我知道你们也没多少时间看了,所以我大概总结了一下:

第二点讲的是 GAT(Garph Attention Networks)架构以及 GAT 的优点.第三点讲的是使用 GAT 进行实验,分为两类:半监督学习(Transductive Learning)与归纳学习(Inductive learning),其中 Transductive Learning 与半监督嵌入,流体正则化,Deep-Walk,ICA 与 Planetoid 进行了比较,Inductive Learning 与四个监督 GraphSAGE 归纳方法进行比较,主要就是那四个聚合器(GCN,mean,LSTM,pool).

2 GAT 架构

2.1 图注意力层

2.1.1 输入与输出

图注意力层需要一个节点特征集 h 作为输入:

$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

其中 N 为节点个数,F 为每个节点的特征个数.

输出一个新的节点特征集 h':

$$\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$$

按照节点输入特征预测节点输出特征,其中每个节点有 F'个特征.

2.1.2 初始化步骤

为了有足够的表达能力对高层次的输入特征进行变换,至少需要一个可学习的线性变换.因此,一个通过权重矩阵 W 参数化的线性变换会应用到每一个节点:

$$\mathbf{W} \in \mathbb{R}^{F' \times F}$$

这个矩阵表示输入节点的 F 个特征与输出节点 F'个特征的关系.

2.1.3 self-attention 与 masked-attention

然后在节点上进行"自关注(self-attention)",这个共同的注意力机制 a 为:

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$$

据此计算注意力相关系数:

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

这个公式表示节点 j 的特征对节点 i 的特征的重要性,h 就是前面的特征向量.

作者通过”相邻注意”(masked-attention),去把图注意力机制注入到图结构.这个”相邻注意”具体来说就是对于计算$e_{ij}$,只计算 i 的邻居节点 j.

2.1.4 正则化

下一步是使用 softmax 函数进行正则化,用于对所有 i 的相邻节点 j 进行正则化:

$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k\in\mathcal{N}_i}\exp(e_{ik})}.$$

2.1.5 LeakyReLU 激活

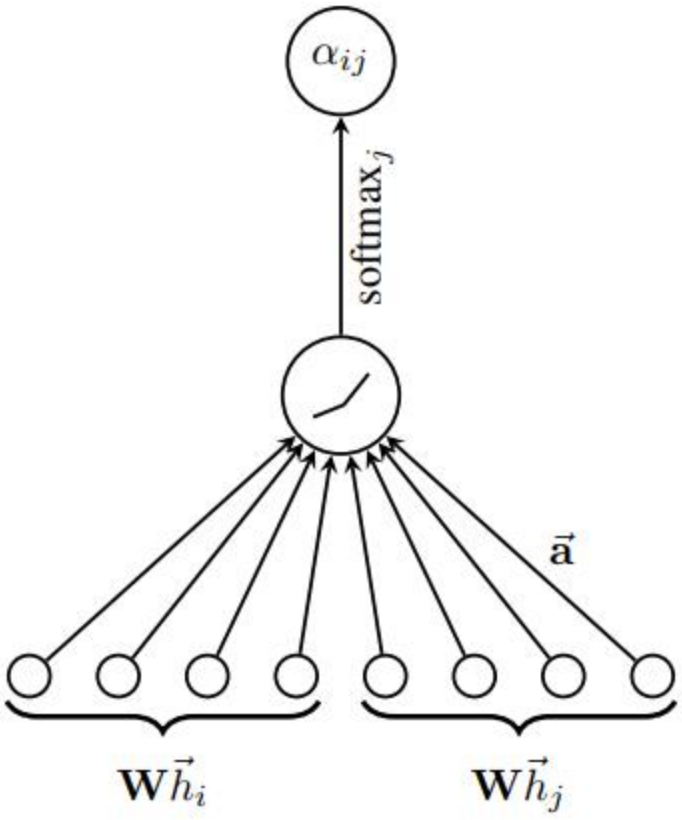
在这个实验中,注意力机制 a 是一个单层的前馈神经网络,通过一个权重向量:

$$\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$$

进行参数化,并加入 LeakyReLU 非线性激活(输入的负斜率为 0.2),然后就可以计算注意力相关系数了:

