天津大学本科生实验报告专用纸

学院 智能与计算学部 年级 2019 专业 计算机科学与技术

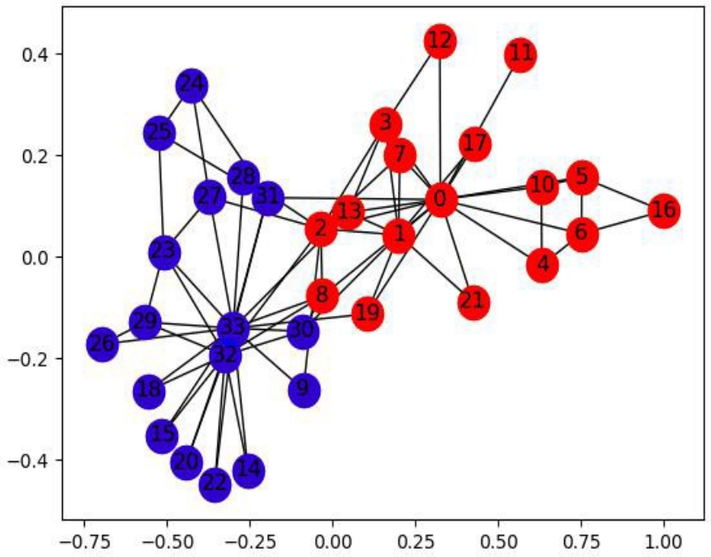
班级 2 班 姓名 张明君 学号 6319000359

课程名称 人工智能基础 实验日期 2021.06.15 成绩

实验项目名称：使用 GCN 解决空手道俱乐部

1. 实验内容

空手道俱乐部是一个包含 34 个成员的社交网络，有成对的交互发生在成员之间。 俱乐部后来分裂成两个群体，分别以指导员（节点 0）和俱乐部主席（节点 33） 为首，任务：使用 GCN 预测每个节点会加入哪一边，节点 0 或节点 33？ 整个网络可视化如下图：

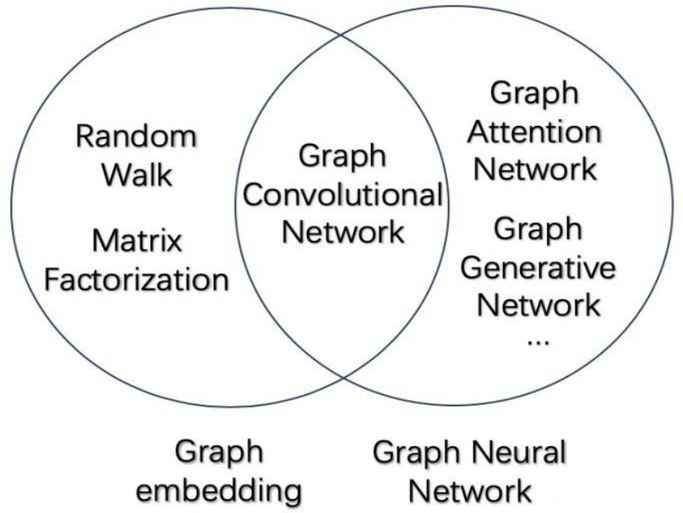


Karate 空手道俱乐部复杂网络是常用于复杂网络社区发现研究的网络，该网络共有 34 个节点和 78 条边，其中 34 个节点表示某空手道俱乐部的 34 名成员，节点之间的边表示两个成员相互认识，该数据集是一个真实的数据集，其对应于美国的一个空手道俱乐部的人物关系的研究。针对于复杂网络中的社区发现研究有着非同寻常的意义。

天津大学本科生实验报告专用纸

2. 实验原理与步骤

GCN (Graph Convolutional Network) 是一类采用图卷积的神经网络，发展到现在已经有基于最简单的图卷积改进的无数版本，在图网络领域的地位正如同卷积操作在图像处理里的地位。



如上图所示，图卷积神经网络 GCN 属于图神经网络 GNN 的一类，是采用卷积操作的图神经网络，可以应用于图嵌入 GE。

假设有一批图数据，其中有 N 个节点（node），每个节点都有自己的特征，设这些节点的特征组成一个 N×D 维的矩阵 X，然后各个节点之间的关系也会形成一个 N×N 维的矩阵 A，也称为邻接矩阵（adjacency matrix）。X 和 A 便是我们模型的输入。

