两篇论文–标签位置偏差和尾节点偏差

何放　邓金海

Week3, April 2024

Table of Contents

Table of Contents

Towards Label Position Bias in Graph Neural Networks

解决的问题

Nodes far away from labeled nodes tend to yield subpar performance.

例

应用场景 例如，在金融体系中，标签位置偏见可能导致对远离标签的个人的不公平评估，可能会剥夺他们获得金融资源的机会。
其次，减轻这种偏差有可能提高gnn的性能，特别是如果可以正确分类距离较远的节点。

- Label Position Bias
- 提出了Label Proximity Score(LPS)来度量这种bias
- 提出了Label Position unbiased Structure Learning method (LPSL)来减轻Label Position Bias的图结构
- 目标：学习一个新的图结构，其中每个节点将会有相似LPS

LPS

定义

LPS是本文提出的一个度量标准，用于量化有标签的训练节点和测试节点之间的“接近度”

$$LPS = \mathbf{PT1_n}, \text{ and } \mathbf{P} = \left(\mathbf{I} - (1-\alpha)\tilde{\mathbf{A}}\right)^{-1}$$

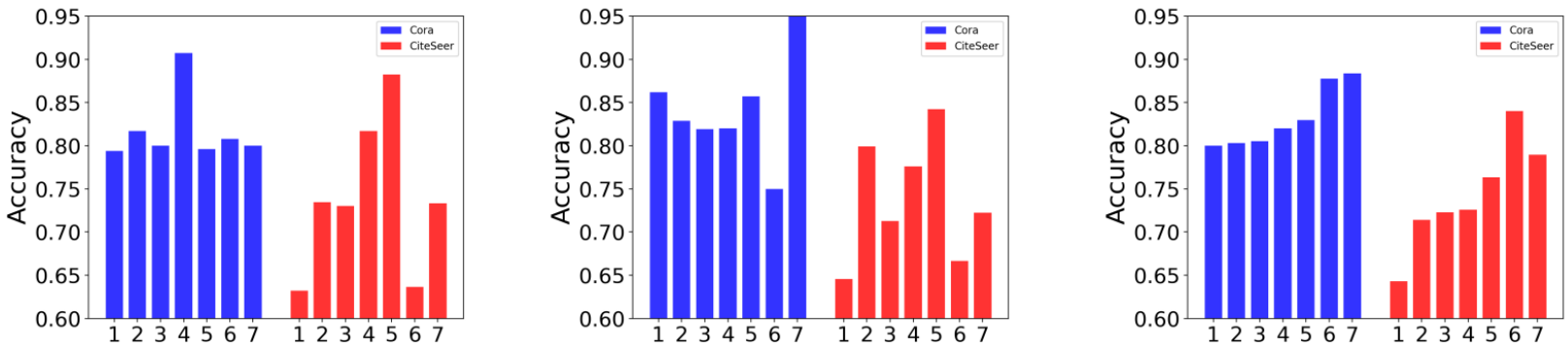
- P 是Personalized PageRank matrix（在图中从随机顶点开始，在每次迭代中以概率α跳至随机顶点，并以概率1-α跟随当前顶点的随机输出边缘，PPR会跳回起始顶点，更加个性化）
- T is the label mask matrix
- 1n 是全1列向量
- α 是跳转概率，范围(0，1]

Sensitive Groups

bias指标对比

- 节点的度——根据度的实际值划分7个sensitive groups
- 最短路径距离——根据最短路径距离的值划分7个sensitive groups
- 注意，在图中只有极少数节点的度或最短路径距离大于7
- LPS——根据公式计算LPS的值，将测试节点均匀地划分为7个sensitive groups，每组范围等长

在采用与APPNP相同的超参设置下，三种度量的不同组精确度如下



(a) Degree (b) Shortest Path Distance (c) Label Proximity Score

Figure 1: APPNP with 20 labeled nodes per class on Cora and CiteSeer datasets.

结论：LPS与性能差异表现出更一致和更强的相关性，且高度和短路径节点的LPS更高

Framework

图的结构是导致label position bias的主要来源

challenge1

How can we define a label position unbiased graph structure,
and how can we learn this structure based on the original graph?

challenge2

how can we ensure that the learned data structure is also sparse to avoid excessive memory consumption?