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

上标 T 表示转置,||表示串联运算(concatenation operation),图示如下:




四步:

1.权重矩阵 W 分别和 i 的特征 hi 与 j 的特征 hj 相乘

2.把结果串联在一起,再与 $\vec{\mathbf{a}}$ 相乘

3.经过 LeakyReLU 进行激活

4.使用 softmax 进行归一化得出注意力系数 aij

2.1.6 计算输出特征

得到注意力系数后,通过如下公式计算输出特征:

$$\vec{h}_i' = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j\right).$$

其中:

(1)$W\vec{h}_j$表示权值矩阵 W 与特征 hj 相乘的矩阵

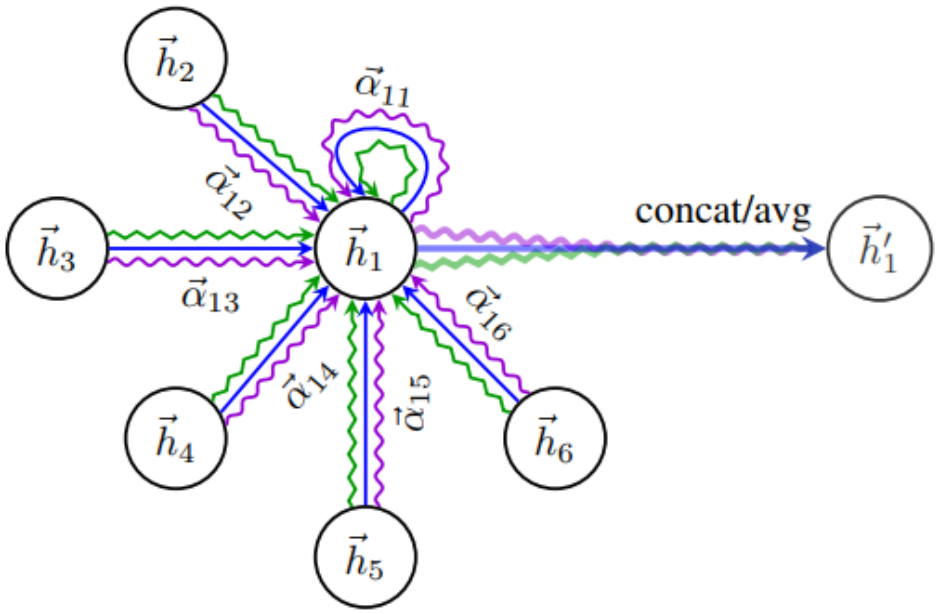
(2)$\alpha_{ij}$为注意力系数

(3)∑表示遍历所有 i 的相邻节点 j

(4) $\sigma$为非线性激活函数

2.1.7 多端注意机制

“多端注意”(multi-head attention)机制用于稳定化”自注意”(self-attention)的学习过程,可以采用两种方式:串联或 K 平均.



