Name: Arsany Osama
ID: 2205122

GraphSAGE and Node Classification

In this lab, I tried to understand how GraphSAGE works by experimenting with a very small graph and a simple classification task. The whole point was to see how a Graph Neural Network can decide whether a user looks benign or malicious based on both the user's own features and how they are connected inside the network. Working with such a tiny example helped me see the idea without getting lost in too many details.

1. Understanding the Idea Behind GraphSAGE

GraphSAGE does not treat every node separately. Instead, it learns a general rule that tells it how to combine a node's features with those of its neighbours. This makes it more practical for graphs that never stop changing, like social networks. What stood out to me is that each node ends up with a representation influenced by two things: the features it already has, and the kind of neighbours surrounding it.

From a security point of view, this is useful. Malicious accounts usually behave or connect in ways that do not resemble normal users. When GraphSAGE mixes each node with its neighbours, these differences show up naturally in the learned embeddings.

2. Dataset Used in the Lab

The graph we used was intentionally simple: six nodes in total. Each node had just two numerical features. Benign users were given the vector $[1, 0]$, while malicious ones had $[0, 1]$.

The structure of the graph formed two small clusters. Nodes 0, 1, and 2 (all benign) were fully connected to each other. Nodes 3, 4, and 5 (all malicious) also formed a tight group. There was only one link between the two sides: an edge connecting node 2 to node 3. This small "bridge" made it easier to notice how the neighbourhood influences each node.

Every node had a clear label: 0 for benign, 1 for malicious.

3. Model Architecture

The model consisted of two GraphSAGE layers. The first layer combined each node with its neighbours and passed the result through a ReLU function. The second layer produced the final outputs for the two classes. Even though GraphSAGE usually samples neighbours, in a graph this small the model effectively used all neighbours anyway.

4. Training Process

The model trained for 50 epochs. In each round, it generated predictions, computed its loss using negative log-likelihood, and adjusted its weights with the Adam optimiser. Because the data was very clean and the two groups were well separated, the model converged quickly.

5. Output and Interpretation

At the end, the model predicted the labels:

$$[0, 0, 0, 1, 1, 1]$$

This means that all six nodes were classified correctly. The benign nodes stayed close to each other in the embedding space, and the malicious ones formed their own region. Even the edge connecting node 2 to node 3 did not confuse the network, because each node was still mainly surrounded by neighbours of the same type.

6. Security Insight

Although this is only a toy example, it shows why GraphSAGE is useful for security tasks. Suspicious accounts often form their own patterns of connections or behave differently from ordinary users. When the model learns embeddings based on neighbourhood structure, these differences become easier to spot. A system built on the same idea could help detect fake accounts, coordinated attacks, or abnormal activity by looking at how much a node's behaviour deviates from what the model learned as "normal."