LPSL

learn a new graph structure in which each node exhibits similar LPSs

$$\arg\min_{\mathbf{B}} \|\mathbf{I}-\mathbf{B}\|_F^2 + \lambda \mathrm{tr}(\mathbf{B}^\top \tilde{\mathbf{L}}\mathbf{B})$$

$$\text{s.t.} \quad \mathbf{BT1}_n = c\mathbf{1}_n,$$

测量平滑度

$$tr\left(B^T \tilde{L} B\right)$$

- B debiased graph structure matrix
- I 鼓励自环，为了避免过度平滑
- λ 控制平衡和自环的比例
- c 该超参鼓励新图结构中节点有相同的LPS，从而减轻label position bias

约束优化

The constrained optimization problem is a convex optimization problem. It can be solved by the Lagrange Multiplier method.

$$L_\rho(\mathbf{B},\mathbf{y}) = \|\mathbf{I}-\mathbf{B}\|_F^2 + \lambda \mathrm{tr}(\mathbf{B}^\top \tilde{\mathbf{L}}\mathbf{B}) + \mathbf{y}^\top(\mathbf{B}\mathbf{T}\mathbf{1}_n - c\mathbf{1}_n) + \frac{\rho}{2}\|\mathbf{B}\mathbf{T}\mathbf{1}_n - c\mathbf{1}_n\|_2^2,$$

y是拉格朗日乘子， ρ 为增广拉格朗日参数
L对B求梯度:

$$\frac{\partial L_\rho}{\partial \mathbf{B}} = 2(\mathbf{B}-\mathbf{I}) + 2\lambda\tilde{\mathbf{L}}\mathbf{B} + \mathbf{y}(\mathbf{T}\mathbf{1}_n)^\top + \rho(\mathbf{B}\mathbf{T}\mathbf{1}_n - c\mathbf{1}_n)(\mathbf{T}\mathbf{1}_n)^\top.$$

然后可以用双上升算法（dual ascent algorithm）来求解

$$\mathbf{B}^{k+1} = \arg\min_{\mathbf{B}} L_\rho(\mathbf{B}^k, \mathbf{y}^k)$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\mathbf{B}^k\mathbf{T}\mathbf{1}_n - c\mathbf{1}_n),$$

understanding

将学习到的图结构B直接作为传播矩阵进行特征聚合。如果B是LPSL公式定义的无约束的主要问题的近似或精确解，则相当于在原始图中应用GNN消息传递

就是说我们可以直接用学习到的B代替GNN中的传播矩阵

l1-regularized Label Position Unbiased Sparse Structure Learning

根据上述推导，可能会学习到一个很密集的B，但是为了使任务耗时小，要将B稀疏化

$$\arg\min_{B} \|\mathbf{I}-\mathbf{B}\|_F^2 + \lambda \mathrm{tr}(\mathbf{B}^\top \mathbf{L}\mathbf{B}) + \beta\|\mathbf{B}\|_1$$
$$\text{s.t.} \quad \mathbf{B}\mathbf{T}\mathbf{1}_n = c\mathbf{1}_n,$$

- 鼓励B中有尽可能多的0

β 是控制B稀疏性的超参数

$$\frac{\partial L_\rho}{\partial \mathbf{B}_{:,j}} = 2(\mathbf{B}_{:,j}-\mathbf{I}_{:,j}) + 2\lambda\tilde{\mathbf{L}}\mathbf{B}_{:,j} + \mathbf{y}(\mathbf{T}\mathbf{1}_n)_j^\top + \rho(\mathbf{B}\mathbf{T}\mathbf{1}_n - c\mathbf{1}_n)(\mathbf{T}\mathbf{1}_n)_j^\top$$

Algorithm of LPSL

Algorithm 1 Algorithm of LPSL

Input: Laplacian matrix $\tilde{\mathbf{L}}$, Label mask matrix $\mathbf{T}$, Hyperparamters λ, c, β, ρ, learning rate γ
Output: Label position unbiased graph structure $\mathbf{B}$

Initialization: $\mathbf{B}^0 = \mathbf{I}$ and $\mathbf{y}^0 = \mathbf{0}$
while Not converge **do**
  for each block j **do**
    for $i = 0$ **to** update steps t **do**
      $\mathbf{B}_{:,j} = \mathbf{B}_{:,j} - \gamma * \frac{\partial L_\rho}{\partial \mathbf{B}_{:,j}}$
    end for
    $\mathbf{B}_{:,j} = S_{\beta/\rho}(\mathbf{B}_{:,j})$
  end for
  $\mathbf{y} = \mathbf{y} + \rho(\mathbf{BT1}_n - c\mathbf{1}_n)$
end while

