第三次实验报告

PB21020651 武宇星

实验过程

首先手动实现 gcnconv 层，然后分别构建节点分类和链路预测的模型，并根据任务分别处理数据集，如链路预测需要额外的训练集的负样本采样，最后进行调参选取验证集准确率/AUC 最高的模型参数在测试集上测试

关键代码展示

```python
import torch
import torch.nn as nn

def pair_norm(x):
        mean = x.mean(dim=0, keepdim=True)
        std = x.std(dim=0, keepdim=True)
        return (x - mean) / (std + 1e-6)

class GCNLayer(nn.Module):
    def __init__(self, in_channels, out_channels, normalize):
        super(GCNLayer, self).__init__()
        self.projection = nn.Linear(in_channels, out_channels)
        self.normalize = normalize

    def forward(self, node_feats, adj_matrix):
        adj_matrix = adj_matrix + torch.eye(adj_matrix.size(0), device=adj_matrix.device)
        deg_matrix = torch.diag(torch.sum(adj_matrix, dim=1))
        deg_inv_sqrt = torch.pow(deg_matrix, -0.5)
        deg_inv_sqrt[deg_inv_sqrt == float('inf')] = 0
        adj_matrix_normalized = torch.mm(torch.mm(deg_inv_sqrt, adj_matrix), deg_inv_sqrt)
        node_feats = self.projection(node_feats)
        if self.normalize:
            node_feats = pair_norm(node_feats)
        node_feats = torch.mm(adj_matrix_normalized, node_feats)
        return node_feats
```

(parameter) adj_matrix: Any

由图卷积的公式,根据邻接矩阵和度矩阵得到归一化拉普拉斯矩阵，以此进行图卷积

```python
class GCN(nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels, dropout, layer_num, drop_prob=0.0, normalize=True):
        super(GCN, self).__init__()
        self.layers = nn.ModuleList()
        self.layers.append(GCNLayer(in_channels, hidden_channels, drop_prob))
        for _ in range(layer_num - 2):
            self.layers.append(GCNLayer(hidden_channels, hidden_channels, drop_prob))
        self.layers.append(GCNLayer(hidden_channels, hidden_channels, drop_prob))
        self.activation = nn.ReLU()
        self.readout = nn.Linear(hidden_channels, out_channels)
        self.dropout = nn.Dropout(dropout)
        self.normalize = normalize

    def forward(self, x, adj_matrix):
        for layer in self.layers:
            x = layer(x, adj_matrix, self.normalize)
            x = self.activation(x)
            x = self.dropout(x)
        x = self.readout(x)
        return x
```

通过实施的图卷积层进行信息聚合，最后通过 MLP 进行分类任务，可直接用于节点分类

```python
class GCN_LP(nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels, dropout, layer_num, normalize, drop_prob):
        super(GCN_LP, self).__init__()
        self.drop_edge = drop_prob
        self.layers = nn.ModuleList()
        self.layers.append(GCNLayer(in_channels, hidden_channels, normalize))

        for _ in range(layer_num - 2):
            self.layers.append(GCNLayer(hidden_channels, hidden_channels, normalize))

        self.layers.append(GCNLayer(hidden_channels, out_channels, normalize))

        self.activation = nn.ReLU()
        self.dropout = nn.Dropout(dropout)

    def encode(self, x, edge_index):
        num_nodes = x.size(0)
        adj_matrix = torch.zeros((num_nodes, num_nodes), device=edge_index.device)
        adj_matrix[edge_index[0], edge_index[1]] = 1
        adj_matrix = adj_matrix + torch.eye(adj_matrix.size(0), device=adj_matrix.device)
        adj_matrix = drop_edge(adj_matrix, self.drop_edge)
        deg_matrix = torch.diag(torch.sum(adj_matrix, dim=1))
        deg_inv_sqrt = torch.pow(deg_matrix, -0.5)
        deg_inv_sqrt[deg_inv_sqrt == float('inf')] = 0
        adj_matrix_normalized = torch.mm(torch.mm(deg_inv_sqrt, adj_matrix), deg_inv_sqrt)

        for i, layer in enumerate(self.layers):
            x = layer(x, adj_matrix_normalized)
            if i < len(self.layers) - 1:
                x = self.activation(x)
                x = self.dropout(x)
        return x

    def decode(self, z, edge_label_index):
        src = z[edge_label_index[0]]
        dst = z[edge_label_index[1]]
        r = (src * dst).sum(dim=-1)
        return r

    def forward(self, x, edge_index, edge_label_index):
        z = self.encode(x, edge_index)
        return self.decode(z, edge_label_index)
```

对于链路预测任务，则在模型中需要将边转化为邻接矩阵，最后通过邻接矩阵的预测得到边的预测

参数分析

自环：在图中给结点添加一条自己到自己的边，可以降低卷积后特征值大小，从而缓解梯度爆炸，在本实验中准确率和 auc 提升显著

Pairnorm：为解决过平滑问题而使用的 normalziaition，但本实验图卷积层数少效果更佳，是否使用 pairnorm 并没有什么区别

Dropedge：在链路预测时解决过平滑问题，在使用两层卷积层时同样没有什么效果

层数：两层效果最好