Module 3 Assignment

Course: EAI6020 21652 AI System Technologies
CPS Winter Quarter Term B

Instructor: Siddharth Rout

Prepared By:
Abhilash Dikshit
Murtaza Vora
Gunjan Paladiya

21/03/2023

Introduction:

There are now enormous networks of connected individuals thanks to the emergence of social media platforms and online gaming groups. Of them, Twitch is a well-known venue where players may interact with viewers, exchange stories, and create communities. The Twitch Gamer network dataset offers a wealth of opportunities to investigate the dynamics of user interactions, game preferences, and content streaming behaviours in the context of network analysis and machine learning. With a particular focus on predicting if a user streams mature content, this study attempts to explore the underlying structure of the Twitch Gamer network by utilising graph neural networks (GNNs) and multi-scale attributed node embedding techniques.

The collection consists of nodes that represent individual Twitch gamers and edges that show follower relationships between them. The node features also contain embeddings of the games that Twitch users are playing, which offers insightful information about their gaming habits and interests. This research aims to identify hidden patterns, community structures, and prediction characteristics connected to content streaming preferences on the Twitch platform by utilising the power of GNNs and sophisticated machine learning techniques.

What is GNN?

A type of neural network model called Graph Neural Networks (GNNs) is created especially for the analysis of graph-structured data. GNNs can capture and model interactions between things represented as nodes and edges in a graph, whereas standard neural networks work with vectorized inputs. The capacity of GNNs to carry out tasks like node classification, link prediction, and graph-level prediction has drawn a lot of attention and popularity in recent years. This makes GNNs indispensable for a variety of applications in social network analysis, recommendation systems, and biological network analysis.

One of the main advantages of GNNs is their ability to concurrently detect both local and global structural patterns since they can combine data from nearby nodes in a graph. For applications where it is essential to comprehend the relationships and interactions between elements in a network, GNNs are especially well-suited. GNNs can efficiently train representations of nodes and graphs that encode rich semantic information and enable complicated reasoning and prediction tasks inside the graph domain by utilising strategies like message passing, graph convolution operations, and attention mechanisms.

Data Analysis:

The given dataset includes details about Twitch gamers and their characteristics, such as views, lifetime, maturity status, creation date, update date, numeric ID, status of deceased accounts, language, and affiliate status. This dataset's goal is to employ Graph Neural Network (GNN) analysis to forecast if a user would stream mature content on Twitch.

The dataset offers a wide range of features that are useful for GNN research. A Twitch user is represented by each entry, which includes details on the user's lifetime, number of views, maturity status (i.e., how mature the content is), creation and update dates, numeric ID, status of deceased accounts, language, and affiliate status. These characteristics will be used by the GNN model to determine, based on a user's profile and activity on Twitch, whether their content is mature. This analysis can be helpful for audience targeting, platform management, and content moderation. It can also improve user experience and ensure that content policies on the Twitch platform are followed.

```
from torch_geometric.datasets import Twitch
from torch.utils.data import random_split

# Import dataset from PyTorch Geometric with a valid dataset name
dataset = Twitch(root="EAI6020", name="RU", transform=None, pre_transform=None)

print(dataset)
print('------------')
print(f'Number of graphs: {len(dataset)}')
print(f'Number of features: {dataset.num_features}')
print(f'Number of classes: {dataset.num_classes}')
```

```
Twitch()
------------
Number of graphs: 1
Number of features: 128
Number of classes: 2
```

Figure 1

One graph with 128 features and two classes, representing a binary classification challenge, makes up the Twitch dataset. Now that the dataset has been processed, more analysis and model building are possible. Based on this information, it appears that the dataset has a substantial quantity of data about Twitch users, their activities, and possibly the qualities of their content. Predicting whether a user's material is mature could be part of the binary classification process, which could have an impact on Twitch platform audience targeting and content control.

```
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.RMSprop(model.parameters(), lr=0.001)
```

