图网络GNN(特别篇)：一文遍览图网络中16种典型的图卷积和9种图池化 Graph Pooling

原创 Archwalker 深度学习与图网络 今天

> GNN的各种模型在近两年来非常火热，在各个会议、期刊上新的模型层出不穷，他们有的做了理论创新，有的对前人的工作提出了改进，在这篇博文中，我想要带大家回顾GNN在近两年来的一些模型的异同，着重体现在他们的数学表达式上的差异。本文中涉及到的公式可以左右滑动。
>
> 原文地址：https://archwalker.github.io/blog/2019/11/10/GNN-Go-Through-Main-Models.html（**阅读原文**）

这篇博文主要遵循 **DGL** 框架和**PyTorch geometric**的梳理脉络，加上一些对公式以及背后思想的解释。这篇博文面向的读者是对图神经网络已经有了一定程度的了解的学者。

文章中整理的GNN模型只是目前提出各种创新的一小部分，欢迎大家补充其他的模型。才疏学浅，如有疏漏，欢迎大家指正。

卷积层的设计

1.GraphConv来自论文Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017) 一作是Thomas N. Kipf，他当时在Amsterdam大学读博士。

这篇论文是可谓是图神经网络的开山之作，在我们前序的博文中也有解析，文中提出了一个简单且有效的图卷积算法：

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j\in\mathcal{N}(i)} \frac{1}{c_{ij}} h_j^{(l)} W^{(l)})$$

其中 $\mathcal{N}(i)$ 表示节点i的邻居节点，$c_{ij} = \sqrt{|\mathcal{N}(i)|}\sqrt{|\mathcal{N}(j)|}$ 是正则化项，这个公式的思想很简单，节点i更新后的Embedding为邻居节点Embedding的加权表示。在论文的实现中，$W^{(l)}$ 以Glorot uniform initialization 的方式初始化，$b^{(l)}$被初始化为0。

2.RelGraphConv来自论文 Modeling Relational Data with Graph Convolutional Networks (ESWC 2018 Best Student Research Paper) Thomas N. Kipf仍是并列一作。

这篇论文主要解决的问题是如果图数据有不同种类的边那么该如何进行卷积，作者提出的做法是对不同种类的边进行加权求和处理：

$$h_i^{(l+1)} = \sigma(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}^r(i)}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)})$$

式中 $\mathcal{N}^r(i)$ 表示在边类型为r时节点i的邻居节点，$c_{i,r} = |\mathcal{N}^r(i)|$是正则化项，

$$W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)} V_b^{(l)}$$

是对W_r的基向量分解，这样分解的原因是如果有太多种的边类型的时候，通过分解，只需要学习基向量$V_b^{(l)}$，减少欠拟合的风险。

3.TAGConv来自论文Topology Adaptive Graph Convolutional Networks 一作是来自CMU的Jian Du(杜建)，他当时在CMU做postdoc。

这篇论文是将GCN中对卷积的简化做了部分还原，细节在我们之前关于谱图卷积的理论博文中介绍过，具体而言，GCN模型是对卷积核进行Chebyshev多项式近似后取$k = 1$，这篇论文提出的方法将变量k保留下来作为超参：

$$\mathbf{X}' = \sum_{k=0}^{K}\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{X}\mathbf{\Theta}_k$$

与上面的记号稍有不同，这个公式用的是矩阵的更新形式，其中$\mathbf{D}ii = \sum_j A_{ij}$表示节点$i$的度。

4.GATConv来自论文 Graph Attention Network (ICLR 2018) 也是GNN各种模型中一个比较知名的模型，在我们之前的博文中介绍过，一作是剑桥大学的Petar Velickovic，这篇文章是在Yoshua Bengio的指导下完成的。

论文的核心思想是对邻居的重要性进行学习，利用学习到的重要性权重进行加权求和再对自身Embedding更新：

$$h_i^{(l+1)} = \sum_{j\in\mathcal{N}(i)}\alpha_{i,j}W^{(l)}h_j^{(l)}$$

其中 $\alpha_{i,j}$是邻居j对节点i的相对重要性权重，是通过下式学习得到的：

$$\alpha_{ij}^l = \mathrm{softmax_i}(e_{ij}^l)$$
$$e_{ij}^l = \mathrm{LeakyReLU}\left(\vec{a}^T[Wh_i|Wh_j]\right)$$

值得一提的是，同年的ICLR上，还有一篇关于Graph Attention的论文 Attention-based Graph Neural Network for Semi-supervised Learning 文章的主要思想是根据当前节点和邻居节点Embedding的cosine相似度作为Attention的加权因子，做了详细的实验和分析。

