中国科学技术大学

University of Science and Technology of China

深度学习导论

实验 4

姓名 徐海阳　　学号 PB20000326　　院系 少年班学院

摘　要

使用 PyTorch 神经网络库编写图卷积神经网络模型 GCN[1]，并在相应的图结构数据集 Cora, Citeseer, PPI 上完成节点分类和链路预测任务，最后分析自环、层数、DropEdge 、PairNorm 、激活函数等因素对模型的分类和预测性能的影响。

GCN 模型在绝大多数任务上表现都很好，但是在 PPI 数据集上做节点分类任务时表现不佳。因此，本文分析了不同数据集的特性，并且针对 PPI 数据集特性，我们选择将 GCNConv 换成 SAGEConv[2]（SAmple and aggreGatE Convolutional）。使用 SAGEConv 后，我们大幅提升了 PPI 验证集上节点分类的准确率（23.4% -> 54.3%）。

目录

1 数据集与数据预处理

1.1 数据集简介

Cora 是由 2708 篇机器学习论文作为节点、论文间引用关系作为有向边构成的图数据集。这个数据集由来自 Cornell、Wisconsin 和 Texas 大学的研究者在 2003 年创建，主要用于比较不同的图机器学习算法。Cora 数据集中的图表示了一组博士论文，每个节点表示一个论文，每个边表示两个论文之间的引用关系。论文被分成 7 个类别，包括计算机科学、数学、物理学、像素、数据库、人工智能和交叉领域。节点的特征由词袋模型生成，它们是论文标题和摘要中所有单词的计数。

Citeseer 是一个常用的图机器学习数据集，与 Cora 数据集类似，用于研究节点分类问题。这个数据集也包含了一组被引用的科学论文，每个节点表示一篇论文，每个边表示两个论文之间的引用关系。与 Cora 不同的是，Citeseer 数据集的论文旨在涵盖计算科学的更广泛领域，并将它们划分为 6 个类别，包括人工智能、数据库、机器学习、信息检索、多媒体和分布式系统。Citeseer 数据集中的节点特征包括标题、摘要和关键字，由于篇幅限制，这些特征的数量被手工选择为 3703 个。这些特征是标准的词袋向量，表示了论文中出现的每个单词的出现次数。Citeseer 的特征维度比 Cora 高，因此相比 Cora 难度更大。

PPI 是一个大规模的图机器学习数据集，与 Cora 和 Citeseer 不同，它是一个用于蛋白质相互作用预测的数据集。相比于传统的图数据集，PPI 具有更高的维度和更复杂的图结构，因此对于图神经网络的设计和训练提出了更高的要求。它包含了一组人类体内的蛋白质及其相互作用网络。每个节点表示一个蛋白质，每个边表示两个蛋白质之间的相互作用关系。整个数据集包含了 24,043 个蛋白质和 37,332 个相互作用关系，并被划分为 3 个类别。PPI 的难度比 Cora 和 Citeseer 都要大很多，这是因为以下三个原因：（1）从蛋白质和相互作用关系之间的数量比例可以看出，PPI 是一个非常稀疏的图数据集，平均每个蛋白质只有 1.5 个相互作用关系；（2）PPI 数据集中的节点特征和标签是多维矩阵，包括蛋白质的氨基酸序列、结构和生化特征，以及蛋白质相互作用的标签，这些特征和标签的维度非常高；（3）蛋白质表达可能受到其他非邻接节点的影响：两种蛋白质无法直接相互作用（非邻接节点），但它们可以间接相互作用，而这在图中是没有边相连来表达这个关系的。

三个数据集的统计数据如下：

	节点数	特征数	训练集	验证集	测试集
Cora	2,708	1,433	7	140	1,000
Citeseer	3,312	3,703	6	120	1,000
PPI	24,043	50-3,634	3	11,000	3,000