return $\mathbf{B}$

实验对比、评估、调参

在GCN和APPNP模型上测试了LPSL学习到的无偏图结构B

- 在几乎所有用GCN和APPNP模型的数据集上都有更高的性能
- 对比不同的标记率（标记节点数量占比），LPSL在低标记率下表现出了很大的性能提升

展望

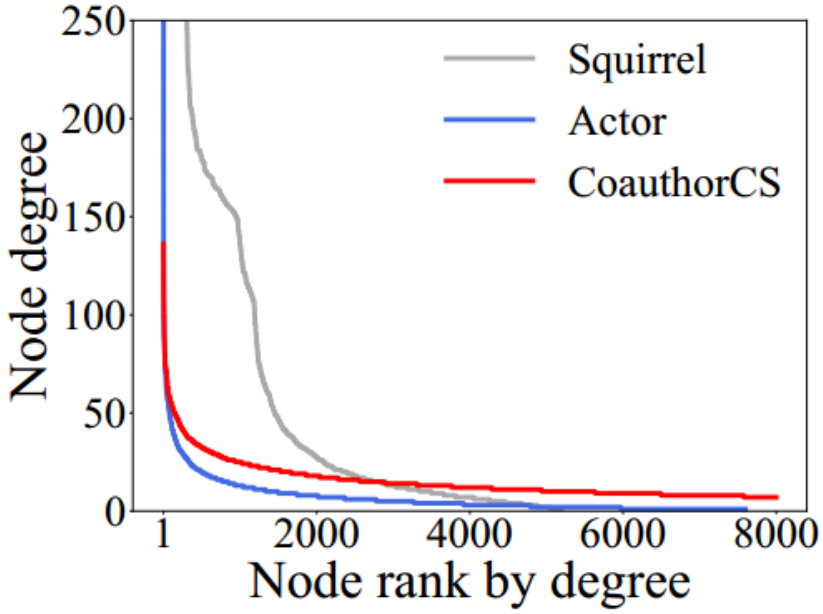
未来可在本文学习新图结构的基础之上纳入特征信息
此外，作者还初步研究了同质图(homophily graphs)。研究标签位置偏差如何影响异质图(heterophily graphs)是一个值得探索的问题。

Table of Contents

Tail-GNN

motivation

现实生活中很多结构都是长尾(long-tailed)的
于是就出现了一个问题：那些head nodes(high-degree)拥有足够的邻居去聚合信息，但是一些tail-node(low-degree)就会出现没有足够的邻居去聚合的情况



(a) Long-tailed node distribution

Tail-GNN

文章做的主要工作：

- 新的模型Tail-GNN
- Neighborhood translation——是为了增强尾节点的表示学习能力，并在此基础上进一步设计了一种头尾知识迁移方案

模型的定义如下：

- 对于节点v，Nv为v的相邻节点集

K 阈值K，对于一个节点，它的度不超过K则为尾节点。本文K作为一个预定的超参

- 是否可以通过自适应机制，使K适合每个图，达到更好的效果呢？

Neighborhood translation

受知识图谱启发，对节点v与其邻域Nv之间的关系关系进行建模

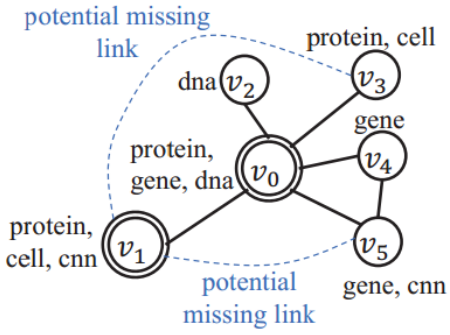
- hv表示头部节点v的嵌入向量，hNv表示v的邻域的嵌入向量
- 同时，为了获取tail node缺失的邻居信息，Tail-GNN设置了一个可以学习的参数r

这个参数是通过整图的信息学到的，再通过聚合tail-node自身的信息，就可以让每一个节点有一个r_{v_i} rv表示的是自己的特征信息到自己邻居聚合之后的信息之间的信息差

