Implementation of Graph Convolutional Layers

Motivation: Graph convolutional layers (GCLs) are the building block of many different Graph Convolution Networks (GNNs). A GNN [1] is a class of deep neural network architecture for processing unstructured data that can be represented as a graph. The key design element of GNNs is the use of pairwise message passing, such that graph nodes iteratively update their representations by exchanging information with their neighbors [2]. There exist multiple types of pairwise message passing mechanism that allow the GNNs to extract spatial features between nearby or distant nodes in the graph, as well as recursive approaches that allow to simultaneously extract spatio-temporal features [3]. Relevant application for GNNs include all domains that handle graph data like social networks, molecular biology, and multivariate time series. This project aims to implement the most relevant GCLs inside Apache SystemDS -which currently lacks such architectures- and benchmark their performance on real life data. Like other open-source libraries as Pytorch Geometric [4] and TensorFlow GNN [5], providing SystemDS with such implementations will allow it to support the kinds of rich heterogeneous graph data that occurs in many real-life applications.

Task: This project is about extending SystemDS by adding graph convolutional neural network layers. Specifically, the implementation of two well-known GCLs: GCN [1] and GAT [6], as SystemDS's built-in functions. In both cases, the implementation consists of two functions that perform the forward and backward pass written on SystemDS's user defined language. The forward functions should receive, among other parameters, the feature matrix X, the weight matrix W, the graph's adjacency matrix A, and the bias vector b. As output, the function returns the new feature representation of the input. For example,

$$h = f(X(A - I)^{-1}W + b)$$

where f is an activation function passed as parameter of the function. The backward pass should receive, besides the previous mentioned input, the gradients with respect the output h and should return the gradients with respect to the input matrices X, W, and b. Other util functions may be necessary to implement the forward and backward functions. To test the implementations, we will compare the results obtained using SystemDS against Pytorch or Tensorflow to solve simple classification problems in datasets contained in NetworkX [8], e.g., Zachary's karate club. As well, we will use the datasets used in [1] and compared the results with those reported in the paper. Besides accuracy, we will record other important KPIs like execution time and memory consumption.

Hints on approaching this task:

- Familiarize yourself with (1) SystemDS and its user defined language, and (2) Graph Neural Networks and the linear algebra behind them. A very good tutorial is [9].
- A good first start is to implement the desired GCNs in python from scratch and test them. Extending it to SystemDS should take few efforts afterwards.
- The implementation should support sparse matrix multiplication. GCNs consist of matrix multiplication between the input, the weights, and the adjacency matrix. In most cases, the adjacency matrix A is sparse given that the number of edges is much less than the number of nodes. Considering that, X*A*W should be done using sparse matrix multiplications to avoid unnecessary computation.

- Study and understand the implementation provided in SystemDS for the Convolutional operation at “SystemDS/scripts/nn/layers/conv2d.dml(conv1d.dml)”. Also, study “Systemds\scripts\builtin\deepwalk.dml” as it provides an example of working with graphs in SystemDS.
- Conduct the experiments, visualize, and interpret the results.

[1] Kipf et. al. Semi-supervised classification with graph convolutional networks. IEEE Transactions on Neural Networks, 2016

[2] Sanchez-Lengeling, et al., "A Gentle Introduction to Graph Neural Networks", Distill, 2021.

[3] Daigavane, et al., "Understanding Convolutions on Graphs", Distill, 2021.

[4] Matthias Fey & Jan Lenssen, Fast Graph Representation Learning with Pytorch Geometric, ICLR, 2019

[5] Google Core ML, TF-GNN: Graph Neural Networks in TensorFlow, 2022

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio. Graph Attention Networks. International Conference on Learning Representations. 2018.

[7] W. Hamilton, Z. Ying, J. Leskovec. Inductive Representation Learning on Large Graphs. Advances in Neural Information Processing Systems, 2017.

[8] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, “Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[9] https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780