表 1: Cora、Citeseer 和 PPI 的统计数据

1.2 数据预处理

参考助教给的代码，数据预处理的主要逻辑如下图1所示。首先得到论文单词组成，也就是节点特征；接着得到论文的类别，也就是节点标签；然后提取边；最后进行节点分类任务和链路预测任务的数据集划分和图数据封装。

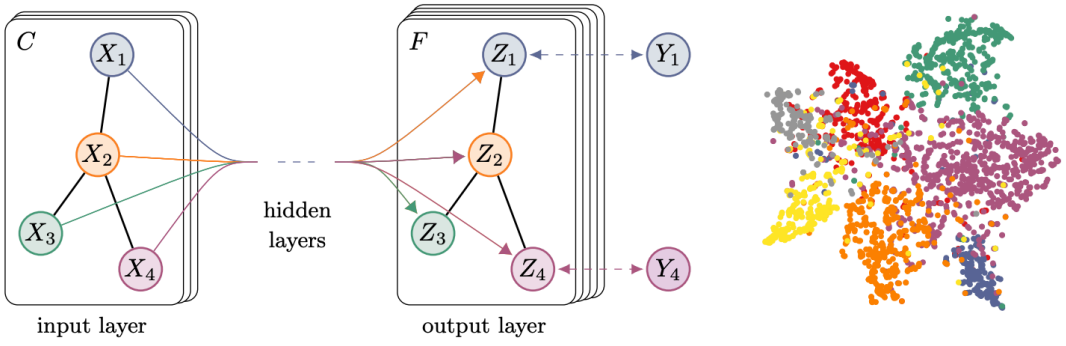
```python
def load_cite_data(path="../data/cora/", dataset="cora", task='node'):
    idx_features_labels = np.genfromtxt("{}{}.content".format(path, dataset),
                                        dtype=np.dtype(str))
    # 得到论文单词组成，也就是节点特征
    x = np.array(idx_features_labels[:, 1:-1], dtype=np.float32)
    # 得到论文的类别，也就是节点标签
    y = encode_label(idx_features_labels[:, -1])
    num_classes = torch.tensor(np.max(y) + 1)
    # 提取边
    edges_unordered = np.genfromtxt("{}{}.cites".format(path, dataset),dtype=np.dtype(str))
    edge_index = [[], []]
    for i in range(edges_unordered.shape[0]):
        ...
    edge_index = np.array(edge_index, dtype=np.int32)
    if task == 'node':  # 节点分类任务
        # 随机划分训练集、验证集和测试集
        ...
        # 把数据转换为合适的类型，并且封装到图数据的类型
        ...
    else:  # 链路预测任务
        # 随机划分边为训练集，验证集和测试集。训练集无负样本。图为有向图。
        ...
```

图 1: 数据预处理逻辑图

2 图神经网络模型

2.1 GCN (Graph Convolutional Networks)

GCN[1] 的主要组成部分即是 GCNConv，然后通过堆叠 GCNConv 和非线性层得到整个网络。GCN 网络图如下图2 所示。



(a) Graph Convolutional Network (b) Hidden layer activations

Figure 1: *Left*: Schematic depiction of multi-layer Graph Convolutional Network (GCN) for semi-supervised learning with C input channels and F feature maps in the output layer. The graph structure (edges shown as black lines) is shared over layers, labels are denoted by Y_i. *Right*: t-SNE (Maaten & Hinton, 2008) visualization of hidden layer activations of a two-layer GCN trained on the Cora dataset (Sen et al., 2008) using 5% of labels. Colors denote document class.

