - GAN相比GCN多了一层网络结构
  - 注意力的应用
    - 为邻居赋予权重
    - 整合多个模型
    - 指导Random Walks

论文阅读1



(b) Graph Auto-encoder with GCN [62]. The encoder uses GCN layers to get latent rerpesentations for each node. The decoder computes the pair-wise distance between node latent representations produced by the encoder. After applying a non-linear activation function, the decoder reconstructs the graph adjacency matrix.

- GAE
  - GCN based
    - 使用GCN，然后用距离来做损失函数进行评价
  - AGRA
    - GCN产生，GAN使用min-max策略训练。先验：节点的表示服从某一分布
  - 杂类
    - NetRA: 同AGRA，加了正则化节点的隐藏表示，使其服从先验分布，训练中非重建邻接矩阵，而是恢复节点的序列，通过sequence-to-sequence的random walks采样得到

论文阅读1

- GAE
  - 杂类
    - DNGR: 使用堆叠的自动编码器来做，加入了噪声，以及基于统计数据确定共现矩阵
    - SDNE: 使用了两个指标来学习，评价邻居和结果，正则化项
    - DRNE: 目标是重建节点，使用LSTM作为聚集函数

$$\mathbf{PPMI}_{v_1,v_2} = max(\log(\frac{count(v_1,v_2)\cdot|D|}{count(v_1)count(v_2)}),0) \quad (29)$$

其中$|D| = \sum_{v_1,v_2} count(v_1,v_2)$

$$L_{1st} = \sum_{i,j=1}^{n} A_{i,j}||h_i^{(k)} - h_j^{(k)}||^2$$

$$L_{2nd} = \sum_{i=1}^{n} ||(\hat{x}_i - x_i)\odot b_i||^2$$

$$L = L_{2nd} + \alpha L_{1st} + \lambda L_{reg}$$

$$L = \sum_{u\in V} ||h_v - aggregate(h_u \in N(v)||^2$$

论文阅读1

GAE 评价

- DNGR+SDNE: 仅考虑拓扑结构
- 为了解决邻接矩阵的稀疏性：
  - GAE: 重新赋予权重
  - NetRA: 线性化
  - DNGR: 密集举证
  - SDNE: 惩罚

GGN-MolAN

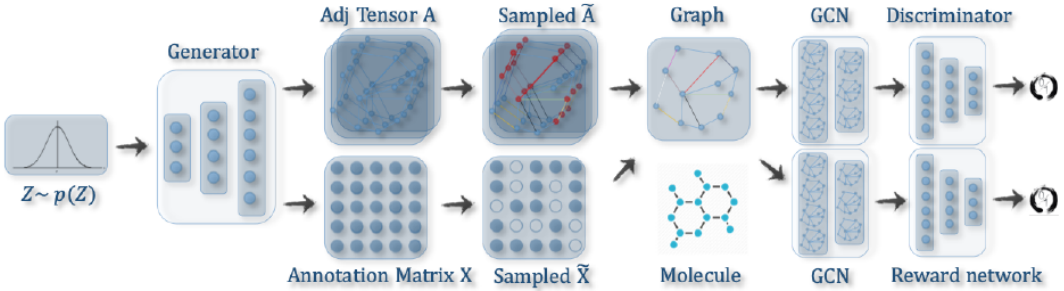


Fig. 9: Framework of MolGAN [70]. A generator first samples an initial vector from a standard normal distribution. Passing this initial vector through a neural network, the generator outputs a dense adjacency matrix A and a corresponding feature matrix X. Next, the generator produces a sampled discrete $\tilde{A}$ and $\tilde{X}$ from categorical distributions based on A and X. Finally, GCN is used to derive a vector representation of the sampled graph. Feeding this graph representation to two distinct neural networks, a discriminator and a reward network output a score between zero and one separately, which will be used as feedback to update the model parameters.

论文阅读1——GGN

- MolGAN

  分布->Generator->密集邻接矩阵tensor(张量)->sample产生Graph-> GCN-> Disciminator产生反馈和进行调节

> 需要预知分布，共同产生边和节点

- DGMG

  采用加点和边的方式，有一个停止的标准；当decision为假时，加边；为真时，评估将加入的节点连接到已存在的节点的可能分布，然后随机选择一个点。当一个新的节点和它的连接都被加入图中，更新图的表示。

