# 实验2图表示学习实验

## 图节点表示实验

### 实验内容

在图节点表示学习的第一部分实验中，我们将进行节点表示学习的整个流程，具体分为以下三步：

* 第一步，我们先加载经典社交网络图Karate Club Network。并探索这个图上的多种统计信息。
* 接着我们尝试将图结构转化成tensor形式，为我们进一步在图上进行机器学习算法做准备。
* 最后，我们将实现首个图算法：一个图节点表示模型。

### 实验过程

#### Question1:karate club network的average degree(平均度)是多少？

由于karate club network是无向图，所以每一条边能贡献的度数为2，由此可知，总的度数为总边数的两倍，然后再除以节点个数，最后对结果四舍五入到整数得到答案。

代码如下：

##### 

该karate club network的平均度如下：

##### 

#### Question2:karate club网络的平均聚类系数是多少k?

这里直接调用自带的average\_clustering函数进行计算，然后对结果保留两位小数，可以得到答案。

代码如下：

##### 

平均聚类系数如下：

##### 

#### Question3:在一次PageRank迭代之后，节点0(id为0的节点)的PageRank值是多少?

PageRank使用网络的链接结构来衡量图中节点的重要性。来自重要页面的“vote”更有价值。

PageRank算法输出一个概率分布，表示随机浏览者点击链接到达任何特定页面的可能性。在每个时间步，随机的冲浪者有两个选择：

* 的概率随机跟随一个页面；
* 的概率随机跳转到一个页面。

因此，一个特定页面的重要性由下面的PageRank等式计算：

根据上述公式，我们可以得到如下代码：

##### 

首先，算法计算了一个常数(1-beta)/G.number\_of\_nodes()，其中beta是阻尼系数，G.number\_of\_nodes()是图中节点的数量。这个常数表示了随机跳转到任意节点的概率。

然后，算法遍历节点node\_id的所有邻居节点，对于每个邻居节点n，计算了beta\*r0/G.degree(n)，其中beta是阻尼系数，r0是初始PageRank值，G.degree(n)是节点n的度数。这一部分表示了从邻居节点跳转到当前节点的概率。

最后，将这两部分的结果相加得到节点的一次迭代PageRank值r1。

PageRank第一次迭代结果如下：

##### 

#### Question4:获取karate club network的边列表并将其转换为 torch.LongTensor。pos\_edge\_index tensor的 torch.sum 值是多少?

首先获取G的所有边，并将其转化为list，如下图所示：

##### 

接着将边列表转化为torch.LongTensor类型，，最后计算edge的shape和sun值，如下图所示。

##### 

结果如下所示：

##### 

#### Question5:请实现以下对负边进行采样的函数。然后回答哪些边(edge\_1到edge\_5)是karate club network中的负边?

两两配对G中的节点，由这两个节点构成的边如果不在pos\_edge\_list中，则为负边，接着对负边列表进行采样，得到结果。

代码如下：

##### 

判断负边结果如下：

##### 

#### Question6:训练表示：你能得到的最好表示是什么?请在实验报告上记录最佳损失和准确性。

在训练时，首先将需要训练的边进行embedding，然后将两个节点的embedding结果进行点乘，经过sigmoid得到预测结果，将预测结果和真实结果计算loss和accuracy，然后反向传播进行训练。

训练代码如下：

##### 

计算准确率代码如下：

##### 

部分训练结果如下：

##### 

## GNN

### 实验内容

在图节点表示学习的第二部分实验中，我们将使用PyTorchGeometric(PyG)构建我们自己的图神经网络，然后将该模型应用于两个OpengraphBenchmark(OGB)数据集。这两个数据集将用于在两个不同的基于图的任务上对模型的性能进行基准测试:

1. 节点属性预测，预测单个节点的属性;
2. 图属性预测，预测整个图或子图的属性。

以下是实验步骤：

* 首先，我们将学习PyTorchGeometric如何将图存储为PyTorch张量。
* 然后，我们将使用ogb包加载和检查开放图形基准(OGB)数据集之一。OGB是一个现实的、大规模的、不同的基准数据集的集合，用于图上的机器学习。ogb包不仅为每个数据集提供数据加载器，还提供模型评估器。
* 最后，我们将使用PyTorchGeometric构建我们自己的图神经网络。然后，我们将在OGB节点属性预测和图属性预测任务上训练和评估我们的模型。