图 2: GCN 网络图

GCNConv 的计算过程如下

$$H = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW\right)$$

其中，$\tilde{A} = A + I_n$ 表示邻接矩阵 A 加上自环，I_n 为 $n \times n$ 的单位矩阵；$\tilde{D}$ 为度矩阵，定义为

$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, 即每个节点的度数之和 (包括自环) ; σ 表示激活函数, 通常使用 ReLU 函数。

上式中的矩阵乘法可以理解为以下几个步骤:

1. 对节点特征矩阵 X 进行线性变换: XW, 得到 $n \times h$ 的中间特征矩阵 XW 。
2. 对邻接矩阵 $\tilde{A}$ 进行归一化处理: $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, 得到 $n \times n$ 的归一化邻接矩阵 $\tilde{A}_{\text{norm}}$
3. 对中间特征矩阵 XW 和归一化邻接矩阵 $\tilde{A}_{norm}$ 进行矩阵乘法: $\tilde{A}_{\text{norm}}\, XW$, 得到 $n \times h$ 的新特征矩阵 Z 。
4. 对新特征矩阵 Z 进行激活函数操作: $\sigma(Z)$, 得到 $n \times h$ 的最终特征矩阵 H 。

2.2 SAGEConv（SAmple and aggreGatE Convolutional）

SAGEConv 出自 GraphSAGE[2] 这篇论文，它相比 GCNConv 的改进之处是通过采样，将节点的自身特征与它的邻居节点的特征信息进行聚合操作, 从而得到节点的新特征表示。这种聚合操作可以看作是一种信息传递过程, 通过邻居节点的特征信息和自身特征来更新当前节点的特征信息。SAGEConv 图如下图3 所示。

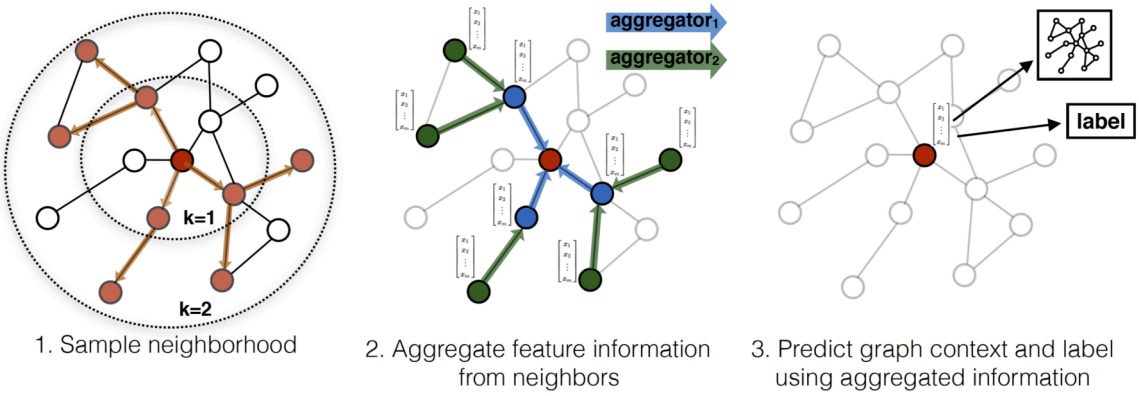


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

图 3: SAGEConv 图

SAGEConv 的计算过程可以分为以下几个步骤:

1. 将图数据表示为邻接矩阵 A 和节点特征矩阵 X, 其中 A_{ij} 表示节点 i 和节点 j 之间是否存在边, X_i 表示节点 i 的特征向量。
2. 聚合操作: 对于一个节点 i, 将一定距离内（图的距离即为两个节点间最短路径的长度）的节点设置为邻居节点。
3. 聚合操作: 对于一个节点 i, 将它的自身特征向量 X_i 和采样的邻居节点特征向量 X_j 进行聚合操作, 得到邻居节点的聚合特征向量 agg_i。聚合操作可以有多种实现方式, 包括 mean、max 等。
4. 对所有邻居节点的聚合特征向量求和并除以邻居节点数, 得到节点 i 的新特征向量 H_i :