GCN 也是一个神经网络层，它的层与层之间的传播方式是：

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{H}^{(l)}\mathbf{W}_0^{(l)} + \tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}_1^{(l)}\right)$$

with $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$

这个公式中：

- A 波浪=A+I，I 是单位矩阵
- D 波浪是 A 波浪的度矩阵（degree matrix），公式为
- H 是每一层的特征，对于输入层的话，H 就是 X
- σ 是非线性激活函数

这个部分，是可以事先算好的，因为 D 波浪由 A 计算而来，而 A 是我们的输入之一。

GCN 是一类非常强大的用于图数据的神经网络架构。现在我们使用 GCN 来实现一个简单的例子就是用来解决空手道俱乐部问题。现在我们来看他的过程和步骤。

1）建立图

创建 karate club 图如下：

```
import dgl

import matplotlib

import torch

import networkx as nx

def build_karate_club_graph():

    g = dgl.DGLGraph()

    g.add_nodes(34)

    edge_list = [(1, 0), (2, 0), (2, 1), (3, 0), (3, 1), (3, 2),

        (4, 0), (5, 0), (6, 0), (6, 4), (6, 5), (7, 0), (7, 1),

        (7, 2), (7, 3), (8, 0), (8, 2), (9, 2), (10, 0), (10, 4),

        (10, 5), (11, 0), (12, 0), (12, 3), (13, 0), (13, 1), (13, 2),

        (13, 3), (16, 5), (16, 6), (17, 0), (17, 1), (19, 0), (19, 1),

        (21, 0), (21, 1), (25, 23), (25, 24), (27, 2), (27, 23),

        (27, 24), (28, 2), (29, 23), (29, 26), (30, 1), (30, 8),

        (31, 0), (31, 24), (31, 25), (31, 28), (32, 2), (32, 8),

        (32, 14), (32, 15), (32, 18), (32, 20), (32, 22), (32, 23),

        (32, 29), (32, 30), (32, 31), (33, 8), (33, 9), (33, 13),

        (33, 14), (33, 15), (33, 18), (33, 19), (33, 20), (33, 22),

        (33, 23), (33, 26), (33, 27), (33, 28), (33, 29), (33, 30),

        (33, 31), (33, 32)]

    src, dst = tuple(zip(*edge_list))

    g.add_edges(src, dst)

    g.add_edges(dst, src)

    return g
```

输出创建的节点和边的数量：

```
G = build_karate_club_graph()

print('We have %d nodes.' % G.number_of_nodes())

print('We have %d edges.' % G.number_of_edges())
```

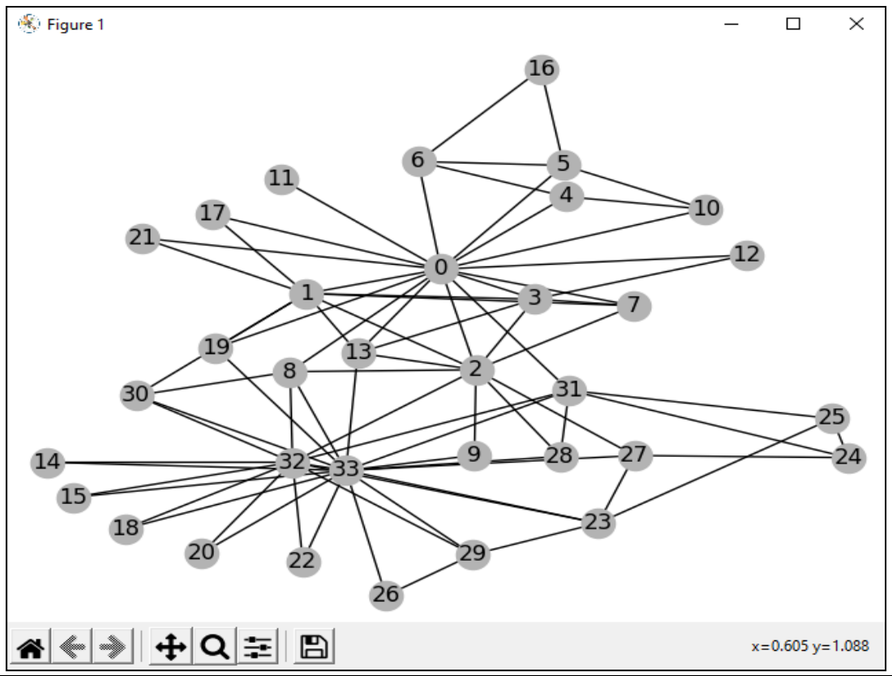
输出结果：

```
We have 34 nodes.
We have 156 edges.
```

