Project description and feedback

July 21, 2024

1 Project description

In this project, you will implement a generative model for graphs based on the techniques discussed in the course. The model must be able to generate graph adjacency matrices (not node features). You may or may not use the node features in your solution.

1.1 Formalities

Hand in a single report in PDF format and your code in a single file (zip or tar archive). The report must include:

- A single page with the main text, including figures and tables. Include names, student numbers, course number, and the title "Mini-project 3".
- Unlimited pages of references.
- A single page of well-formatted code snippets.

Use at least font size 10pt and margins of at least 2cm. Content violating these limitations will not be evaluated.

1.2 Dataset

Use the MUTAG dataset introduced by Debnath et al. It is a collection of nitroaromatic compounds (molecular graphs) with the task to predict their mutagenicity on Salmonella typhimurium (graph-level binary classification). Vertices represent atoms and edges represent bonds. There are 7 discrete node labels representing atom types (one-hot encoded). The dataset contains 188 graphs.

1.3 Baseline

Implement an Erdös-Rényi model as follows:

- Sample the number of nodes N from the empirical distribution of the number of nodes in the training data.

- Compute the link probability r as the graph density (number of edges divided by total possible number of edges) computed from the training graphs with N nodes.
- Sample a random graph with N nodes and edge probability r according to the Erdös-Rényi model.

1.4 Deep Generative Model

Implement at least one deep generative model, such as a VAE with node-level latents, a VAE with a graph-level latent, or a GAN. Use either a message passing graph neural network or a graph convolution neural network. Include a technical description of your model in the report, with enough information to replicate your results. Include code snippets of central components of your implementation.

1.5 Sample Evaluation Metrics

Sample 1000 graphs from both the baseline and the deep generative model.

1.5.1 Novelty and Uniqueness

Compare the baseline and the deep generative model using the following metrics:

- **Novel:** Percentage of sampled graphs different from the training graphs.
- **Unique:** Percentage of sampled graphs that are unique.
- **Novel and Unique:** Percentage of sampled graphs that are both novel and unique.

Use the Weisfeiler-Lehman algorithm to compare graph isomorphism (e.g., using Python package NetworkX).

1.5.2 Graph Statistics

Compare statistics of the generated graphs to the empirical distribution of the training graphs. Plot histograms for:

- Node degree
- Clustering coefficient
- Eigenvector centrality

Align histograms for visual comparison.

2 Feedback

The report follows the required structure, and the required plots and table are included.

The technical description of the generative model is clear and there are no important details missing.

Baselines are correctly implemented, and baseline statistics match expectations. The baseline clustering coefficient does not match, not sure why.

The design choices related to the generative model are technically sound and meaningful

You managed to generate a reasonable number of unique and novel graphs that match the training statistics better than random.

However, a different choice for the histogram plot would help you visualise the differences in metrics better. Now it is relatively hard to interpret. Not sure if the VAE is better than the baseline?

The included code is readable, and it demonstrates your ability to implement a graph-generative model.