CS 224W Project (2021 Fall)
Tutorials and Case Studies for Applying Graph ML to Real-World Problems

Table of Contents

Overview

For this year's CS224W course project, we want to develop a set of tutorials and case studies of applying Graph Machine Learning to a given problem domain. The goal is to create a long-lasting resource, which will help the community and further popularize the field of graph machine learning and PyG/GraphGym GNN library in particular.

We ask you to create a Medium-based-blog tutorial/case study on applying state-of-the-art graph ML to a real-world application. Making a blog tutorial is a great exercise for you to better understand and implement graph ML algorithms to solve real problems. Moreover, your graph ML blog will become a great resource for the broader community to study the emergent field of graph ML. After our reviews, you can also share your blog posts on your own websites / social media accounts to impress your peers! Students can form a group of up to 3 people to work on the project. We are going to provide a place for you to publish a blog post.

For the implementation tool, we highly recommend using the PyG, the most popular graph deep learning framework built on top of Pytorch. As we teach in the course, PyG is suitable to quickly implement GNN models, with abundant graph models already implemented. Moreover, GraphGym, available as part of the PyG, allows a training/evaluation pipeline to be built in a few lines of code.

We are going to publish the best tutorials on PyG.org website and promote them via different channels, such as Twitter!

Real-world application domains of graph ML

Below, we list several real-world application domains, specifying how graphs and tasks are defined in these domains, as well as representative graph ML models and public datasets that can be used for your investigation. These are just examples; you should feel free to investigate other domains, models, and public datasets.

- **Recommender systems**

Graphs:
- Nodes: Users, items
- Edges: User-item interactions

Tasks:
- Predicting the edge ratings. Metric: RMSE
- Predicting edge existence. Metric: Recall@K

Model(s): LightGCN
Public datasets: Movielens, Recsys repository

- **Molecule classification**

Graphs:
- Nodes: Atoms
- Edges: Bonds

Tasks:
- Predicting properties of molecules. Metric: ROC-AUC

Model(s): See the leaderboard
Datasets: ogbl-molhiv

- **Paper citation graphs**

Graphs:
- Nodes: Papers
- Edges: Paper citations

Tasks:
- Predicting subject areas of papers. Metric: Classification accuracy

Model(s): See the leaderboard
Datasets: ogbn-arxiv

- **Knowledge graph**

Graphs:
- Nodes: Entities
- Edges: Knowledge triples

Tasks:
- Predicting missing triples. Metric: Mean Reciprocal Rank (MRR)

Model(s): TransE, DistMult, ComplEx, RotatE
Datasets: FB15k-237, WN18RR

- **Author collaboration networks**

Graphs:
- Nodes: Authors
- Edges: Author collaboration

Tasks:
- Predicting future author collaboration. Metric: Hits@50

Example model(s): See the leaderboard
Datasets: ogbl-collab

- **Heterogeneous academic graph**

Graphs:
- Nodes: Papers, authors, institutions, fields of study
- Edges: an author is “affiliated with” an institution, an author “writes” a paper, a paper “cites” a paper, and a paper “has a topic of” a field of study

Tasks:
- Predicting paper publication venues. Metric: Classification accuracy

Example model(s): See the leaderboard.
Datasets: ogbn-mag
Note: This is a medium-scale OGB dataset that requires more computational resources than the other datasets.

- **Product co-purchasing graph**

Graphs:
- Nodes: Products
- Edges: Product co-purchasing relations

Tasks:
- Predicting product categories. Metric: Classification accuracy

Example model(s): See the leaderboard
Datasets: ogbn-products
Note: This is a medium-scale OGB dataset that requires more computational resources than the other datasets.

- **Fraud Detection in Transaction Graphs**

Graphs:
- Nodes: Financial users (customers, banks)
- Edges: Transaction (money and amount sent)

Tasks:
- Edge classification - predict which edges are fraudulent. Metric: Hits@50

Example model(s): See https://github.com/safe-graph/graph-fraud-detection-papers
Datasets: Bitcoin Fraud Dataset (only use labeled data!)

- **Protein-Protein Interaction Networks**

Graphs:
- Nodes: Gene nodes
- Edges: Interaction between gene nodes

Tasks:
- Node classification - protein function prediction. Metric: Classification accuracy

Example model(s): See methods on OGB node classification leaderboard
Datasets: https://snap.stanford.edu/biodata/datasets/10013/10013-PPT-Ohmnet.html

- **Drug-Drug Interaction Network**

Graphs:
- Nodes: FDA-approved or experimental drug
- Edges: interactions between drugs (joint effect of taking the two drugs together)

Tasks:
- predict drug-drug interactions - Metric: Hits@K

Example model(s): See the leaderboard
Datasets: ogbl-ddi

- **Friend recommendation**

Graphs: social network
- Nodes: users
- Edges: potentially heterogeneous -- friend, follow, reply to, message, like, etc.

Tasks:
- Recommending/ranking new friends for user -- metrics: Hits@K, NDCG@K, MRR

Example model(s): GraFRank (paper, GitHub)
Datasets: Facebook, Google+, Twitter

You are free to choose any of the above domains and find other public datasets for the application that you choose. You can adopt the readily available graph datasets from Open Graph Benchmark (OGB)* and the datasets available in PyG. Alternatively, you are free to explore other application domains and the corresponding public datasets yourself.