> 需要预知分布，采用序列化的方式来做
> 图的表示用来计算前面的decision, 所以该方法又称为根据图的表示产生图的节点和边

- 杂项
  - GraphRNN

    两个RNN，对于graph和level，操作在二进制序列上。如何添加根据假设分布来做。对于graph为了训练，用BFS线性化为序列

> 采用序列化的方式来做，利用了统计数据？

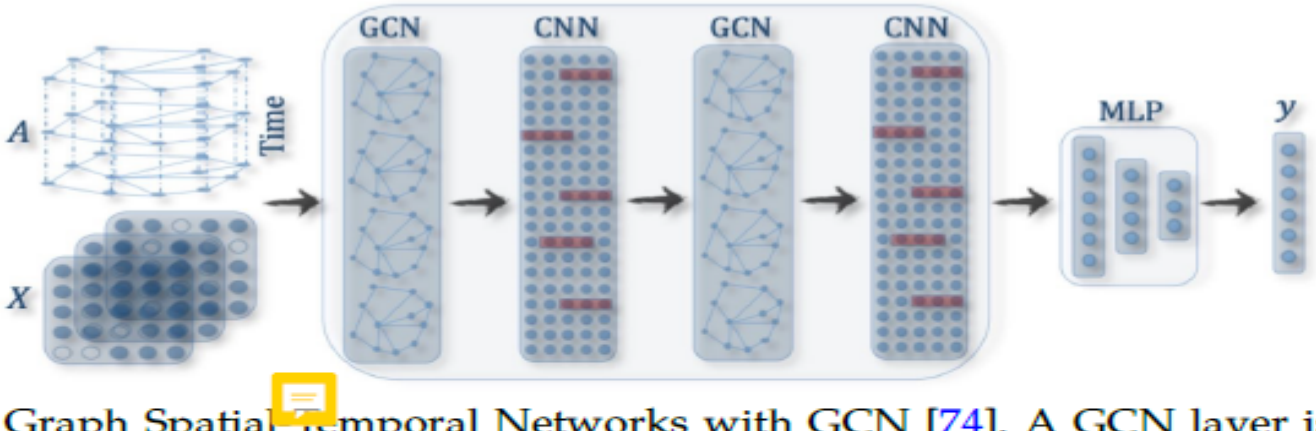
  - NetGAN

    产生器LSTM产生合理的random walks, 测试器GAN挑选，最后获得共现矩阵

> 利用统计数据？ 同时产生节点和边

论文阅读1

- GGN评价
  - 基本是都是基于产生+测试的模块
  - 序列化的方式，长序列难以维持；其他方式维护全局属性难；
  - 最近采用的变化的auto-encoder,增加了输出空间
  - 没有方法有很好的大规模可扩展性
- GSN



(c) Graph Spatial-Temporal Networks with GCN [74]. A GCN layer is followed by a 1D-CNN layer. The GCN layer operates on A_t and X_t to capture spatial dependency, while the 1D-CNN layer slides over X along the time axis to capture the temporal dependency. The output layer is a linear transformation, generating a prediction for each node.

Fig. 5: Different Graph Neural Network Models built with GCNs.

3.5 GSN

目标：可以预测未来的节点的value和值

- DCRNN
  扩散的卷积，出度入度来反映空间性。 DCGRU来统计上一步和邻居节点的卷积信息来反映瞬时性

> ??,可能是因为这里用的循环结构，可以循环多次，所以就可以考虑反映了瞬时性

- CNN-GCN
  GCN空间信息， 1DCNN来捕捉每个节点的瞬时信息，最终产生每个节点的一个预测
- ST-GCN
  考虑瞬时流作为节点的边（因此认为兼顾了空间结构，因为考虑了边），所以可以用一个GCN来统计两种信息。用一个标记为两个节点之间的距离。最终每个节点的值为邻接矩阵的求和
- 杂项
  - Strutual-RNN
    预测节点在每个时间步骤的标记
    用语义组来划分，每个语义组使用不同的RNN模型；这里是考虑了瞬时信息；对于空间信息的考虑，nodeRNN将edgeRNN的输出作为输入认为是考虑了空间性

DCRNN能处理长时间

CNN-GCN有效取决于1DCNN

ST-CNN考虑瞬时流作为节点的边，导致邻接矩阵平方增长，会导致计算图卷积层的开销增大，但是要堆叠很多次

Strutural-RNN需要先验知识来分割语义组

空间+瞬时

论文阅读1

- Future Directions
  - Go Deep：从理解上说，深度的网络的结构，所有的节点的表示可以聚集为一个节点；而且可以加入更多的信息，所以提高层数依然是个好办法
  - Receptive Field：观察一个节点时，考虑k邻居，假设邻居的数量服从一个合法的分布；
  - Scalability：大规模图的扩展性，现有办法：快采样和子图训练
  - 动力学和异质性：现有的图都是静态的同质的图，考虑图的结构变化，节点和边的来源不同的情况

