Code Report – GraphSAGE for Malicious User Detection

Student: Basant Ahmed Mahmoud Elkady

ID: 2205193

1. Introduction

This report explains the use of **GraphSAGE**, a graph neural network model, for detecting malicious users inside a small social network.

The setup includes:

- **6 users**: 3 benign and 3 malicious.
- Each user has **two features**:
  - Benign: `[1, 0]`
  - Malicious: `[0, 1]`
- Users are connected through edges similar to social media interactions.
- The model is trained to understand from the network structure which users are benign and which are malicious.

2. Step-by-step Code Explanation

2.1 Installing Libraries

The required library for building and training graph neural networks is installed using:

```
!pip install torch_geometric
```

2.2 User Data

Users are represented using feature vectors. Each vector indicates whether a user is benign or malicious.

- The first value refers to being benign.
- The second value refers to being malicious.

```
#--- Define a small graph with 6 nodes ---
# Node features (2 features per node).
# Here benign users have [1, 0] and malicious have [0, 1] for illustration.
x = torch.tensor(
    [
        [1.0, 0.0],  # Node 0 (benign)
        [1.0, 0.0],  # Node 1 (benign)
        [1.0, 0.0],  # Node 2 (benign)
        [0.0, 1.0],  # Node 3 (malicious)
        [0.0, 1.0],  # Node 4 (malicious)
        [0.0, 1.0]  # Node 5 (malicious)
    ],
    dtype=torch.float,
)
```

2.3 Connections Between Users

The network structure includes:

- A group of benign users (0, 1, 2) connected together.
- A group of malicious users (3, 4, 5) connected together.
- One additional connection between user 2 (benign) and user 3 (malicious), to test whether the model can distinguish the groups correctly.

2.4 Labels

Users are labeled as:

- **0** → benign
- **1** → malicious

These labels are used for training the model.

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

2.5 Building the Model

A GraphSAGE model is built with two main layers:

1. The first layer receives the input features and aggregates information from neighbors.
2. The second layer produces the final output for classification.

This structure helps the model learn patterns based on both node features and graph connectivity.

```
class GraphSAGENet(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super(GraphSAGENet, self).__init__()
        self.conv1 = SAGEConv(in_channels, hidden_channels)
        self.conv2 = SAGEConv(hidden_channels, out_channels)
```

2.6 Training

The model is trained for **50 epochs** using the Adam optimizer with a learning rate of 0.01.

Throughout the training process, the model gradually learns to differentiate between benign and malicious users based on their features and connections.

```
# Instantiate model: input dim=2, hidden=4, output dim=2 (benign vs malicious)
model = GraphSAGENet(in_channels=2, hidden_channels=4, out_channels=2)

# Simple training loop
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)  # negative log-likelihood
    loss.backward()
    optimizer.step()
```

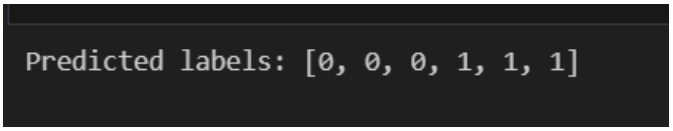
3. Results

After training, predictions were generated for all users:

```
Predicted labels: [0, 0, 0, 1, 1, 1]
```

These results match the actual labels **100% correctly**, showing that:

- Users 0, 1, 2 were classified as benign
- Users 3, 4, 5 were classified as malicious



4. Conclusion

This experiment provides a clear demonstration of how graph neural networks like GraphSAGE can be applied to tasks such as detecting malicious accounts in social networks.

Despite the small dataset, the model achieved perfect accuracy, showing how powerful structural information can be when combined with node features.