Figure 2

The optimizer utilised in the above code snippet is RMSprop, a well-liked optimisation method frequently applied to deep learning problems. RMSprop is renowned for its capacity to adaptively modify the learning rates according to the gradient magnitudes of each parameter. Compared to conventional optimisation techniques, RMSprop converges more quickly and effectively because of its flexibility. The hyperparameter that controls the step size during optimisation is the learning rate, which is set at 0.001. It is common practice to use a lower learning rate, such as 0.001, to guarantee stable convergence and avoid going above the optimal solution. For training models on the Twitch dataset, these optimizers and learning rate selections are essential since they have a big impact on the training procedure, rate of convergence, and overall performance of the model.

```
Epoch   0 | Loss: 0.94 | Acc: 24.77%
Epoch  10 | Loss: 0.75 | Acc: 40.07%
Epoch  20 | Loss: 0.68 | Acc: 55.78%
Epoch  30 | Loss: 0.65 | Acc: 63.08%
Epoch  40 | Loss: 0.64 | Acc: 66.73%
Epoch  50 | Loss: 0.62 | Acc: 69.21%
Epoch  60 | Loss: 0.61 | Acc: 70.31%
Epoch  70 | Loss: 0.60 | Acc: 71.33%
Epoch  80 | Loss: 0.60 | Acc: 72.22%
Epoch  90 | Loss: 0.59 | Acc: 72.70%
Epoch 100 | Loss: 0.58 | Acc: 73.07%
Epoch 110 | Loss: 0.58 | Acc: 73.41%
Epoch 120 | Loss: 0.57 | Acc: 73.84%
Epoch 130 | Loss: 0.57 | Acc: 74.09%
Epoch 140 | Loss: 0.56 | Acc: 74.18%
Epoch 150 | Loss: 0.56 | Acc: 74.48%
Epoch 160 | Loss: 0.56 | Acc: 74.66%
Epoch 170 | Loss: 0.56 | Acc: 74.94%
Epoch 180 | Loss: 0.55 | Acc: 75.07%
Epoch 190 | Loss: 0.55 | Acc: 75.10%
Epoch 200 | Loss: 0.55 | Acc: 75.07%
```

Figure 3

During a machine learning model's training phase, the supplied training progress displays the evolution of accuracy and loss metrics over several epochs. There is a lot of uncertainty and misclassification at the beginning of the loss, which begins at 0.94 with an accuracy of 24.77% at epoch 0. Nonetheless, both the accuracy and loss continuously increase as the training goes on over epochs. A decrease in the model's error rate is demonstrated by the loss, which drops to 0.55. Meanwhile, an increase in accuracy to 75.07% shows that the model is improving with time and becoming more predictive. This pattern implies that the model is successfully identifying characteristics and patterns in the data, which leads to better performance as training goes on.

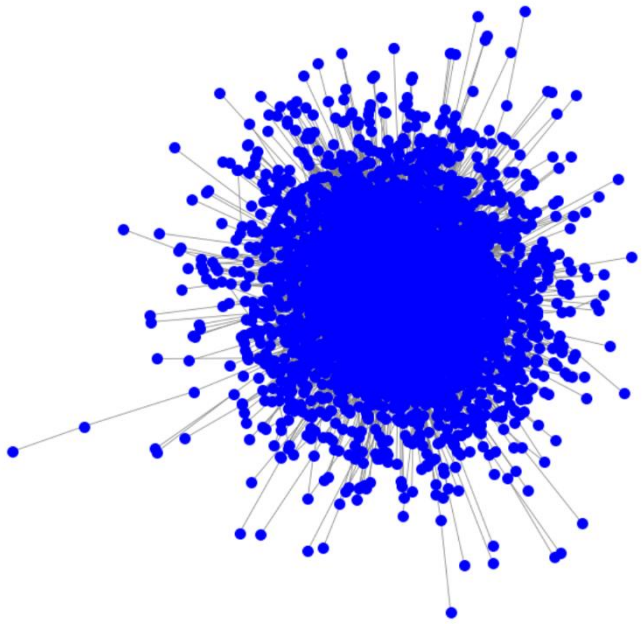

Figure 4

The image converts a graph data structure into a NetworkX graph for visualization, then plots it using a spring layout with blue nodes and grey edges, displaying the graph without axis. This image appears to be a visualization of a graph, with nodes represented by dots and the density of connections suggesting clusters within the graph.