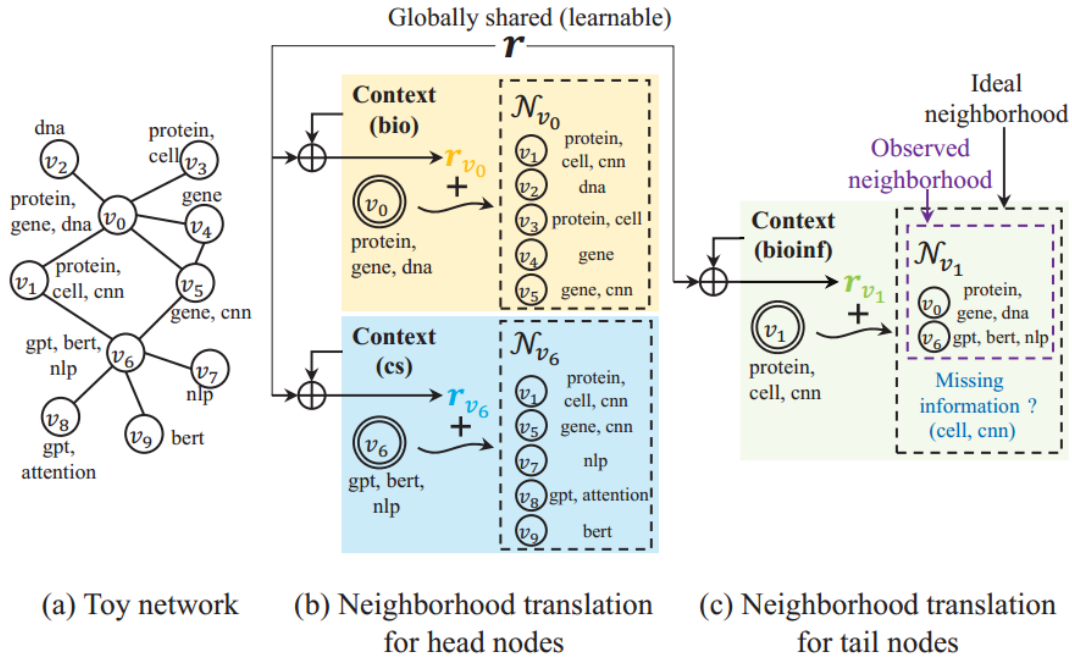
Neighborhood translation可以表示为

$$h_v + r_v \approx h_{N_v}$$

tail-GNN framework

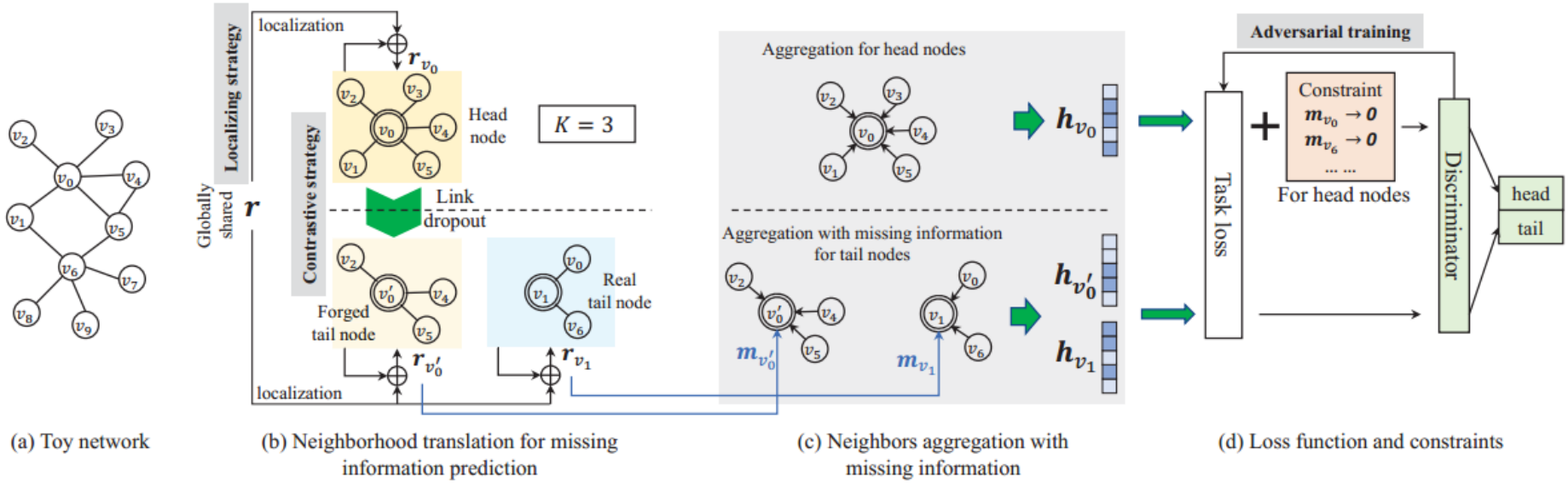


(b) Toy citation network



(a) Toy network (b) Neighborhood translation for head nodes (c) Neighborhood translation for tail nodes

- 对于头节点，自己在第L层的hv+rv应与实际聚合信息一致
- 将head-node作为训练集，tail-node再使用head-node训练出来的r来填补缺失的邻居信息