分隔线

论文阅读2

- Introduciton

4.1 基于图的半监督学习

拉普拉斯矩阵： 标签传播、多种正则化、深度半监督表示

注意力机制： DeepWalk,

> skim-gram model, 简单来说就是，当要看当前单词时，向前或向后看几个单词
> 通常这些做法有种更多的步骤或者宽度优先模式，所以需要考虑用哪一步进行优化才是最恰当的

4.1 关于图的神经网络

2009的框架重复利用了收缩的想法进行传播误差直到最终节点的表示到达一个稳定的状态

2015年的做法介绍了一种类似卷积的传播规则作用在图上，并且针对于图级别的分类。但是可以发现，因为卷积要求有着固定的邻居。所以这种做法是不符合当节点的度有着很广的分布的情况。

> 这篇文章会重新考虑计算邻接矩阵，所以不会担心有这种因素的出现

2016,1-DNN, 引入了排序的想法
本文的方法，2014,简单，可扩展

论文阅读2

- 结构图

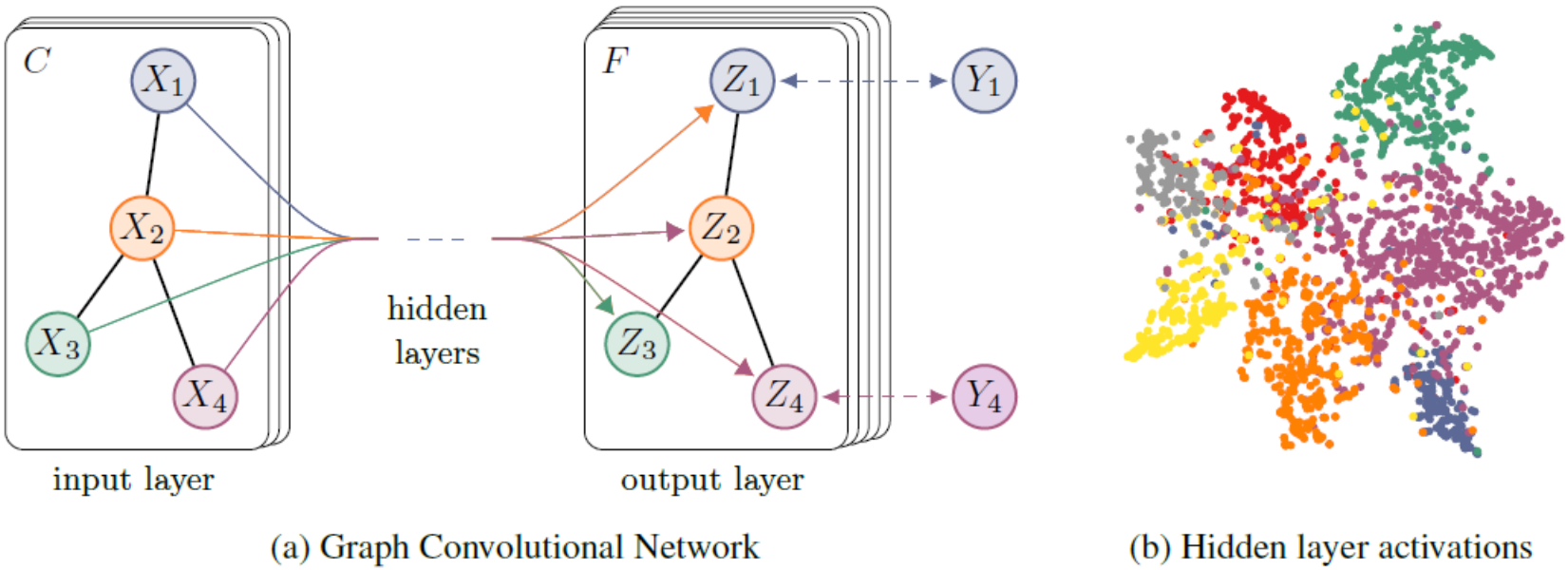


Figure 1: *Left*: Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Y_i. *Right*: t-SNE (Maaten & Hinton, 2008) visualization of hidden layer activations of a two-layer GCN trained on the Cora dataset (Sen et al., 2008) using 5% of labels. Colors denote document class.