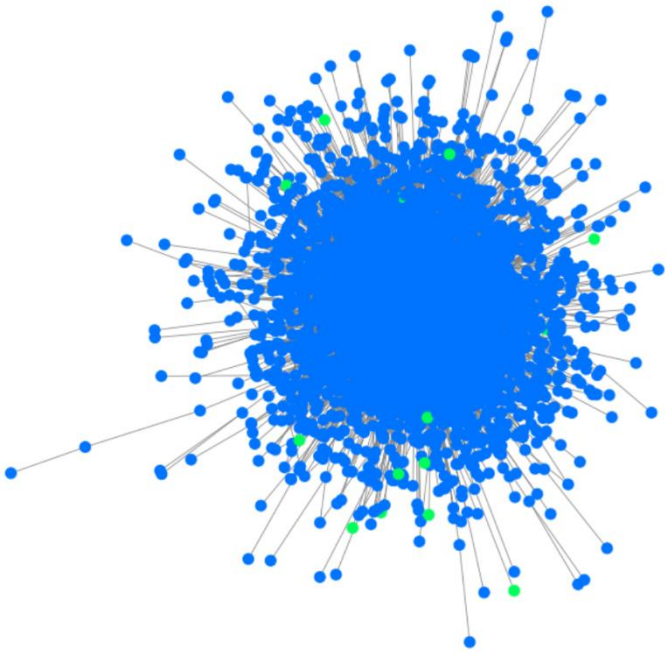

Figure 5

A graph with nodes color-coded according to their projected classes prior to training is shown in the image. Tightly coupled nodes are indicated by a dense centre cluster, whereas dispersed nodes point to less connected subgraphs or outlier points.

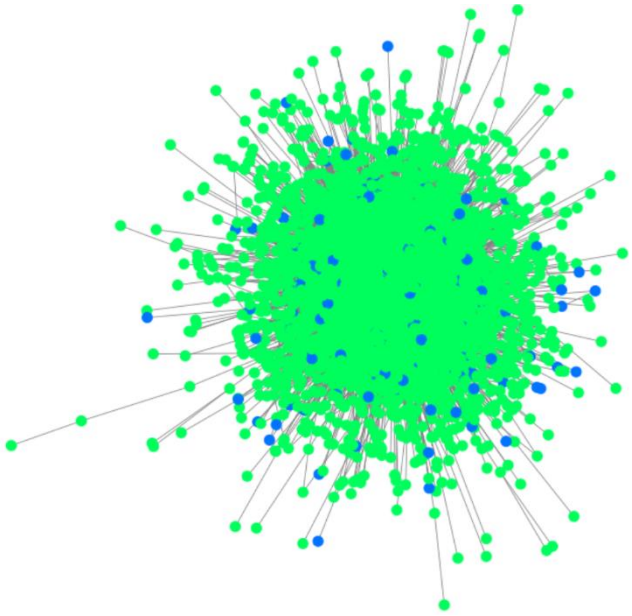

Figure 6

The picture shows a network where nodes are coloured according to the classes they are expected to belong to after training. At first, a variety of colours on the graph indicate that the nodes are projected to belong to different classes. But after training, there is a discernible rise in the quantity of green nodes, especially in the dense centre cluster. This shows that more nodes have been effectively classified by the model into the green class after training, resulting in a denser and more concentrated cluster of green nodes in the graph. Furthermore, the presence of dispersed nodes with differing colours indicates that distinct subgraphs or outlier sites have differing levels of connection and classification accuracy.

Conclusion:

In conclusion, the study explores whether users stream mature content by utilising Graph Neural Networks (GNNs) on the Twitch Gamer network dataset. The dataset is appropriate for GNN research because it provides comprehensive information about the behaviours, language preferences, and content maturity status of Twitch users. By utilising both local and global structural patterns inside networks, GNNs have revolutionised tasks such as node classification and link prediction since their inception in the context of graph-structured data. Additionally, the GNN model's training process shows a notable increase in accuracy over time, from 24.77% to 75.07% across epochs, demonstrating the model's capacity to capture underlying traits and improve prediction skills.

References:

- *torch_geometric.datasets.Twitch — pytorch_geometric documentation*. (n.d.). https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.Twitch.html#torch_geometric.datasets.Twitch