Learning from Counterfactual Links for Link Prediction

Aditya Choudhary and Sooraj A P

IITB

December 2, 2023

Running The Code

Due to computational requirements we resorted to running the code on Google Colab. However to satisfy the requirements we had to downgrade Colab's python version to 3.8.10 and CUDA to 10.2. These were required for torch v1.6.0 as higher versions of torch and corresponding PyG libraries had deprecation issues.

The accompanying Colab file is linked CS768.ipynb

Datasets

The code was tested on 5 datasets namely

- Cora
- CiteSeer
- Facebook
- PubMed
- OBG-DDI

The primary metrics used were

- AUC Score
- Hits@20

Our Proposed Work

Our proposed ideas were ;

1. To test the performance of the code when trained with different norms than the one used in the paper, which was the Frobenius norm.
2. To try and use variables for distances of nodes from communities that were not just 0/1 binaries to try and incorporate the concept of 'closeness from a cluster'.

All our experimental results are from tests run on the CORA and CiteSeer datasets as the others would not load on Colab and we did not have access to a machine running CUDA10-2, which was crucial for us to run the data.

Results and Inferences

As part of our contributions we included KL Divergence Loss for $\mathcal{L}_{disc}$. There were no significant differences in the results bwtween Frobenius norm and KL Divergence.
Where we did see an improvement though, was training times as the train time for KL Divergence was lesser than that for the Frobenius Norm by a factor of 16% for CiteSeer and 25% for Cora. The nearly identical results can be observed in the coming slides, where we compare the reported results of the authors to our results with both norms.

	Frobenius	**KL**
Cora	62min.	46 min.
CiteSeer	92min.	77 min.

CiteSeer Dataset

	Reported	**Calculated (Frobenius)**	**Calculated (KL)**
AUC	0.9212±0.0047	0.9336±0.0044	0.9328±0.0044
AP	0.9392±0.0041	0.9461±0.0038	0.9456±0.0035
Hits@20	0.6809±0.0149	0.6996±0.0278	0.6984±0.0181
Hits@50	0.7701±0.0192	0.7974±0.0149	0.7934±0.0135
Hits@100		0.8490±0.0135	0.8497±0.0120
Best Val		0.9474±0.0030	0.9464±0.0044

Cora Dataset

	Reported	**Calculated (Frobenius)**	**Calculated (KL)**
AUC	0.9305±0.0024	0.9320±0.0036	0.9328±0.0031
AP	0.9424±0.0028	0.9440±0.0026	0.9441±0.0022
Hits@20	0.6557±0.0105	0.6383±0.0184	0.6293±0.0217
Hits@50	0.7549±0.0154	0.7537±0.0109	0.7510±0.0144
Hits@100		0.8360±0.0068	0.8364±0.0083
Best Val		0.9386±0.0027	0.9377±0.0035

Results and Inferences

The differences in reported and evaluated results can be attributed towards hardware differences. Google Colab used by us utilized Nvidia T4 GPU with 15 GB of system and GPU RAM. The original code was evaluated on a Linux server with Intel Xeon Gold 6130 Processor (16 Cores @2.1Ghz), 96 GB of RAM, and 4 RTX 2080Ti cards (11 GB of RAM each)

Calculated results were higher for all instances except Hits@20 and Hits@50 for Cora. for Cora, calculated value for Hits@50 were slightly less but well within the standard deviation reported. The mean for Hits@20 was deviated from the reported figure by more than 1 standard deviation.

Our 'Multi-Class' Idea

The second among our proposed ideas was to try and capture the idea of "distances between communitites" (cosine similarity between centers of clusters being the distance metric) as opposed to just membership in a community. The modifications we made to the code can be found in the accompanying report. Just as above, we trained our multi-class model on the CiteSeer and CORA datasets and got the following results, We compare our results after training the multiclass variant of our code with the results we obtained using standard treatment, a complete comparison with the reported data is available in the report.

MultiClass Results on CiteSeer

	Standard	**Multiclass**
AUC	0.9336±0.0044	0.9417 ±0.0033
AP	0.9461±0.0038	0.9546±0.0031
Hits@20	**0.6996**±0.0278	**0.7542**±0.0234
Hits@50	0.7974±0.0149	0.8291±0.0187
Hits@100	0.8490±0.0135	0.8750±0.0122
Best Val	0.9474±0.0030	0.9583±0.0031

Multiclass treatment on CORA

	Standard Treatment	**Multiclass treatment**
AUC	0.9320±0.0036	0.9339±0.0039
AP	0.9440±0.0026	0.9443±0.0030
Hits@20	0.6383±0.0184	0.6430±0.0281
Hits@50	0.7537±0.0109	0.7553±0.0166
Hits@100	0.8360±0.0068	0.8338±0.0114
Best Val	0.9386±0.0027	0.9451±0.0033

Results and Inferences

Significant improvement was noticed for the CiteSeer dataset, in particular Hits@20 improved from 69.96% to 75.42%, noticeable improvements are also present for other metrics as well. However for Cora Dataset there isn't significant improvement in performance. One possible reason for this could be that KCore for Cora provided only 4 labels, the centers of these 4 labels had cosine similarity (used for calculating multi-class treatment) quite close to 0 and 1, hence the treatment values were very similar to the binary treatment for lot of nodes.

Where we did see an improvement though, was training times as the train time for Multiclass approach was lesser than the standard by a factor of 10% for CiteSeer and 24% for Cora. This was observed because Multiclass converged much faster than standard approach and Early Stopping condition was invoked in a lesser number of epochs

Results and Inference

	KCore	**KCore with Multiclass**
Cora	62min.	47 min.
CiteSeer	92min.	83 min.

Other Datasets

PubMed Dataset failed to load in Google Colab's memory due to a significantly higher count of nodes (19717 vs 3327 for CiteSeer) which led to significantly larger node embedding matrix. Hence the code was auto-killed.

For Facebook and OBG-DDI, Colab was able to load the data however computation was time-taking due to high number of links (88234 for Facebook, 1334889 for OBG-DDI vs 5278 for Cora) which led to Colab crashing on several accounts and running out of memory during execution

Attempts Made

After communicating the problems faced with Colab we were alloted DGX9 server at IITB for computation. However we faced challenges with it. The server has CUDA 11.4 drivers and the code is compatible with torch v.1.6.0 and accompanying PyG libraries which required CUDA 10.2 or lower. Upgrading library versions to those compatible with CUDA 11.4 was infeasible due to dependency issues and deprecation issues

We tried downgrading to CUDA version but were unsuccessful due to lack of root access due to which we couldn't install CUDA 10.2. After that we were allotted DGX2 which we had requested for Docker however we found that docker wasn't installed for us.