5.PointConv来自论文 PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation 这是较早得将GNN应用在点云上的文章：

$$\mathbf{x}'i = \gamma\mathbf{\Theta}\left(\max_{j\in\mathcal{N}(i)\cup i} h_{\mathbf{\Theta}}(\mathbf{x}_j,\mathbf{p}_j-\mathbf{p}_i)\right),$$

其中$\gamma_{\mathbf{\Theta}}$和$h_{\mathbf{\Theta}}$都表示多层感知机层，$\mathbf{P}\in\mathbb{R}^{N\times D}$表示每个点的坐标向量。

6.EdgeConv来自论文 Dynamic Graph CNN for Learning on Point Clouds (TOG 2019) 第一作者是MIT的博士Yue Wang, 这是另一一篇将图神经网络应用在点云上的文章，对于邻居节点Embedding的汇聚方法，他们是这么定义的：

$$x_i^{(l+1)} = \max_{j\in\mathcal{N}(i)} \mathrm{ReLU}(\Theta\cdot(x_j^{(l)}-x_i^{(l)})+\Phi\cdot x_i^{(l)})$$

这里使用两个节点Embedding的差在点云的数据上有其场景意义，因为在点云的数据集上，节点的Embedding一般取的是节点的坐标向量，所以两个节点Embedding的差表示的是两个坐标向量的差，即自当前节点i出发，到其邻居节点j的向量。

7.SAGEConv来自论文 Inductive Representation Learning on Large Graphs 第一作者是Stanford的博士William L. Hamilton 这篇文章是一篇非常经典的文章，里面提到的采样汇聚的方法也是目前将图神经网络应用到大规模数据集上的基础，我们在之前的博文中也对其进行了详细的介绍：

$$h_{\mathcal{N}(i)}^{(l+1)} = \mathrm{aggregate}\left(h_j^l, \forall j\in\mathcal{N}(i)\right)$$
$$h_i^{(l+1)} = \sigma\left(W\cdot\mathrm{concat}(h_i^l, h_{\mathcal{N}(i)}^{l+1}+b)\right)$$
$$h_i^{(l+1)} = \mathrm{norm}(h_i^l)$$

文章的思想如下，因为图数据每个节点的邻居个数是不一定的，带来了计算上的一些困难，所以文章通过采样的方法确保每个节点参与汇聚的邻居个数一定。在公式中j是采样得到的节点之一，aggregate函数对采样得到的节点集合进行汇聚，邻居集合的汇总Embedding和节点i本身的Embedding拼起来，再经过转换得到节点i更新后的Embedding。

8.SGConv来自论文 Simplifying Graph Convolutional Networks (ICML 2019) 第一作者是来自Cornell的Felix Wu ，文章通过实验发现不需要在每个卷积层进行线性变化和激活，这些操作可以合在一起做：

$$H^{l+1} = (\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2})^K H^l\Theta^l$$

这里$H^{(l+1)}$可以直接作为节点Embedding的输出结果。个人认为使用$H^{(l+1)}$表达是不准确的，这样写好像在每一层都需要这个卷积操作，反而加大了计算量，没有达到论文中“简化”的目的。

9.APPNPConv来自论文 Predict then Propagate: Graph Neural Networks meet Personalized PageRank ，文章试着将节点Embedding用一个类似于PageRank的框架更新：

$$H^0 = X$$
$$H^{t+1} = (1-\alpha)\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^t+\alpha H^0\right)$$

即每个节点的Embedding更新时会包含一部分历史的Embedding。

10.GINConv来自论文 How Powerful are Graph Neural Networks? (ICLR 2019 oral)，这是一篇比较有名的文章，作者是来自MIT的xukeyulu，这篇论文算是比较早的想要从理论上分析GNN模型的表达能力的文章，在我们的博文中有详细的介绍，文章想要研究GNN在图同构测试中能力，通过和Weisfeiler-Leman对比，得到一个具有和Weisfeiler-Leman想当能力的图神经网络模型：

$$h_i^{(l+1)} = f_\Theta\left((1+\epsilon)h_i^l + \text{aggregate}\left(\left\{h_j^l, j \in \mathcal{N}(i)\right\}\right)\right)$$

这篇论文主要做了两点创新，第一，公式中的aggregate采用add而非大部分GNN模型中的mean pooling，第二，给节点自身的Embedding加了少许扰动ϵ。后来的很多GNN模型都采用了该论文提出的方法作为子模块。

