University of Vienna
Faculty of Computer Science
Assoz. Prof. Dr. Nils M. Kriege

Graph Learning
2025W

Programming assignment 1: Graph and node classification with graph neural networks

General Remarks:

- This programming assignment should be conducted in teams of up to 4 students.
- The deadline for the submission via Moodle is 15.12.2024 at 23:59.
- Use Python for solving the programming tasks. Your code must be well-documented and readable; see the Style Guide for Python[1] for common style conventions. You may use Jupyter Notebooks for your report in which you explain and visualize your results.
- Upload your solutions as a zip archive with the following naming scheme **team_X.zip**, where X is your team's number. The archive should contain your code as well as a PDF or HTML report describing your approach including your assumptions, results and instructions on how to compile and run your code. Additionally, you should prepare an 7-minute presentation of your approach and your main results using slides.
- Mark and cite external sources you are using in the code and report.
- You can use the Moodle forum to ask questions about the project or contact the tutors of the course.

1 Graph classification with PyTorch Geometric

Familiarize yourself with the library PyTorch Geometric[2] [1]. The Juypter Notebook `GNN_tutorial.ipynb` available in Moodle shows code examples that should be extended as described in the notebook. Follow the tutorial, solve all tasks and save your solutions and results. There are two options for running the Jupyter Notebook.

(a) Install PyTorch Geometric and Jupyter on your local system. The datasets used in the tutorial are small and a GPU is not required to complete the tasks. Please follow the installation instructions provided on the websites of these frameworks.

(b) Load, edit and run the Jupyter Notebook in Google Colab.[3]

2 Semi-Supervised Node Classification

Perform node classification using Graph Convolution Networks (GCN) [2] on the Cora dataset. The dataset can be downloaded using PyTorch Geometric using `from torch_geometric.datasets import Planetoid`. You will find various examples and tutorials using GCN and the dataset Cora online.

[1] `https://www.python.org/dev/peps/pep-0008/`
[2] `https://pytorch-geometric.readthedocs.io`
[3] `https://colab.google/`

(a) Find a GCN-based graph neural network that reaches at least 80% test accuracy on the Cora dataset using the standard split into train, test, and validation set. Consider varying the following settings: number of layers, number of hidden channels, learning rate, dropout, output layer, loss function, etc.

Report your findings and give details on the architecture that led to the best results.

(b) Analyze the homophily of the Cora graph dataset. Compute the node homophily and investigate its relationship to the predictive performance. Are nodes with low homophily more likely to be misclassified? Design experiments suitable to answer this questions and present your results appropriately.

(c) Analyze the influence of the number of convolution layers on the predictive performance. Do your results indicate that oversmoothing is an issue? Consider quantifying the degree of oversmoothing using a metric such as the rank-one distance [3], the error of low-rank approximations, or the Dirichlet energy [3]

$$\mathcal{E}(\boldsymbol{X}^n) = \frac{1}{2} \sum_{(i,j)\in E} \Big\| \frac{\boldsymbol{X}_i^n}{\sqrt{1+\deg(i)}} - \frac{\boldsymbol{X}_j^n}{\sqrt{1+\deg(j)}} \Big\|_2^2,$$

where $\boldsymbol{X}^n$ is the node feature matrix at layer n.

(d) Analyze how oversmoothing is affected by the labels of the dataset. Replace the labels of the Cora dataset with synthetic labels that encode the local structure of the nodes, e.g., using a suitable vertex invariant.

(e) Analyze how oversmoothing is affected by the choice of the activation function. Run experiments with a linear activation function, and the non-linear activations ReLU and sigmoid and compare the results.

References

[1] Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)

[2] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)

[3] Roth, A.: Simplifying the theory on over-smoothing (2024), `https://arxiv.org/abs/2407.11876`