利用 qhwz run{ 画 judsk

```
import networkx as nx

import matplotlib.pyplot as plt

fig = plt.figure(dpi=150)

nx_G = G.to_networkx().to_undirected()

pos = nx.kamada_kawai_layout(nx_G)

nx.draw(nx_G, pos, with_labels=True, node_color=[[.7, .7, .7]])

plt.show()
```

输出结果：



2) 给边和节点赋予特征

GCN 将特征与节点和边关联起来进行训练。对于我们的分类示例，我们为每个节点分配一个输入特征作为一个热向量：节点 vivi’s 的特征向量是[0，…，1，…，0][0，…，1，…，0]，其中第 i 个位置是 1。在 DGL 中，可以使用沿第一个维度批处理节点特征的特征张量，一次为所有节点添加特征。下面的代码为所有节点添加了一个热特性：

联合边和节点信息做图训练。对于整个节点分类的例子，将每个节点的特征转化成 one-hot 向量：节点变为[0,⋯,1,⋯,0][0,⋯,1,⋯,0]，对应的位置上的数值为 1。在 DGL 里面，可以使用一个 feature 张量在第一维上一次性给所有的节点添加特征，代码如下:

```
import torch

G.ndata['feat'] = torch.eye(34)

print(G.nodes[2].data['feat'])

print(G.nodes[[10, 11]].data['feat'])
```

输出结果：

```
tensor([[0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
tensor([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.,
         0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

3）定义一个 GCN

上面整个步骤可以看作一个 message-passing 的范式：每个节点会接受邻居节点的信息从而更新自身的节点表示。一个图形化的例子就是：向目标节点流动

接下来，是一个使用 GDL 实现 GCN 的例子：

```
import torch.nn as nn
import torch.nn.functional as F

    def gcn_message(edges):

      return {'msg' : edges.src['h']}

    def gcn_reduce(nodes):

      return {'h' : torch.sum(nodes.mailbox['msg'], dim=1)}
```

```
class GCNLayer(nn.Module):

    def __init__(self, in_feats, out_feats):

        super(GCNLayer, self).__init__()

        self.linear = nn.Linear(in_feats, out_feats)

    def forward(self, g, inputs):

        g.ndata['h'] = inputs

        g.send_and_recv(g.edges(),gcn_message,gcn_reduce)

        h = g.ndata.pop('h')

        return self.linear(h)

class GCN(nn.Module):

    def __init__(self, in_feats, hidden_size, num_classes):

        super(GCN, self).__init__()

        self.gcn1 = GCNLayer(in_feats, hidden_size)

        self.gcn2 = GCNLayer(hidden_size, num_classes)


    def forward(self, g, inputs):

        h = self.gcn1(g, inputs)

        h = torch.relu(h)

        h = self.gcn2(g, h)

        return h
```

4）数据准备和初始化

我们使用 one-hot 向量初始化节点。因为是一个半监督的设定，仅有指导员（节点 0）和俱乐部主席（节点 33）被分配了 label，实现如下：

```
net = GCN(34,5,2)

inputs = torch.eye(34)

labeled_nodes = torch.tensor([0, 33])

labels = torch.tensor([0, 1])
```

5) 训练和可视化

训练的步骤和 PyTorch 模型一样，（1）创建优化器，（2）输入 input 数据，（3）计算 loss，（4）使用反向传播优化模型

```
optimizer = torch.optim.Adam(net.parameters(), lr=0.01)

all_logits = []

import matplotlib.animation as animation

import matplotlib.pyplot as plt

nx_G = G.to_networkx().to_undirected()

pos = nx.kamada_kawai_layout(nx_G)


for epoch in range(40):

    logits = net(G, inputs)

    all_logits.append(logits.detach())

    logp = F.log_softmax(logits, 1)

    loss = F.nll_loss(logp[labeled_nodes], labels)

    optimizer.zero_grad()

    loss.backward()

    optimizer.step()

    print('Epoch %d | Loss: %.4f' % (epoch, loss.item()))
```

输出结果：

```
Epoch 0 | Loss: 1.1879
Epoch 1 | Loss: 0.8704
Epoch 2 | Loss: 0.6550
Epoch 3 | Loss: 0.5078
Epoch 4 | Loss: 0.4035
Epoch 5 | Loss: 0.3176
Epoch 6 | Loss: 0.2451
Epoch 7 | Loss: 0.1889
Epoch 8 | Loss: 0.1456
Epoch 9 | Loss: 0.1112
Epoch 10 | Loss: 0.0833
Epoch 11 | Loss: 0.0611
Epoch 12 | Loss: 0.0439
Epoch 13 | Loss: 0.0307
Epoch 14 | Loss: 0.0214
Epoch 15 | Loss: 0.0149
Epoch 16 | Loss: 0.0105
Epoch 17 | Loss: 0.0074
Epoch 18 | Loss: 0.0053
Epoch 19 | Loss: 0.0038
Epoch 20 | Loss: 0.0028
Epoch 21 | Loss: 0.0021
Epoch 22 | Loss: 0.0016
Epoch 23 | Loss: 0.0012
Epoch 24 | Loss: 0.0010
Epoch 25 | Loss: 0.0008
Epoch 26 | Loss: 0.0006
Epoch 27 | Loss: 0.0005
Epoch 28 | Loss: 0.0004
Epoch 29 | Loss: 0.0004
Epoch 30 | Loss: 0.0003
Epoch 31 | Loss: 0.0003
Epoch 32 | Loss: 0.0002
Epoch 33 | Loss: 0.0002
Epoch 34 | Loss: 0.0002
Epoch 35 | Loss: 0.0002
Epoch 36 | Loss: 0.0002
Epoch 37 | Loss: 0.0001
Epoch 38 | Loss: 0.0001
Epoch 39 | Loss: 0.0001
>>>
```

这是一个非常简单的小例子，甚至没有划分验证集和测试集。因此，因为模型最后输出了每个节点的二维向量，我们可以轻易的在 2D 的空间将这个过程可视化出来，下面的代码动态的展示了训练过程中从开始的状态到到最后所有节点都线性可分的过程。

```
import matplotlib.animation as animation

import matplotlib.pyplot as plt


def draw(i):

    cls1color = '#00FFFF'

    cls2color = '#FF00FF'

    pos = {}

    colors = []

    for v in range(34):

        pos[v] = all_logits[i][v].numpy()

        cls = pos[v].argmax()

        colors.append(cls1color if cls else cls2color)

    ax.cla()

    ax.axis('off')

    ax.set_title('Epoch: %d' % i)

    nx.draw_networkx(nx_G.to_undirected(), pos, node_color=colors,

            with_labels=True, node_size=200)


fig = plt.figure(dpi=150)

fig.clf()

ax = fig.subplots()

draw(0)  # draw the prediction of the first epoch

plt.close()
```

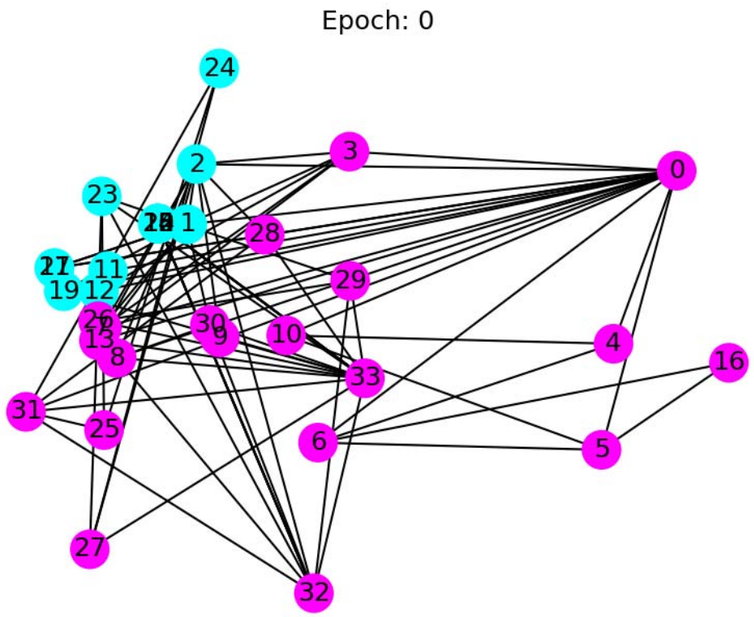
下面的动态过程展示了模型经过一段训练之后能够准确预测节点属于哪个群组。