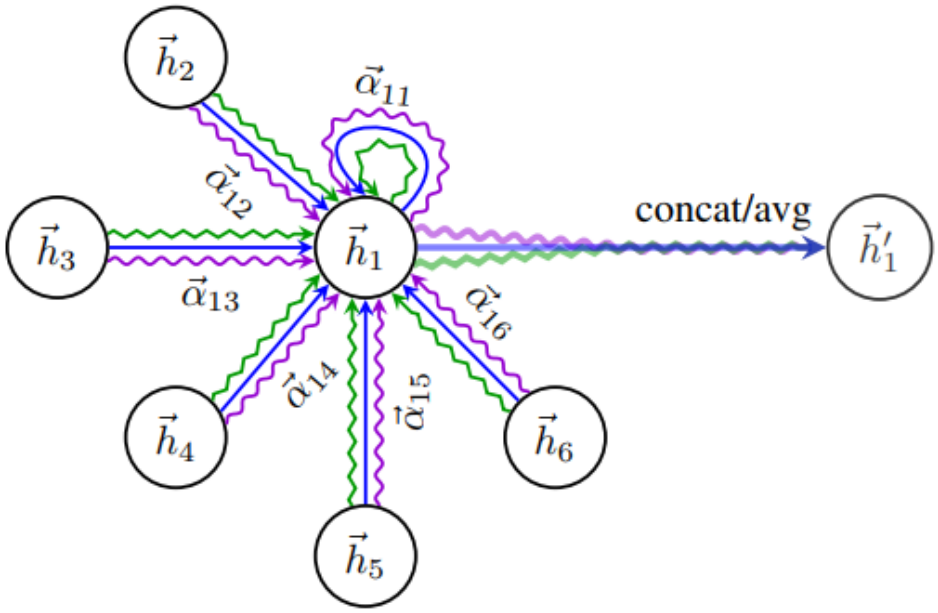

2.1.7.1 串联

把 K 个特征串联起来,输出特征如下:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

||表示串联操作, $a_{ij}^k$是由第 k 个注意力机制$a^k$计算出来的正则化注意力系数,一共有大 K 个”端”需要计算,$W^k$为对应输入特征的线性变换权重矩阵,最终输出为 h’,由 KF’个特征组成(而不是原来的 F’个特征).

当 K=3 时,计算如下:




节点 h1 具有多端注意机制,不同风格颜色的箭头表示独立的注意力计算,从每个端聚合特征后串联(即图中的 concat)得到 h1'.

2.1.7.2 K 平均

如果使用多端注意机制计算最终预测层的输入,串联操作可能不那么敏感,换句话说,可以使用均值化(K 平均,即图中的 avg)后再应用到最终的非线性函数(通常使用分类问题中的 softmax 或 logistic sigmoid):

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

2.2. 优点

2.2.1 计算高效

时间复杂度为:

$$O(|V|FF' + |E|F'),$$

自注意力层可以并行计算每条边,可以并行计算每个节点得出输出特征.

2.2.2 更好鲁棒性

不同于 GCNs,GAT 能够对同一邻居的不同相邻节点给予不同的重要性.

2.2.3 不需要整张 Graph

引入注意力机制后,只与相邻节点有关,与共享边节点无关

2.2.4 强于 LSTM

2.2.5 可以作为 MoNet 的一个实例

(优点这部分就不怎么翻译了…)

3 实验

实验分为两部分,半监督学习(Transductive Learning)与归纳学习(Inductive Learning).

3.1 数据集

Table 1: Summary of the datasets used in our experiments.

| | Cora | Citeseer | Pubmed | PPI |
|---|---|---|---|---|
| **Task** | Transductive | Transductive | Transductive | Inductive |
| **# Nodes** | 2708 (1 graph) | 3327 (1 graph) | 19717 (1 graph) | 56944 (24 graphs) |
| **# Edges** | 5429 | 4732 | 44338 | 818716 |
| **# Features/Node** | 1433 | 3703 | 500 | 50 |
| **# Classes** | 7 | 6 | 3 | 121 (multilabel) |
| **# Training Nodes** | 140 | 120 | 60 | 44906 (20 graphs) |
| **# Validation Nodes** | 500 | 500 | 500 | 6514 (2 graphs) |
| **# Test Nodes** | 1000 | 1000 | 1000 | 5524 (2 graphs) |

其中 Cora,Citeseer,Pubmed 用于半监督学习,PPI 用于归纳学习.

3.2 对比的方法

对于半监督学习,比较的方法有 LP(label propagation),SemiEmb,(semi-supervised embedding,半监督嵌入),ManiReg(manifold regularization,流体正则化),Deep-Walk(skip-gram based graph embeddings,基于 skip-gram 的图嵌入),ICA(the iterative classification algorithm,迭代分类算法),Planetoid.同时直接把 GAT 模型与 GCNs 模型,高阶 Chebyshev 滤波器图卷积模型以及 MoNet 模型进行了比较.

对于归纳学习,比较的方法有四个不同的半监督 GraphSAGE 归纳方法:GraphSAGE-GCN(对归纳参数扩展了图卷积风格的操作),GraphSAGE-mean(计算特征向量的元素均值),GraphSAGE-LSTM(对邻域特征注入 LSTM 进行聚合)与 GraphSAGE-pool(进行 共享非线性多层感知器转换的特征向量中的元素最大化操作).