11.GatedGraphConv来自论文 Gated Graph Sequence Neural Networks，这篇论文是一篇早期的探索图神经网络中的长依赖的论文，一作是来自多伦多大学的Yujia Li，论文利用了时序建模中的GRU模块：

$$\begin{aligned} h_i^0 &= [x_i|\mathbf{0}] \\ a_i^t &= \sum_{j\in\mathcal{N}(i)} W_{e_{ij}} h_j^t \\ h_i^{t+1} &= \text{GRU}(a_i^t, h_i^t) \end{aligned}$$

可以看到，和之前介绍的GNN模型不同，邻居节点汇聚后Embedding a_i^t不再直接加到自身Embedding上(GCN)，也不再直接concat到自身Embedding上(GraphSAGE)，而是采用GRU的方式汇聚，以保持对长依赖的建模。

12.GMMConv来自论文 Geometric Deep Learning on Graphs and Manifolds using Mixture Model CNNs (CVPR 2017) ，论文提出了一个叫MoNet的框架，这个框架我不是特别熟悉，故暂且仅留下公式：

$$\begin{aligned} h_i^{l+1} &= \text{aggregate}\left(\left\{\frac{1}{K}\sum_k^K w_k(u_{ij}), \forall j \in \mathcal{N}(i)\right\}\right) \\ w_k(u) &= \exp\left(-\frac{1}{2}(u-\mu_k)^T\Sigma_k^{-1}(u-\mu_k)\right) \end{aligned}$$

13.ChebConv来自论文 Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering，这是对于谱图卷积的切比雪夫多项式近似，细节在我们之前关于谱图卷积的理论博文中介绍过，公式为：

$$
\begin{aligned}
h_i^{l+1} &= \sum_{k=0}^{K-1} W^{k,l} z_i^{k,l} \\
Z^{0,l} &= H^l \\
Z^{1,l} &= \hat{L} \cdot H^l \\
Z^{k,l} &= 2 \cdot \hat{L} \cdot Z^{k-1,l} - Z^{k-2,l} \\
\hat{L} &= 2 \left(I - \hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2} \right) / \lambda_{max} - I
\end{aligned}
$$

14.AGNNConv来自论文 Attention-based Graph Neural Network for Semi-Supervised Learning，上文中介绍过，这篇论文和Graph Attention Network 一起投稿在ICLR 2018上，这篇论文的主要思想是通过余弦相似度计算邻居节点和当前节点的加权权重，但是这篇论文最终被拒了，有可能是因为提出的方法比较简单，不过论文中做了详尽的实验分析，还是值得一看的：

$$
\begin{aligned}
H^{l+1} &= PH^l \\
P_{ij} &= \mathrm{softmax}_i(\beta \cdot \cos(h_i^l, h_j^l))
\end{aligned}
$$

15.NNConv来自论文 Neural Message Passing for Quantum Chemistry，主要用来解决边上有权重的图神经网络改如何更新Embedding的问题，提出的架构为：

$$
h_i^{l+1} = h_i^l + \mathrm{aggregate}\left(\left\{ f_\Theta(e_{ij}) \cdot h_j^l, j \in \mathcal{N}(i) \right\} \right)
$$

其中边上的权重e_{ij}被显示的建模到模型中来，作为邻居加权求和的权重，这里免去了对邻居权重的学习，直接用边的权重表征邻居的相对重要性。

16.DNAConv来自论文 Just Jump: Towards Dynamic Neighborhood Aggregation in Graph Neural Networks ，也着重于邻居相对重要性，与Graph Attention Network不同，这篇论文的方法更加直接得将“QKV”式的attention引入到公式中：

$$
\mathbf{x}_v^{(t)} = h_{\mathbf{\Theta}}^{(t)} \left(\mathbf{x}_{v \leftarrow v}^{(t)}, \left\{ \mathbf{x}_{v \leftarrow w}^{(t)} : w \in \mathcal{N}(v) \right\} \right)
$$

其中邻居节点的Embedding由"QKV"式的attention计算：

$$
\mathbf{x}_{v \leftarrow w}^{(t)} = \mathrm{Attention}\left(\mathbf{x}_v^{(t-1)}, \mathbf{\Theta}_Q^{(t)}, [\mathbf{x}_w^{(1)}, \ldots, \mathbf{x}_w^{(t-1)}], \mathbf{\Theta}_K^{(t)}, , [\mathbf{x}_w^{(1)}, \ldots, \mathbf{x}_w^{(t-1)}], \mathbf{\Theta}_V^{(t)} \right)
$$

其中 $\mathbf{\Theta}_Q^{(t)}, \mathbf{\Theta}_K^{(t)}, \mathbf{\Theta}_V^{(t)}$ 分别代表query, key 和 value 的隐射矩阵。

更多：待更新

池化层的设计

池化层其实就是上文中提到的各个aggregator，用来将邻居的Embedding聚合起来生成一个汇总的Embedding再和节点自身的Embedding进行操作。

