Self-supervision for graph data structures

Wenbin Wang

wwenbin@bu.edu

1 Introduction

In this project. I decided to use Graph Neural Networks (GNN) method to implement the self-supervision graph data structure. GNN, is a class of neural networks for processing data represented by graph data structures. A typical application of GNN is node classification. Essentially, every node in the graph is associated with a label, and we want to predict the label of the nodes [1]. This topic covers both self-supervision and graph data structure.

2 Why GNN?

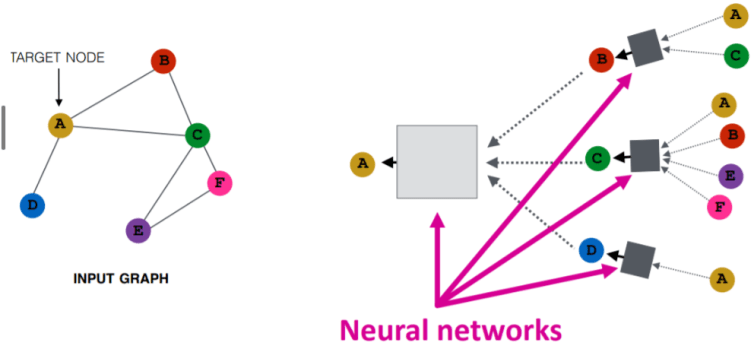
In this part, I want to clarify why I use GNN instead of another very similar model: Convolutional Neural Network (CNN).

In CNN, a very important idea is convolutional layer. Basically, convolutional layer is used to detect pattern. It used a filter to filter the image and get the result image. Using different filter can get different image with certain characteristic. In deepen layer, filters will detect more and more sophisticated object. The filter can move over every 3 by 3 block of pixels from the entire image. The filter is going to convolve across each 3 by 3 block of pixels from the input and get smaller image with certain characteristic. Using this strategy, we can avoid training all the image’s pixels into neural network and thus decrease the parameter input and make the model faster and easier to solve.

We can find the convolutional layer should use the filter, but the filter is a block. Thus, it only works when the local connections make sense. In a photo, local connections make sense, because it’s consisted of pixels and if we only pick one part of the photo, it still makes sense. For example, a dog’s photo if we only pick the leg’s part, this part is making sense and can be trained using machine learning. But what if we just pick one part of the graph? That does not make sense anymore because the graph is consist of nodes and edges and the local connection doesn’t make sense anymore. Thus, I pick the GNN.

3 How GNN works?

In this part, I want to briefly summarize how GNN works. In graph theory, we implement the concept of Node Embedding. It means mapping nodes to a d dimensional embedding space (low dimensional space rather than the actual dimension of the graph), so that similar nodes in the graph are embedded close to each other. Our goal is to map nodes so that similarity in the embedding space approximates similarity in the network [2]. We can use neighbors’ information and put it into neural network to get the final answer. As the picture shown below, we can use neural network to train the features which are generated by nodes neighbors’ information. And one of the researches shows that, self-supervised learning can enhance the performance of the GCN, which is a model of GNN. The research’s results show that, with properly designed task forms and incorporation mechanisms, self-supervision benefits GCNs in gaining both generalizability and robustness [3].



Using Neural Networks in GNN [4]

4 GNN Applications

GNN are widely use in today’s society. Below are where it uses based on my search.

First of all, the text classification. A classic application of GNNs in NLP is Text Classification. GNNs utilize the inter-relations of documents or words to infer document labels. We can use graph convolutional network to solve this problem.

Secondly, Neural machine Translation. The neural machine translation (NMT) is considered a sequence-to-sequence task. One of GNN’s common applications is to incorporate semantic information into the NMT task. To do this, we utilize the Syntactic GCN on syntax-aware NMT tasks [5].