论文阅读2

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$ 交叉熵

- 目标函数

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2 = f(X)^\top \Delta f(X). \qquad (1)$$

- 深度网络表示函数

$$Z = f(X, A) = softmax(\hat{A}ReLU(\hat{A}XW^{(0)})W^{(1)}$$

- 对于多层中间的传递公式为

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

  - $\tilde{A} = A + I_N$: 这里增加考虑了自己的因素;
  - $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
  - $W_{(l)}$是每层的权重
  - δ是激活函数，比如$ReLU(\cdot) = max(\cdot, 0)$
  - $H^{(0)} = X$, 表示每一层表示得到的值

对应的还有的就是 Residual(隐藏层之间的残差连接):

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

论文阅读2

5.1 数据集

类别

1. 引用网络
   Citeseer, Cora, Pubmed
2. 知识图
   NELL, 本身属于二部图

内容

数据： 数据类别， 节点， 边（这里会进行定义）， classes:节点的类别， features:输入的channels， Label rate: 用作训练的数据集

Table 1: Dataset statistics, as reported in Yang et al. (2016).

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

论文阅读2

预处理

- 引用网络
  无向图，不用处理，二进制邻接举证；每个类只使用了20个标记，使用了全部的特征

> 1. 是如何筛选出训练集和测试集的？
> 2. 是否保证了训练集中各个标记的样本的比例相同？

- NELL
  有向图，预处理同2016. 做法将(e1, r, e2)拆分为(e1,r1), (e2, r2) 也就是将节点拆分成了很多片，这样从某种程度就可以使得一条边可以被两个点同时共用 所以也就使得得到的节点表示是稀疏向量，one-hot作为特征，所以产生了61278维， 这里可以注意到，one-hot每一个特征所拥有的取值即其位数 ，如果两个节点中存在边的话，则说明边是存在的，则对应邻接矩阵中置为1
- Random graphs
  我们模拟各种尺寸的随机图数据集，并在每轮中测试训练时间。
  一般的做法时，随机地为N个节点赋予2N条边。对于每个节点的特征初始化为I_N, 表示每个节点并没有什么特征。并为每个节点置一个假标记$Y_i = 1$

论文阅读2

- baselines: 使用了目前基于图的半监督学习中的所有模型，忽略了TSVM, 增加了迭代分类算法ICA

- 性能指标
  - 分类准确率
  - 运行时间

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

论文阅读2

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
1st-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**
1st-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

- 性能指标
  - 分类准确率
  - 运行时间

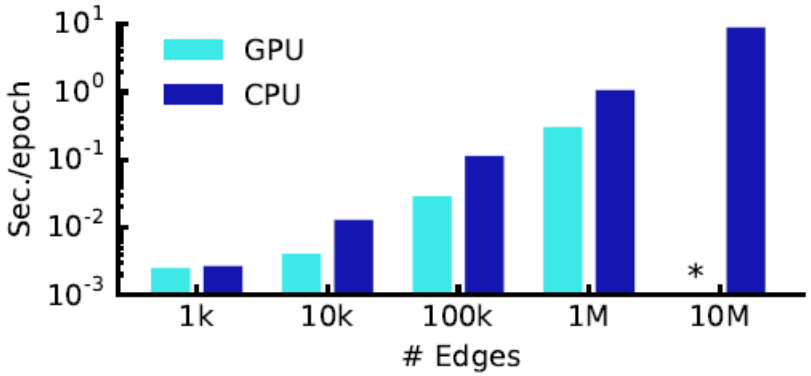


Figure 2: Wall-clock time per epoch for random graphs. (*) indicates out-of-memory error.

论文阅读2

5.2 实验部分

首先是3层的网络，附录B中有10层的网络。
data split技术 `Yang, 2016`

> ？ 这里是1000个样本做测试，500个样本做交叉验证的意思吗？

3层的网络，超参数：

- dropout
- L2
- number of units
- optimizer

> 优化器，就是采用什么样的梯度下降方式

> 这里不使用验证集作为训练，那还是用的交叉验证的方法吗？？

在Cora上优化得到的超参数直接用到Citeseer, Pumbed。 训练中优化器使用的是Adam， 学习率为0.01

> Adam优化器，自适应梯度下降优化器
> https://www.jianshu.com/p/aebcaf8af76e

停止条件：最大训练轮数200轮，如果交叉验证集误差连续10次不下降则停止训练。
权重初始化和特征向量标准化方法：Glorot & Bengio 2010
隐藏层单元：32个
忽略正则化项

> 隐藏层单元的设置除了对业务的理解，还有什么其他的建议点？
> 否则一般有经验公式

论文阅读2

- 不足之处
  - Memory requirement: GPU放不下所有的图
    - Min-batch 随机梯度方法
  - Directed edges and edge features: 只对无向图有用
  - Limiting assumptions: 相比于邻接矩阵，增加自己的重要性

$$\tilde{A} = A + \lambda I_N \ .$$