1.SumPooling顾名思义，将邻居Embedding的每一维求和：

$$r^{(i)} = \sum_{k=1}^{N_i} x_k^{(i)}$$

其中$x_k^{(i)}$表示邻居k Embedding的第i维。

2.AvgPooling将邻居Embedding的每一维求均值：

$$r^{(i)} = \frac{1}{N_i} \sum_{k=1}^{N_i} x_k^{(i)}$$

3.MaxPooling将邻居Embedding的按每一维取最大值：

$$r^{(i)} = \max_{k=1}^{N_i} \left(x_k^{(i)} \right)$$

4.SortPooling来自论文 An End-to-End Deep Learning Architecture for Graph Classification ，这篇文章的主要思路是通过 WL算法 可以对节点进行着色，而节点的颜色可以定义节点之间的次序，有了节点的次序，我们就可以通过1-D卷积的方法进行卷积运算。因为过程比较抽象，具体请参考原论文。

5.TopKPooling来自论文 Graph U-Nets ，论文的主要思路是将节点Embedding隐射到1维空间中选择其中top k个节点再进行图卷积的计算，以下是构造top k节点小图的选取过程：

$$\begin{aligned}
\mathbf{y} &= \frac{\mathbf{X}\mathbf{p}}{\|\mathbf{p}\|} \\
\mathbf{i} &= \text{top}_k(\mathbf{y}) \\
\mathbf{X}' &= (\mathbf{X} \odot \tanh(\mathbf{y}))_{\mathbf{i}} \\
\mathbf{A}' &= \mathbf{A}_{\mathbf{i},\mathbf{i}}
\end{aligned}$$

也可以设置一个阈值 $\tilde{\alpha}$：

$$\begin{aligned}
\mathbf{y} &= \text{softmax}(\mathbf{X}\mathbf{p}) \\
\mathbf{i} &= \mathbf{y}_i > \tilde{\alpha} \\
\mathbf{X}' &= (\mathbf{X} \odot \mathbf{y})_{\mathbf{i}} \\
\mathbf{A}' &= \mathbf{A}_{\mathbf{i},\mathbf{i}},
\end{aligned}$$

6.SAGPooling来自论文 Self-Attention Graph Pooling，论文可以看做是TopKPooling的衍生工作，在TopKPooling中，节点的排序因子y按照线性映射的方式生成，而在这篇文章的工作中，作者是用GNN的方法来生成节点排序因子：

$$\begin{aligned}
\mathbf{y} &= \text{GNN}(\mathbf{X}, \mathbf{A}) \\
\mathbf{i} &= \text{top}_k(\mathbf{y}) \\
\mathbf{X}' &= (\mathbf{X} \odot \tanh(\mathbf{y}))_{\mathbf{i}} \\
\mathbf{A}' &= \mathbf{A}_{\mathbf{i},\mathbf{i}}
\end{aligned}$$

对应的设置阈值$\tilde{\alpha}$的版本：

$$
\begin{aligned}
\mathbf{y} &= \text{softmax}(\text{GNN}(\mathbf{X}, \mathbf{A})) \\
\mathbf{i} &= \mathbf{y}_i > \tilde{\alpha} \\
\mathbf{X}' &= (\mathbf{X} \odot \mathbf{y})_{\mathbf{i}} \\
\mathbf{A}' &= \mathbf{A}_{\mathbf{i},\mathbf{i}},
\end{aligned}
$$

7.GlobalAttentionPooling来自论文 Gated Graph Sequence Neural Networks 对图中所有节点进行pooling，主要用来做对整个图的分类等任务：

$$
r^{(i)} = \sum_{k=1}^{N_i} \text{softmax}\left(f_{gate}\left(x^{(i)}k\right)\right) f_{feat}\left(x_k^{(i)}\right)
$$

其中$r^{(i)}$ 被称作“读出器”，是对于整个图的Embedding描述。

8.Set2Set来自论文 Order Matters: Sequence to sequence for sets，和Graph attention 有点类似，其中加权求和的权重是通过LSTM建模得到的：

$$
\begin{aligned}
q_t &= \text{LSTM}(q_{t-1}^*) \\
\alpha_{i,t} &= \text{softmax}(x_i \cdot q_t) \\
r_t &= \sum_{i=1}^{N} \alpha_{i,t} x_i \\
q_t^* &= q_t \| r_t
\end{aligned}
$$

9.SetTrasformerEncoder & SetTrasformerDecoder来自论文 Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks，是Set2Set的衍生模型，公式也有点复杂，可以查下原论文

相关阅读

综述|从9篇研究综述看图神经网络GNN的最新研究进展

- end-

阅读原文