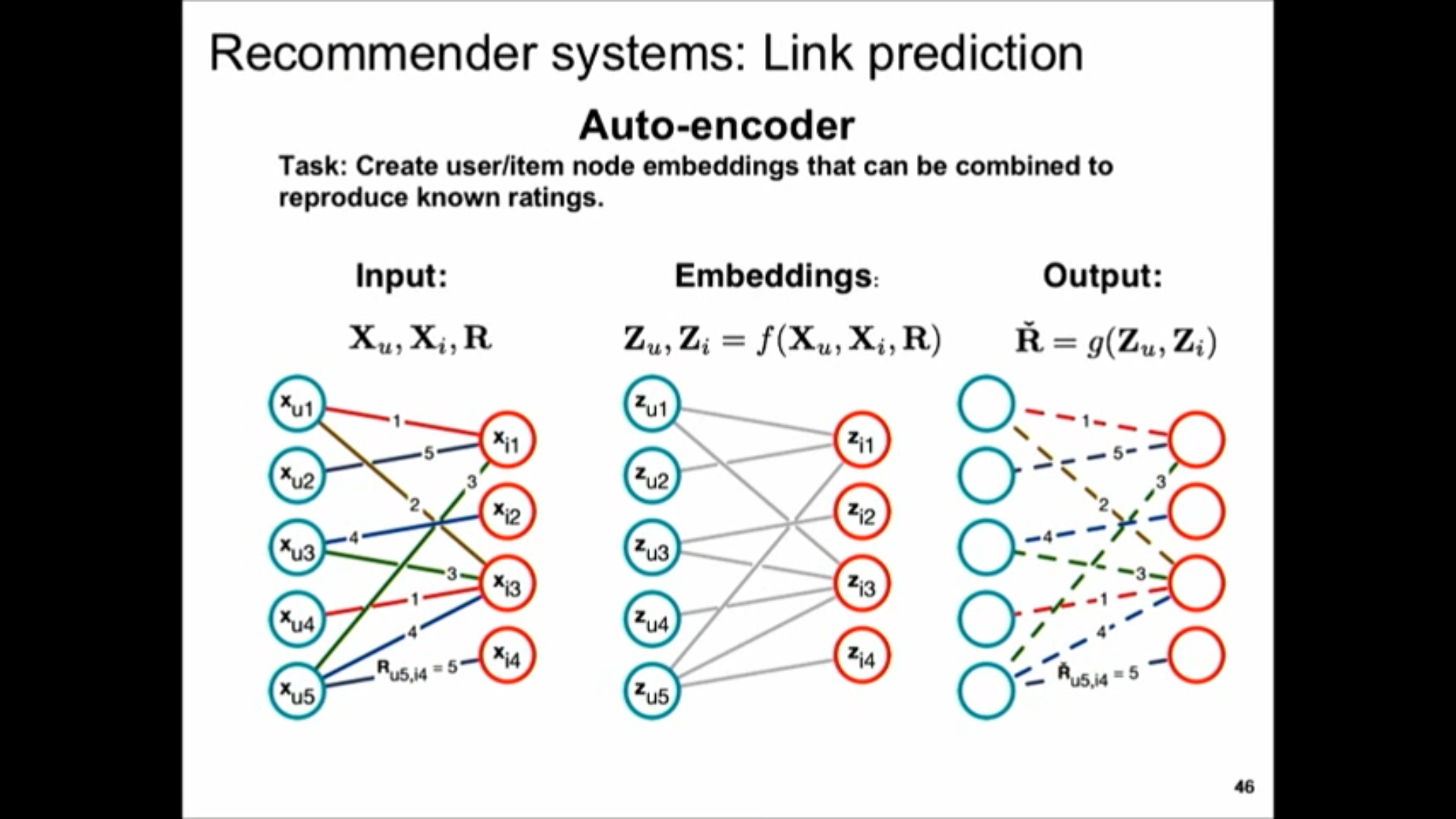
Thirdly, GNN can also be used in recommender system. In recommender systems, the main challenge is to learn the effective user/item representations from their interactions and side information (if any). Since most of the information essentially has graph structure

and GNN has superiority in representation learning, the field of utilizing GNN in recommender systems is flourishing [6].

Actually, it can be used in other many fields. But in my project, I want to design and optimize the recommender system using this self-supervision graph model.

5 My Product

My product, as I said before, is supposed to use in recommendation system. I am trying to build a more intelligent recommendation system. Because today’s recommendation system often recommends almost same content which are based on users’ recently browsed page or video and maybe I just click one tedious video by mistake, the recommendation system may recommend me this sort of videos many times. My users can be video watcher, online buyer, all the people who using the Internet for searching. How we can use the GNN to recommendation system is that we can transform the customer and their rating for items into bipartite graph. Using input to get a node embedding and using it to reproduce known ratings, like the picture below. Until now I find two ways to optimize the GCN. As I said before, using self-supervised method will improve its performance. Another way is graph attention design which is designed to assign larger weights to important neighbor nodes for better representation [7].



Using bipartite graph to denote recommendation system [8]

6 Literature review

In this part, I searched for some literature which related to my product part. Because these theses are kind of long and hard to understand. I try to summarize the important content for my product and detail some problem which is attracted me.

