Deep Learning Approaches for Mutagenicity Prediction in Chemical Compounds

Jonas-Mika Senghaas, *mika.senghaas@epfl.ch*
Deep Learning in Biomedicine (CS-502), EPFL, Switzerland

***Abstract*—Report for Homework 2 in CS502 - Deep Learning in Biomedicine.**

I. Introduction

Graph-structured data is ubiquitous the biomedical domain. Amongst many more, chemical compounds such as molecules or proteins can be represented as graphs, and serve as the basis for a wide-range of research questions and applications in cheminformatics. Having expressive models for tackling these problems is therefore of great importance. This project aims to explore the potential of deep learning methods in the biomedical domain by applying graph convolutional neural networks to the task of mutagenicity prediction in chemical compounds on the `MUTAG` dataset. The results show that graph convolutional neural are a powerful tool for graph classification tasks and that they can be used to predict the mutagenicity of chemical compounds with high accuracy

II. Data

The MUTAG dataset is a popular benchmark dataset for graph classification in the biomedical domain. It consists of a total of a 188 graphs, each representing a chemical compound. The dataset is labelled, with each graph labelled as either mutagenic (positive class) or non-mutagenic (negative class). Both nodes and edges are attributed with categorical features, where nodes can be one of seven different atom types and edges can be one of four different bond types. Both node and edge features are one-hot encoded. In the following we will refer to a single graph G with N nodes through its node feature matrix $\mathbf{H}$ with dimension $N \times D$, its adjacency matrix $\mathbf{A}$ with dimension $N \times N$, its edge features, organised in a edge feature matrix $\mathbf{E}$ with dimension $N \times N \times P$ and its label $y \in \{0, 1\}$.

III. Methodology

The goal of this project is to accurately predict the mutagenicity of chemical compounds by learning both from from the graph's topology, its node - and later also edge - features. Architecturally, each model is a feed-forward graph neural network, that iteratively updates the node feature representation into a meaningful latent space. To perform the final graph prediction a global pooling operation is applied to the node features, which are then fed into a fully connected layer classification head with a non-linear sigmoid activation to interpret the final output as the probability of a graph being mutagenic. This project explored a total of four different graph convolutional layers and two different pooling operations.

A. Graph Convolution Layer

In the regular graph convolutional layer each node's representation is updated by averaging the representations of its neighbours, then applying a linear transformation and finally applying a non-linear activation function. Thus, updating the node representation $\mathbf{h}_v$ of node v in the l-layer can be expressed as

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_l \sum_{u \in N(v)} \frac{\mathbf{h}_u^{(l)}}{|N(v)|} + \mathbf{B}_l \mathbf{h}_v^{(l)}\right), \tag{1}$$

where $\mathbf{W}_l$ and $\mathbf{B}_l$ are both learnable weight matrices and σ is a non-linear activation function.

B. GraphSAGE

Graph convolutional layers can be seen as a special case of the more general GraphSAGE layer. Here, the node representation is updated by aggregating the neighbours' representations using a modular aggregator function AGG. The node representation is then updated by applying a linear transformation to the concatenation of the node's current representation and the aggregated neighbour information and finally applying a non-linear activation function. Again, we can express the update of the node representation $\mathbf{h}_v$ of node v in the l-layer as

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}_l \cdot \left[\mathbf{h}_v^{(l)} || \mathrm{AGG}\left(\left\{\mathbf{h}_u^{(l)}, \forall u \in N(v)\right\}\right)\right]\right), \tag{2}$$

following the same notation as in the previous section.

C. Graph Attention Layer

Graph attention layers [1] utilise the attention mechanism that is prevalent in the natural language processing domain [2]. Here, a node's neighbours are aggregated by computing a weighted sum of their representations, where the weights are computed by a learnable attention mechanism. The mechanism allows for each node to learn a different weighting of its neighbours' representations.

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{u \in N(v) \cup \{v\}} \alpha_{vu}^{(l)} \mathbf{W}_l \mathbf{h}_v^{(l)}\right), \tag{3}$$

D. Edge Convolution Layer

Finally, edge convolution layers update the node representations utilising the edge features. In this implementation, which is adapted from [3], each dimension of the edge feature matrix $\mathbf{E}_{..p} \forall p \in P$ is treated as its own edge feature. W

$$H^{(l+1)} = \sigma\left(\prod_{p=1}^{P} \tilde{E}_{..p} H^{(l)} W_l\right), \tag{4}$$

where $\tilde{E}_{..p}$ is the p-channel of the edge feature matrix $\mathbf{E}$.

IV. Experiments

Given the individual building blocks for graph convolutional and global pooling layers, we can now construct a variety of different models. All models tested follow the same custom graph neural network architecture, which is parametrised with the type of convolutional layer (including the type of non-linear activation function), the global pooling operation and the hidden dimensions in each of the convolutional layers. Table XY shows the different model configurations that were tested.

All experiments were conducted using `PyTorch` [4].

V. Findings

VI. Summary

References

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2018.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[3] L. Gong and Q. Cheng, "Exploiting edge features in graph neural networks," 2019.

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf