Social net Lab

A report

Using GraphSAGE

By

Omar Hossam Ahmed

2205150

So lets begin by explaining the code ::

```
!pip install torch_geometric
```

- First thing is first , install the library that we will use for GNN.

```
import torch
from torch_geometric.data import Data
from torch_geometric.nn import SAGEConv
import torch.nn.functional as F
```

- Then , we import the lib's in our project
- Torch lib : main PyTorch library'
- Data : Stores the graph (node, edges, labels, etc..)
- SAGEconv : GraphSAGE convoltion layer
- F : contain the activation functions and losses

```
x = torch.tensor([
    [1.0, 0.0],  # benign
    [1.0, 0.0],
    [1.0, 0.0],
    [0.0, 1.0],  # malicious
    [0.0, 1.0],
    [0.0, 1.0]
], dtype=torch.float)
```

- Then, we create the Graph nodes
- 6 nodes , 2 features.
- Valid (non-malicious\benign): [1,0]
- Mal: [0,1]

```
edge_index = torch.tensor([
    [0,1], [1,0],  # 0—1
    [1,2], [2,1],  # 1—2
    [0,2], [2,0],  # 0—2  (benign triangle fully connected)
    [3,4], [4,3],
    [4,5], [5,4],
    [3,5], [5,3],  # malicious triangle fully connected
    [2,3], [3,2]   # one cross edge between communities
]).t().contiguous()
```

- Now , we create edges
- GraphSAGE needs edge_index shape == [2, num_edges]
- We duplicate the edges to represent the graph in undirected shape.
- Benign community : 0,1,2 fully connected
- Malicious comm : 3-4-5 fully connnected
- Only on bridge between 2 groups : 2 – 3

```
y = torch.tensor([0, 0, 0, 1, 1, 1], dtype=torch.long)
```

- Now , labels
- Labels define the true class of each node.
- 0 = benign
- 1 = malicious

```
data = Data(x=x, edge_index=edge_index, y=y)
```

- Creating the PyG Data object
- Graph structure will look like:
- x: node features
- edge_index: graph connections
- y: labels
- Data : what you pass to the GNN model.

```
class GraphSAGENet(torch.nn.Module):

def init(self, in_channels, hidden_channels, out_channels):

super().init()

self.conv1 = SAGEConv(in_channels, hidden_channels)

 self.conv2 = SAGEConv(hidden_channels, out_channels)
```

- Defining the GraphSage neural network
- Here , we create 2 layer GraphSAGE model:
- Conv1 (in: 2, out :4)
- Conv2 (in: 4, out:2)

```
def forward(self, x, edge_index):

  x = self.conv1(x, edge_index)

  x = F.relu(x)  # activation

  x = self.conv2(x, edge_index)

  return F.log_softmax(x, dim=1)
```

- This is the forward pass
- GraphSAGE layers do :"new_representation(node_i) = combine(node_i features + neighbors' features)"
- This allow the model to detect some things ::
- Communities
- Graph structure
- Suspicious cross-connections

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

model.train()

for epoch in range(50):

  optimizer.zero_grad()

  out = model(data.x, data.edge_index)
```

```
loss = F.nll_loss(out, data.y)

loss.backward()

optimizer.step()
```

- Here, we train our GNN model
- Pass the node features and edges to the model
- Model gives output of log probabilities of class 0/1
- Then, it compare predictions with true labels using NLL loss
- Backprop – update weights

```
model.eval()

pred = model(data.x, data.edge_index).argmax(dim=1)

print("Predicted labels:", pred.tolist())
```

- Last , we test the predictions by printing it out
- It gives something like : Predicted labels: [0, 0, 0, 1, 1, 1]
- Which means , the model learned the structure in a perfect way.