Comp 6730 Advanced Database Systems Project

GNN Implementation

Chris Winsor

12-December-2021

The project investigates GNN (Graph Neural Network) by way of two graph datasets: CORA (a citation network where nodes are documents and edges are citations), and ENZYMES (a graph dataset representing the protein structure of enzymes).

The toolset is PyG (PyTorch Geometric), an extension to PyTorch tailored for GNNs. GraphSage, GCN, and GAT are models within PyG that we will use.

Compute infrastructure: Google Colab, PyGym

Benchmarking: OGB (Open Graph Benchmark) which is like Kaggle but run by Stanford and focuses on graph datasets - see <http://ogb.stanford.edu/> and <https://arxiv.org/pdf/2005.00687.pdf>

## Graph Neural Networks and Embeddings

GNNs are different from traditional ML in both goal and approach. The goal of traditional machine learning is to map a set of attributes to some target class or measure. The dataset can be visualized as a table with attributes and class as the columns, and each row being a sample.

A Graph is a collection of nodes and edges where nodes represent things and edges represent relationships. The definition is very general - edges can be directed or undirected, the graph structure can be cyclic or acyclic, nodes and edges can include attributes.

This general structure means graphs are useful in representing many real-world problems. Some examples include drive time planning <reference>, social network relationships <reference>, <example, example>.

This general structure also presents new challenges when applying machine learning techniques. Where traditional datasets have fixed structure that can be directly leverages (an image dataset will have fixes X/Y pixel data) a graph dataset does not have this reliable structure. Thus one of the steps in establishing a graph network is establishing a consistent "compute framework" upon which the neural network can be based. The elements this are "similarity function", "encoder" and "computational framework".

Embeddings in graph networks are an active area of research and a key component of a GNN. An embedding is a vector associated with each node in the network. The key characteristic is that given two nodes that are similar, the dot product of their embeddings is high. This characteristic is used to summarize attributes from nodes, edges and network structure, into a representation that is convenient for the intended use. An example is visualizing the relationship between music genres for a selection of songs. <picture here>. Other uses would be recommending purchase based on past purchases.

## Theoretical Underpinnings

<here we review the theory behind embeddings, graphs>

## Code Snippets

<here we review the PyG code, classes, and procedures>

Message Passing in GCNs:

References:

# Theory:

PyTorch Models:

* GCN is based on the paper [“Semi-supervised Classification with Graph Convolutional Networks”](https://arxiv.org/abs/1609.02907). GCN uses GCNConv for message passing.
* GraphSAGE is based on the paper [“Inductive Representation Learning on Large Graphs”](https://arxiv.org/abs/1706.02216) paper. GraphSAGE uses SAGEConv for message passing.
* GAT is based on the paper [“Graph Attention Networks”](https://arxiv.org/abs/1710.10903) paper. GAT uses GATConv operator for message passing.

Ramp-Up

[**https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html**](https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html)

Implementation: <https://pytorch-geometric.readthedocs.io/en/latest/notes/create_gnn.html>