3.3 实验建立

对于半监督学习:

- 应用了两层 GAT 模型

- 第一层包含 K=8 的多注意力端,去计算 F’=8 个特征(总共 64 个特征),ELU 作为非线性激活函数(exponential linear unit)

- 第二层用于分类,一个注意力端计算 C 个特征(C 为分类数目),后跟 softmax 激活.

- 为了应对小型训练集,训练过程中会应用$\lambda = 0.0005$的 L2 正则化.$p = 0.6$ 的 dropout 会应用到这两层的输入以及归一化注意力系数.

对于归纳学习

- 应用了三层 GAT 模型

- 第一第二层包含 K=4 的多注意力端,去计算 F’=256 个特征(总共 1024 个特征),后跟 ELU 作为激活函数.

- 最后一层用于多标签分类,包含 K=6 的多注意力端,去计算 F’=121 个特征,激活函数为 logistic sigmoid.

两个模型都使用 Glorot 初始化进行初始化,训练过程中使用 Adam SGD 优化器优化最小化交叉熵.

3.4 结果

半监督学习:

| | ***Transductive*** | | |
|---|---|---|---|
| **Method** | **Cora** | **Citeseer** | **Pubmed** |
| MLP | 55.1% | 46.5% | 71.4% |
| ManiReg (Belkin et al., 2006) | 59.5% | 60.1% | 70.7% |
| SemiEmb (Weston et al., 2012) | 59.0% | 59.6% | 71.7% |
| LP (Zhu et al., 2003) | 68.0% | 45.3% | 63.0% |
| DeepWalk (Perozzi et al., 2014) | 67.2% | 43.2% | 65.3% |
| ICA (Lu & Getoor, 2003) | 75.1% | 69.1% | 73.9% |
| Planetoid (Yang et al., 2016) | 75.7% | 64.7% | 77.2% |
| Chebyshev (Defferrard et al., 2016) | 81.2% | 69.8% | 74.4% |
| GCN (Kipf & Welling, 2017) | 81.5% | 70.3% | **79.0%** |
| MoNet (Monti et al., 2016) | 81.7 ± 0.5% | — | 78.8 ± 0.3% |
| GCN-64* | 81.4 ± 0.5% | 70.9 ± 0.5% | **79.0** ± 0.3% |
| **GAT** (ours) | **83.0** ± 0.7% | **72.5** ± 0.7% | **79.0** ± 0.3% |

归纳学习:

| *Inductive* | |
|---|---|
| **Method** | **PPI** |
| Random | 0.396 |
| MLP | 0.422 |
| GraphSAGE-GCN (Hamilton et al., 2017) | 0.500 |
| GraphSAGE-mean (Hamilton et al., 2017) | 0.598 |
| GraphSAGE-LSTM (Hamilton et al., 2017) | 0.612 |
| GraphSAGE-pool (Hamilton et al., 2017) | 0.600 |
| GraphSAGE* | 0.768 |
| Const-GAT (ours) | 0.934 ± 0.006 |
| **GAT** (ours) | **0.973** ± 0.002 |

GAT 最厉害就对了….具体结果就不翻译了,看原文.

(偷偷给你们)几个问题(应该也许或许可能或者大概能在明天晚上提问):

1.文章使用了 LeakyReLU,其中提到了负斜率,请问是什么意思?其他几种 ReLU 函数的斜率是怎么样的?

2.如何理解 LeakyReLU 中的串联操作?

3.正则化过程中用到了 softmax 函数,什么是 softmax 函数?

4.计算输出特征过程中使用的非线性激活函数可以是什么函数?

5.多端注意机制中的串联操作与 K 平均怎么理解?

6.建立实验的过程中用到了 ELU 函数,请问什么是 ELU 函数,对该实验的作用?

7.GAT 的优点?

8.什么是 L2-正则化? $\lambda$如何影响 L2 正则化?

9.什么是 Glorot 初始化?什么是 Adam SGD 优化器?