$$H_i = \sigma\left(W \cdot \text{concat}\left(X_i, agg_i\right)\right)$$

其中, σ 表示激活函数, W 表示卷积核参数, concat 表示向量拼接操作。

3 任务

3.1 节点分类任务

目标是将节点根据其特征和网络结构分类到相应的类别中，如 Cora、Citeseer 数据集中每篇论文属于什么领域，PPI 数据集中每种蛋白质属于什么类别。

3.2 链路预测任务

该任务旨在预测两节点间之间是否存在边，如 Cora、Citeseer 数据集中两篇论文是否存在引用关系，PPI 数据集中两种蛋白质是否存在相互作用。

4 实验结果与分析

在充分训练 1000 个 epochs 的情况下（early stop 的 patience 设置为 500 epochs），最优的图网络设置（自环、层数、DropEdge、PairNorm 等因素）以及测试集结果如下表2 所示：

数据集	任务	层数	自环	DropEdge	PairNorm	Val Acc@1(%) ↑	Test Acc@1(%) ↑
Cora	节点分类	2	✓	✗	✗	84.9	88.2
Citeseer	节点分类	1	✓	✓	✗	74.8	73.3
PPI	节点分类	2	✓	✗	✗	54.3	55.2
Cora	链路预测	1	✓	✗	✗	91.6	88.6
Citeseer	链路预测	1	✓	✗	✗	92.4	93.2
PPI	链路预测	1	✓	✗	✗	75.7	75.5

表 2: 最优图网络设置及测试集结果

从表中可以看出，所有任务、所有数据集都要求自环为 True，这说明自身节点信息的增加对于节点间信息的相互作用非常有利；同时可以看到 Citeseer 的节点分类任务要求 DropEdge 为 True，这说明在稠密图中适当地 dropout 可以抑制过拟合提升模型性能；在调参过程中发现其实有没有 PairNorm 对于结果的影响不大，False 的时候结果稍好一些；调参得到最优学习率为 0.1，特殊情况是在 PPI 上做节点分类任务时的学习率需要调高至 0.2，否则收敛不充分。

特别地，由于直接在 PPI 数据集的节点分类任务中直接使用 GCNConv 效果很差 (23.4%)，因此我们使用 SAGEConv。本文认为这是数据集特征决定的。Cora 和 Citeseer 数据集都是论文引用数据集，高密度的互相引用以及引用之间的直接联系决定了用简单的 GCN 模型即可获得很好的性能；而对于 PPI 数据集来说，它是蛋白质相互作用数据集，这种数据集拥有以下两个特点（1）低密度性，即每种蛋白质通常只与少数几种蛋白质相互作用；（2）非邻接性，即节点对应的蛋白质表达可能受到其他非邻接节点的影响：两种蛋白质无法直接相互作用（非邻接节点），但它们可以间接相互作用。因此，针对 PPI 数据集的这两种特性，我们选择将 GCNConv 换成 SAGEConv（SAmple and aggreGatE Convolutional）[2]。SAGEConv 在每个节点周围采样一定数量的邻居节点（注意不是邻接，即使不是邻接节点，只要是在采样距离内的邻居节点即可），然后聚合该节点和邻居节点的特征，并使用线性变换对其进行编码。经过多次这样的“聚合-采样”过程后，SAGEConv 可以更好地处理低密度图，提升每个节点的全局信息表达能力。实验结果也验证了我们的分析，将 GCNConv 换为 SAGEConv 后，验证集准确率直接从 23.4% 上升到 54.3%。

三个数据集在两种任务上的训练集和验证集损失图如4, 5, 6, 7, 8, 9 所示。从图8中也可以看出 SAGEConv 的收敛非常快，说明” 采样-聚合” 对于全局信息提取能力很强，能够让网络快速收敛。

