LAB4

PB19000196 晏瑞然

目录

提交文件

本次实验提交`node_class_demo.py`,`link_pred_demo.py`,`ppi_node_class_demo.py`,`ppi_link_pred_demo.py`共四个文件，其中`node_class_demo.py`、`link_pred_demo.py`是对cora和citeseer数据集的实现，另外两个是对PPI数据集的实现，所有的影响因素的测试都已实现，只需将注释去掉即可。所有的文件可直接执行，得到的结果为baseline的结果。

代码细节

修改数据集

```
dataset = Load_graph('citeseer', 'edge')
```

```
dataset = Load_graph('cora', 'node')
```

主要通过修改Load_graph中的参数修改数据集内容以及任务类型。

而对于ppi，需要如下操作：

- link pred: 将每个图单独进行处理，然后封装成三元组`[train_g,train_pos_g,train_neg_g]`，将所有的三元组通过`simple_dataloader`变成dataloader，之后只需在train和test的时候通过itr遍历里面的三元组即可(test set的处理相同)，注意要将`simple_dataloader`中的`shuffer`设为`False`，以达到数据集的对齐。
- node class: 将20份数据分成**16：2：2**的training set : val set : test set，训练过程中对每个train graph进行训练，loss使用**BCE with logits loss**，即sigmoid层后加一层BCEloss，之后acc的计算是logits经过sigmoid后，所有的121维label预测正确的的数量计算出来，当然也可以选择更好的评价指标。

具体实现见`ppi_link_pred`和`ppi_node_class`文件。

self-loop

在**link_pred**中通过

```
...
train_pos_g = dgl.remove_self_loop(train_pos_g)
train_neg_g = dgl.remove_self_loop(train_neg_g)
test_pos_g = dgl.remove_self_loop(test_pos_g)
test_neg_g = dgl.remove_self_loop(test_neg_g)
...
```

移除self-loop。

通过

```
...
train_pos_g = dgl.add_self_loop(train_pos_g)
train_neg_g = dgl.add_self_loop(train_neg_g)
test_pos_g = dgl.add_self_loop(test_pos_g)
test_neg_g = dgl.add_self_loop(test_neg_g)
...
```

加入self-loop。

在**node_class**中通过

```
g = dgl.remove_self_loop(g)
g = dgl.add_self_loop(g)
```

移除/加入self-loop.

注意：对于citeseer数据集需要在GraphConv中加入`allow_zero_in_degree=True`参数以允许0入度的节点，否则对于remove_self_loop无法执行。

number of layers

对于link pred，通过在GraphSAGE中加入SAGEConv层来达到添加层数的效果，实验对比了2层(base)和3层的结果。

对于node class，通过在GCN中加入GraphConv层实现层数的增加，实验同样对比了2层(base)和3层的结果。

DropEdge

对于link pred，只需在GraphSAGE中加入

```
self.dropout = DropEdge()
...
g = self.dropout(g)
```

而对于node class，同理，但要在train的过程中加入model.train()以及model.eval()进行训练模式的切换。同时经过测试，**需要将p设为很小才会有效果。**

PairNorm

由于DGL不提供PairNorm的函数，所以只能自己实现，实现代码如下：

```
class PairNorm(nn.Module):
    def __init__(self, mode='PN', scale=1):
        """
            mode:
              'None' : No normalization
              'PN'   : Original version
              'PN-SI'  : Scale-Individually version
              'PN-SCS' : Scale-and-Center-Simultaneously version

            ('SCS'-mode is not in the paper but we found it works well in
practice,
              especially for GCN and GAT.)
            PairNorm is typically used after each graph convolution operation.
        """
        assert mode in ['None', 'PN', 'PN-SI', 'PN-SCS']
        super(PairNorm, self).__init__()
        self.mode = mode
        self.scale = scale

        # Scale can be set based on origina data, and also the current feature
lengths.
        # We leave the experiments to future. A good pool we used for choosing
scale:
        # [0.1, 1, 10, 50, 100]

    def forward(self, x):
        if self.mode == 'None':
            return x

        col_mean = x.mean(dim=0)
        if self.mode == 'PN':
            x = x - col_mean
            rownorm_mean = (1e-6 + x.pow(2).sum(dim=1).mean()).sqrt()
            x = self.scale * x / rownorm_mean

        if self.mode == 'PN-SI':
            x = x - col_mean
            rownorm_individual = (1e-6 + x.pow(2).sum(dim=1,
keepdim=True)).sqrt()
            x = self.scale * x / rownorm_individual

        if self.mode == 'PN-SCS':
            rownorm_individual = (1e-6 + x.pow(2).sum(dim=1,
keepdim=True)).sqrt()
            x = self.scale * x / rownorm_individual - col_mean

        return x
```