### 实验过程

#### Question1:ENZYMES数据集中的类和特征的数量是多少?

对于类的数量，直接将数据集中的num\_classes赋值给num\_classes；

对于特征的数量，直接将数据集中的num\_features赋值给num\_features。

具体代码如下：

##### 

##### 

结果如下：

##### 

#### Question2:在ENZYMES数据集中索引为100的图的标签是什么?

这里比较简单，直接找到idx索引下的样本，然后取出它的标签即可。

代码如下：

##### 

结果如下：

##### 

#### Question3:索引为200的图有多少条边?

这里先找到idx索引下的样本并返回它的所有边索引，如果这个图是无向图，则两条相反的边只算一条边，所以需要去掉一些边后再计算该图具体有多少条边即可。

代码如下：

##### 

结果如下：

##### 

#### Question4:在ogbn-arxiv图中有多少特征?

和question1类似，直接返回该图的特征数即可。

代码如下：

##### 

结果如下：

##### 

#### Question5:你的best\_model验证和测试精度是多少?

这一部分我们将使用PyTorchGeometric构建我们的第一个图神经网络。然后我们将其应用于节点属性预测(节点分类)任务。

首先构建模型，这里我们将自己构建GCN模型，具体结构如下：

##### 

GCN模型的每一层由以下几个部分构成：

* GCNConv层，也就是图卷积层，用于从图的结构中提取节点的特征信息。GCNConv层使用输入特征x和图的邻接矩阵adj\_t来计算节点的新的特征表示；
* BatchNorm1d层，也就是批量归一化层，用于对每个节点的特征进行归一化；
* ReLU 激活函数；
* Dropout 层（在训练模式下），用于防止过拟合。它随机将一部分节点的特征值置为零，这样可以迫使模型更好地泛化。

最后一层经过一个图卷积层后进行softmax得到预测分类结果。

具体代码如下：

\_\_init\_\_函数：

##### 

forward函数：

##### 

以下是我们的实验设置：

* Dataset：使用ogbn-arxiv数据集，并将数据集中的邻接矩阵转化为对称矩阵；
* 模型层数：设置为3；
* 隐藏层维数：设置为256；
* Dropout参数：设置为0.5；
* 学习率：设置为0.01；
* Epoch数：设置为100；

接下来就可以开始训练了，训练时的每一步都需要计算loss并反向传播更新参数，还要进行测试，具体代码如下：

train函数：

##### 

test函数：

##### 

运行结果如下：

##### 

best\_model验证和测试精度如下：

##### 

#### Question 6: 你的best\_model验证和测试ROC-AUC分数是多少?

在这一部分，我们将使用GCN创建一个用于图属性预测(图分类)的图神经网络。

首先构建模型，这里使用上面已经构建好的GCN模型，并进行进一步处理。具体结构如下：

* AtomEncoder 层，将原子特征编码为隐藏维度的特征向量。
* GCN 层，图卷积网络（GCN）用于从图的结构中提取节点的特征信息。
* Global Mean Pooling 层，对每个图中的节点特征进行平均池化，以获得图级别的特征表示。
* Linear 层，对每个图的特征表示进行线性变换，以预测图的属性。

具体代码如下：

\_\_init\_\_函数：

##### 

forward函数：

##### 

以下是我们的实验设置：

* Dataset：使用ogbg-molhiv数据集；
* 模型层数：设置为5；
* 隐藏层维数：设置为256；
* Dropout参数：设置为0.5；
* 学习率：设置为0.001；
* Epoch数：设置为30；

接下来就可以开始训练了，训练时的每一步都需要计算loss并反向传播更新参数，还要进行测试，具体代码如下：

train函数：

##### 

运行结果如下：

##### 

best\_model验证和测试ROC-AUC分数如下：

##### 

#### Question 7: 在Pytorch Geometric中使用另外两个全局池化层进行实验。

这里直接更改模型的pool层即可，分别改为global\_max\_pool和global\_add\_pool进行实验。

global\_max\_pool具体代码如下：

##### 

训练结果如下：

##### 

best\_model验证和测试ROC-AUC分数如下：

##### 

global\_add\_pool具体代码如下：

##### 

训练结果如下：

##### 

best\_model验证和测试ROC-AUC分数如下：

##### 