Comp 790 Project Proposal

Alex Kan[1] and Armaan Sethi[1]

[1]University of North Carolina at Chapel Hill

Our group is pursuing the Graph Autoencoders project. Our primary idea is to utilize a (Variational) Autoencoder to generate novel graphs, while representing our inputs as adjacency matrices. We will attempt to generate semantically valid graphs which represent general families of data of various types (social networks, molecules, paths). In terms of data, we are considering several different graph datasets from the Stanford Large Network Dataset Collection (https://snap.stanford.edu/data/) including but not limited to: social circles from Google+, Facebook, and Twitter, Wikipedia hyperlinks, Arxiv citations, US patents, Amazon product purchases and reviews, road networks, and Stack Overflow comments.

We are interested in creating a permutation invariant Generative Model of this type which results in better accuracy through the usage of Adversarial Training. We also intend to explore the benefit of different similarity metrics (KL-divergence, Laplacian Eigenmaps and Autoencoder ranking loss, L-2 Reconstruction Loss). Inspiration for many of these ideas came from the various Improvements and Discussions sections from the Deep Learning of Graphs Survey (https://arxiv.org/pdf/1812.04202.pdf).

Existing Methods that we will draw inspiration stem from recent methods in the literature such as: GraphVAE (https://openreview.net/pdf?id=SJlhPMWAW), GraphRNN (https://arxiv.org/pdf/1802.08773.pdf), Adversarially Regularized Graph Autoencoder for Graph Embedding (https://arxiv.org/pdf/1802.04407.pdf).

We will compare the results of our model to both traditional and modern graph evaluation benchmarks mentioned in the GraphRNN paper including:

- Traditional: The Erdos-Renyi model (E-R), the Barabasi-Albert (B-A) model, and those including Kronecker graph models, and mixed-membership stochastic block models (MMSB).
- Modern (Deep learning): GraphVAE, DeemGMG

Our timeline is as follows:

- Read Papers and Understand Benchmarks (1/28 - 2/1)
- Create trivial PyTorch representation of a Graph Autoencoder (2/3)
- Understand Similarity Metrics (2/4 - 2/10)
- Implement Adversarial Training and Refine Model (2/11 - 2/24)
- Obtain solid results (2/25 - 3/7)
- Write up Paper and refine results (3/12 - 4/23)

We will implement our ideas in PyTorch aided by the NetworkX library which is useful for the study and manipulation of networks and graphs.

To evaluate our models, we are concerned about whether our models may work well for general data of various types and scales, and whether our model will have the property of order invariance. Another consideration is also which similarity metric is most useful in informing the quality of our Generative Model.