这是论文原文的实现代码，本实验中使用默认参数(即mode='PN', scale=1)进行测试。之后在两个conv层之间加入norm层即可。

实验结果

实验结果对比

base使用demo中的测试参数直接进行测试，对每个trick的测试都时在base上修改并进行对比。对于citeseer和cora，link pred的评价指标是AUC，node class的是ACC。对于PPI，link pred的评价指标是所有图AUC的平均值

self-loop

数据集	任务类型	base	remove self-loop acc	add self-loop acc
citeseer	link pred	0.846	0.829	0.585
cora	link pred	0.869	0.861	0.605
PPI	link pred	0.823	0.821	0.598
citeseer	node class	0.633	0.612	0.672
cora	node class	0.763	0.776	0.802
PPI	node class	0.774	0.755	0.803

可以看出对于link pred任务remove和add都会降低AUC，特别是add self-loop操作会造成预测AUC的大幅下降，这可能是因为任务是对边的预测，而直接修改图中的边会导致预测不准，而节点分类就没有这个问题，加入自环和去掉自环，反而能提高一定的性能。

number of layers

数据集	任务类型	2层(base)	3层
citeseer	link pred	0.846	0.795
cora	link pred	0.869	0.809
PPI	link pred	0.821	0.778
citeseer	node class	0.633	0.502
cora	node class	0.763	0.743
PPI	node class	0.755	0.688

可以看出，增加不论是Link还是Node任务layer的增加都会导致过拟合使得准确率下降，也有可能是因为模型更大了，原有的epoch数不够，可以通过增加训练epoch来获得更好的结果。

DropEdge

数据集	任务类型	base	dropedge
citeseer	link pred	0.846	0.831
cora	link pred	0.869	0.827
PPI	link pred	0.821	0.642
citeseer	node class	0.633	0.627
cora	node class	0.763	0.730
PPI	node class	0.755	0.771

可以看出dropEdge同样是负优化居多，猜测可能的原因是模型训练的epoch还不够多，加入DropEdge导致了更多的欠拟合，增加训练epoch能很好的解决这个问题。

PairNorm

数据集	任务类型	base	pairnorm
citeseer	link pred	0.846	0.893
cora	link pred	0.869	0.858
PPI	link pred	0.821	0.838
citeseer	node class	0.633	0.588
cora	node class	0.763	0.691
PPI	node class	0.755	0.768

可以看出pairnorm的效果有增有减，当ACC无法提升的时候可以尝试这种方法。

activation function

将relu和sigmoid函数进行了对比

数据集	任务类型	base(relu)	sigmoid
citeseer	link pred	0.846	0.668
cora	link pred	0.869	0.639
PPI	link pred	0.821	0.655
citeseer	node class	0.633	0.627
cora	node class	0.763	0.705
PPI	node class	0.755	0.714

实验总结

本次实验通过GNN模型处理link predict和node classification两种任务，并对各种trick进行了复现，让我对GNN模型有了更深入的理解。

本实验难度适中，主要的难点在于PPI数据集的处理上，需要改变整个实验框架，虽然助教封装了load_graph，但处理起来还是很复杂，本实验大部分时间也花在这上面，原因主要是对DGL框架和GNN框架的不熟悉。

通过这次实验，我对GNN的模型和DGL框架更加熟悉，也算是我对GNN的一个入门，受益良多。