```
ani = animation.FuncAnimation(fig, draw, frames=len(all_logits), interval=200)
```

2. 实验结果与分析

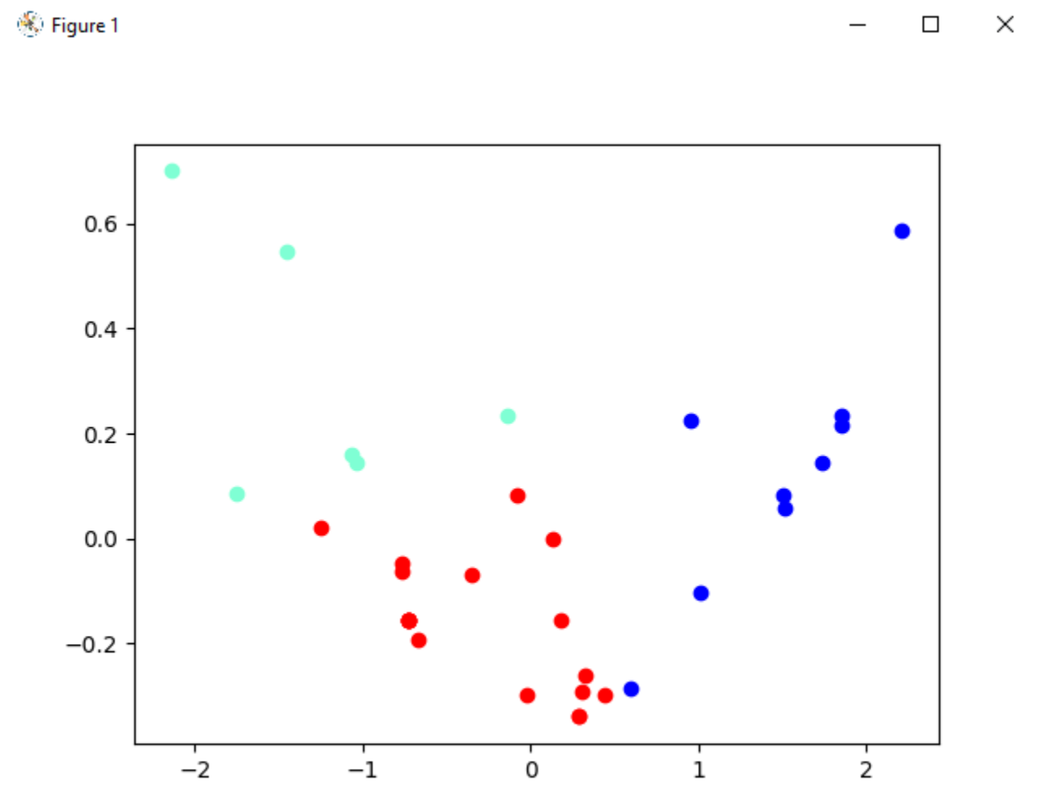
本次实验我用 Python ILDE 来做实验的，使用 Python 语言实现算法。经过很多次程序调试，最后就得了以下结果。到这里我做了两次结果就是将整个网络节点分为 2 类像老师给的，然后另一个就是将整个网络节点分为 3 类。完整代码我就跟这个报告一起交的。

- 这就是将整个网络节点分为 2 类的结果如下：



这就是它的图片，因为这个代码有 animate 的所以它就有动作。可是在这里我不可以放 GIF 的文件在里面所以提交的时候我就跟这个报告一起交的。最后就是这文件的名字就是 final.py 然后他的 animate 就是 final.gif

- 这就是将整个网络节点分为 3 类的结果如下：



这个就是将整个网络节点分为 3 类，然后它的代码我跟报告一起交的名称 final1.py。这个网络的三类也跟 2 类没什么大区别我们只要换一些代码然后变成它为 2D 的图片就行了。

3. 实验总结

通过以上的实验过程中，终于我们想要的结果了，虽然有点难但是能得了很多知识关于 GCN 的用法和算法操作。轮到这里，空手道俱乐部的实现也不太容易的，空手道俱乐部数据集作为一个真实环境下的数据集，对于研究社区发现十分重要。通过一些社区划分方法将该俱乐部进行划分，可以知道该网络有几个社团，每个社团包含的成员。这些信息对于了解整个网络的分布和内部情况具有重要意义。虽然在这次实验过程当中遇到不少问题，但是通过自己和同学的帮助，最后都能将这些问题解决了。

教师签字：

2021 年　月　日