为了增强对比能力，用伪造的尾部节点来补充尾部节点，
伪造的尾部节点可以通过从头部节点随机删除一些链接来模拟实际的尾部节点来生成。它们与相应的头节点进行对比，以增强对缺失邻域信息的预测和利用。

- 尾巴节点v-tail的缺失信息，用mv表示
- mv由其理想邻域的嵌入向量与观测邻域的嵌入向量之差给出

$$\mathbf{m}_v = \mathbf{h}_{\mathcal{N}_v^*} - \mathbf{h}_{\mathcal{N}_v}.$$

Realizing neighborhood translation

对于每一层网络，都有一个可以学习的向量r^l,同时也有一个函数 $\emptyset$ 可以将这个共享向量转化为针对于每个节点独特的r_v^l
公式如下：

$$\mathbf{r}_v^l = \phi(\mathbf{h}_v^l, \mathbf{h}_{\mathcal{N}_v}^l, \mathbf{r}^l; \theta_\phi^l),$$

通过放缩可以近似得到：

$$\mathbf{r}_v^l = \phi(\mathbf{h}_v^l, \mathbf{h}_{\mathcal{N}_v}^l, \mathbf{r}^l; \theta_\phi^l) = (\gamma_v^l + 1) \odot \mathbf{r}^l + \beta_v^l,$$

它相当于对每一层的共享r^l都做了一个变换，这里借鉴了scaling and shifting transformations的方法，来得到 个性化的r

$$\gamma_v^l = \text{LeakyReLU}(\mathbf{W}_\gamma^{l,1}\mathbf{h}_v^l + \mathbf{W}_\gamma^{l,2}\mathbf{h}_{\mathcal{N}_v}^l)$$
$$\beta_v^l = \text{LeakyReLU}(\mathbf{W}_\beta^{l,1}\mathbf{h}_v^l + \mathbf{W}_\beta^{l,2}\mathbf{h}_{\mathcal{N}_v}^l)$$

损失函数

task loss

文章中做的是分类任务，所以使用交叉熵损失

$$\mathcal{L}_t = \sum_{v \in \mathcal{V}_{\text{tr}}} \textsc{CrossEnt}(\mathbf{h}_v^\ell, \mathbf{y}_v) + \lambda_t \|\Theta\|_2^2,$$

missing information loss

我们是希望head-node的embedding加上我们学习出的r_v之后仍然和其理想的embedding基本是一致的，因为head-node是被认为没有信息缺失的

$$\mathbf{m}_v^l = \mathbf{h}_{\mathcal{N}_v^*}^l - \mathbf{h}_{\mathcal{N}_v}^l = \mathbf{h}_v^l + \mathbf{r}_v^l - \mathbf{h}_{\mathcal{N}_v}^l.$$

$$\mathcal{L}_m = \sum_{v \in \mathcal{V}_{\text{tr}}} I_v \sum_{l=1}^{\ell} \|\mathbf{m}_v^{l-1}\|_2^2,$$

损失函数

adversarial constraint loss

为了让结果更加的鲁棒性，最后加上一个对抗训练的损失函数
对抗训练的思路是设置判别器，用来判断输入节点是tail-node还是head-node，判别器和Tail-GNN交替训练，把 Tail-GNN当作生成器
以下为生成器损失，含义是，如果Tail-GNN的效果很好，那判别器看到一个节点向量，应该无法区分它是tail-node还是head-node（就是让尾节点的标签是"头节点"，让头节点的标签是"尾节点"

$$\mathcal{L}_d = \sum_{v \in \mathcal{V}_{\text{tr}}} \textsc{CrossEnt}(I_v, D(\mathbf{h}_v^\ell; \theta_d)) + \lambda_d \|\theta_d\|_2^2,$$

其中D是判别器计算相似度分数的函数，用于计算节点为头节点的概率

$$D(\mathbf{h}_v^\ell; \theta_d) = \sigma(\mathbf{w}_d^\top \textsc{LeakyReLU}(\mathbf{W}_d \mathbf{h}_v^\ell + \mathbf{b}_d)),$$

最后，把三个损失加在一起就成了整体损失

$$\min_\Theta \max_{\theta_d} \; \mathcal{L}_t + \mu \mathcal{L}_m - \eta \mathcal{L}_d,$$

结论和展望

- tail-GNN——缩小头尾节点的差距，从而实现较稳定的尾部节点嵌入，
- neighborhood translation——进一步定位以适应每个节点的本地上下文，预测尾节点缺失的邻域信息，以补充它们的邻域聚合

展望：计划利用高阶邻域信息进行尾节点嵌入