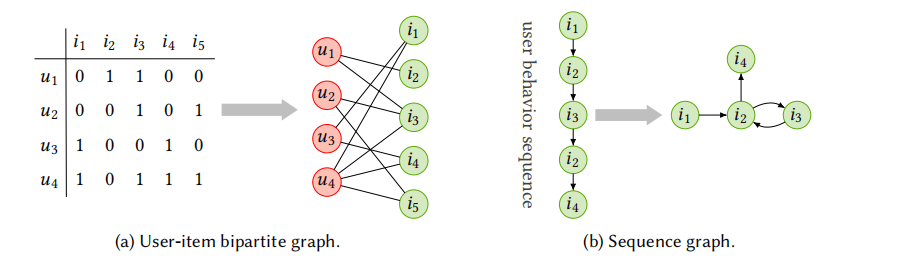
1. Intent Disentanglement and Feature Self-supervision for Novel Recommendation [9]

In this essay, the researcher using self-supervision and disentangled representation learning to achieve novel recommendation. They disentangled the intent into popularity and preference. The highlight of their study is that they develop a new paradigm for novel recommendation of cold-start items which uses the self-supervised learning to model the correlation between collaborative feature and content features. The item cold-start problem refers to when items added to the catalogue have either none or very little interactions [10].Because they have few interactions, the collaborative algorithm may not recommend the correct item. So we should add correlation algorithm and consider the trade-off between this two algorithm. Their work is different from capturing the features and labels from users and items’ bipartite graph. Instead, their first step is to consider the correlations between the collaborative features and user’s inherent features, then they adopt the generative and contrastive self-supervised learning jointly to tackle the cold-start problem.

1. Graph Neural Networks in Recommender Systems: A Survey [11]

This essay mainly provides a comprehensive review of recent research efforts on GNN-based

recommender systems. They divide the recommender system into two parts: General recommendation, which assumes the users have static preferences and models them based on either implicit (e.g., clicks, reads, or purchases) or explicit (i.e., ratings) feedbacks. Sequential recommendation, in contrast to the general recommendation, assumes the user’s preference is dynamic. Then they talk about some popular GNN frameworks which are used in recommendation field, like GCN, GraphSage, GAT, GGNN, etc. Using GNN in recommender system is very intuitive because it can be denoted as a graph. General recommendation can be denoted as bipartite graph and the sequential recommendation can be denoted by sequence graph. They also talk about the potential problem in GNN, which are graph construction, neighbor aggregation, information update, final node representation. I think the most important problem is neighbor aggregation, how to use appropriate method to propagate neighbors’ information to build relationship between two nodes. The mean-pooling method is easy for implementation but might be inappropriate when the important neighbors is significantly different. So, in order to solve this problem, I read another essay which related to this problem.



1. HOW TO FIND YOUR FRIENDLY NEIGHBORHOOD: GRAPH ATTENTION DESIGN WITH SELF-SUPERVISION [7]

In this essay, researchers focus on how to design a more accurate graph attention mechanism which can deal with the noisy graph and not just assign larger weights to important neighbor nodes by using self-supervised learning. They point out that recent models with graph structure learning suffer from high memory and computational complexity and they moderate the problem using graph attention which is without additional parameters. Finally, they summarize graph domains should be preceded to design graph attention. That is, by knowing the average degree and homophily of the graphs, we can choose the optimal graph attention in the design space.

References and Open-Source Link

1. https://en.wikipedia.org/wiki/Graph\_neural\_network
2. https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications
3. http://proceedings.mlr.press/v119/you20a/you20a.pdf
4. <https://web.stanford.edu/class/cs224w/slides/08-GNN.pdf>
5. https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications
6. https://arxiv.org/abs/1812.08434
7. Source: https://openreview.net/forum?id=Wi5KUNlqWty

1. <https://www.youtube.com/watch?v=0_O8PdZBc5s&list=PLBqFQO8mpMz580zhIjvn5CBKdid-hLxMQ&index=5&t=2452s>
2. https://arxiv.org/abs/2106.14388

1. <https://en.wikipedia.org/wiki/Cold_start_(recommender_systems)>

1. <https://arxiv.org/abs/2011.02260>
2. When Does Self-Supervision Help Graph Convolutional Networks? ( https://github.com/Shen-Lab/SS-GCNs/blob/master/README.md)
3. Graph Neural Networks: A Review of Methods and Applications (https://github.com/DenseAI/awesome-biomedical-machine-learning)