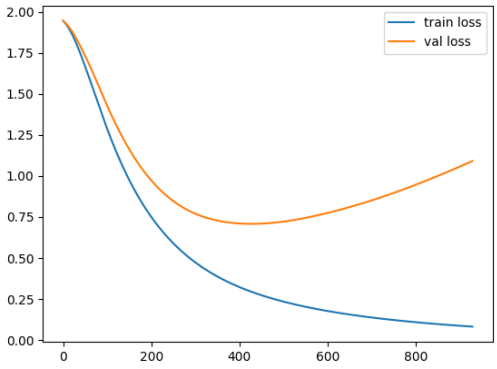


图 4: Cora 节点分类 Train/Val Loss

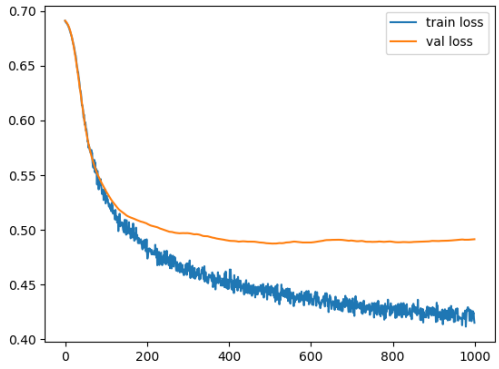


图 5: Cora 链路预测 Train/Val Loss

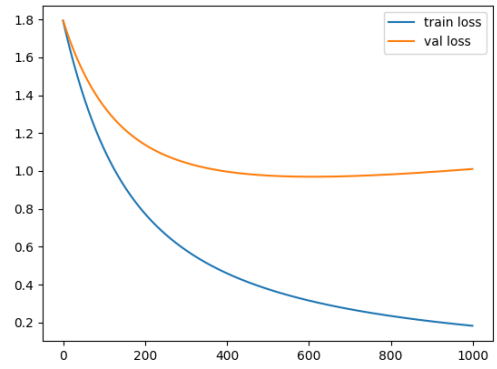


图 6: Citeseer 节点分类 Train/Val Loss

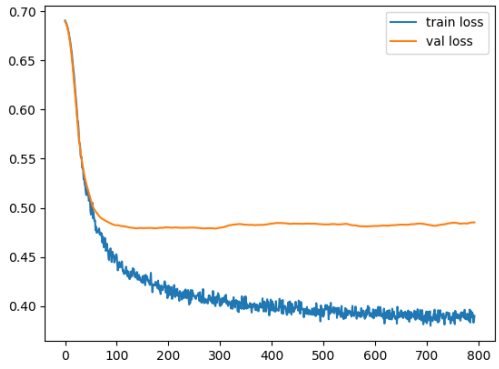


图 7: Citeseer 链路预测 Train/Val Loss

参考文献

[1] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[2] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.

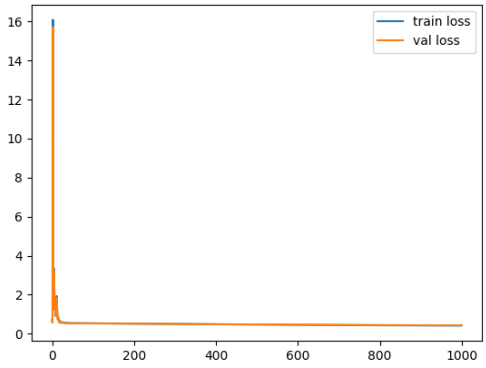


图 8: PPI 节点分类 Train/Val Loss

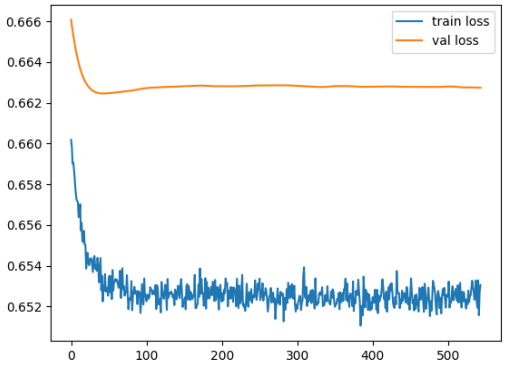


图 9: PPI 链路预测 Train/Val Loss