*Note that some OGB datasets are quite large and require a decent amount of computational resources to handle. We recommend using small-scale OGB datasets that are more tractable.

Resource on graph ML

State-of-the-art models for graph ML can be found at
- OGB leaderboard
- Top ML conferences
  - To find papers related to graph ML, you can search for the term “graph” across the titles.
  - ICML 2019: http://proceedings.mlr.press/v97/
  - ICML 2020: http://proceedings.mlr.press/v119/
  - ICML 2021: https://proceedings.mlr.press/v139/
  - NeurIPS 2019: https://papers.nips.cc/paper/2019
  - NeurIPS 2020: https://papers.nips.cc/paper/2020
  - ICLR 2019: https://openreview.net/group?id=ICLR.cc/2019/Conference
  - ICLR 2020: https://openreview.net/group?id=ICLR.cc/2020/Conference
  - ICLR 2021: https://openreview.net/group?id=ICLR.cc/2020/Conference
  - KDD 2019: https://www.kdd.org/kdd2019/proceedings/
  - KDD 2020: https://www.kdd.org/kdd2020/proceedings/
  - KDD 2021: https://www.kdd.org/kdd2021/proceedings/

A Tutorial on Writing Blog Posts

First, please read this article carefully to learn about how to write machine learning blog posts. For this course project, please also follow the instructions below.

In the blog posts, you should include the following:
- At the beginning of your blog post, include “By XXX, YYY, ZZZ as part of the Stanford CS224W course project.”, where XXX, YYY, ZZZ are the names of the team members.
- The domain that you are applying graph ML to.
- Dataset descriptions (source, pre-processing etc).
- Step-by-step explanation of graph ML techniques you are applying.
  - You can assume the following for the readers.
    - Readers are familiar with machine learning (e.g., CS229) and deep learning (e.g., CS230) concepts. You do not need to explain them in detail.
    - Readers are familiar with Pytorch.
    - Readers are not familiar with graph ML (taught in this course.)
- **Visualizations** that would make the blog posts intriguing to read.
  - Gifs > Images > Text to show your methods and results
  - Try to use videos, images, flow charts as much as possible
  - The more visualization, the better. Reading text-occupied blogs is often painful.
- Some code snippets of how you used PyG/Pytorch to implement the techniques
- Results that you obtain using the model on the dataset
- Image credits if you are adopting figures from other places (we encourage you to make your own figures).
- Link to your Google Colab that can be used to reproduce your results.
- Avoid criticizing research / research orgs. You are here to showcase your work, not to write opinion pieces.

A good blog post should
- be fun to read with many figures and visualizations.
- be easy to follow even for graph ML novice.
- clearly convey the potential of graph ML to the application domain.
- contain a good amount of PyG code snippets to understand how PyG is utilized to build the model.

Using Medium
For this course project, please use Medium to write your blog post. Writing a blog post in Medium is super easy and intuitive. Below are the step-by-step instructions
- Sign up / sign in on Medium.
- For each group, one member should set up a draft
  - To start a new draft, go here.
  - To restart from the existing draft, go here.
- Editing the draft
  - Only one member (who owns the draft) can directly edit the blog post.

    - If you want to write the blog together with your members, we suggest you work on the Google doc first and then copy-paste the eventual texts/images to the Medium draft.
- Submitting your draft
    - The final product of the course project will be the **draft link** of your blog post.
    - On the editing page, click the "..." button (located right next to the "publish").
    - Then click "Share draft link" to get the link.
    - Please also share your account name with us.
        - We will add you as a writer to the stanford-cs224w publication page.
        - Once you are added as our writer, you can click "..." button and further click "Add to publication".
    - **Do NOT publish your blog without our review.**
- TA reviewing & grading
    - Once you have submitted your draft link as your final course project, TAs will then review and grade your blog posts.
    - For those blogs that are easy-to-follow and technically sound, we would like to publish your blogs (under your accounts). We will also advertise your blogs under PyG.org and through our social media.
    - You will get grading regardless of whether your blogs are published or not.

Examples of good blog posts:
- http://peterbloem.nl/blog/transformers
- https://tkipf.github.io/graph-convolutional-networks/
- https://towardsdatascience.com/graph-neural-networks-as-neural-diffusion-pdes-8571b8c0c774
- https://blog.twitter.com/engineering/en_us/topics/insights/2021/temporal-graph-networks

Instructions on Google Colab

Google Colab should include:
- a high-level summary of what the code is about and what the task is
- all the code to reproduce your results in the blog posts (including data preprocessing, model definition, and train/evaluation pipeline).
- detailed comments of what each cell does.

Examples of good Google colabs can be found at
- https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html

Evaluation criteria (20% of the total course grade)

Project proposal (20% of the project grade) [By October 19, 11:59 pm PST]

- Application domains (10% / 20%)
    - Which dataset are you planning to use?
    - Describe the dataset/task/metric.
    - Why did you choose the dataset?
- Graph ML techniques that you want to apply (10% / 20%)
    - Which graph ML model are you planning to use?
    - Describe the model (try using figures and equations).
    - Why is the model appropriate for the dataset you have chosen?
- Note: It is ok to change the domains and techniques in your final proposal. The main purpose of the proposal is for you to get started early and for us to give you early feedback.

Format: Please use the Latex file of the NeurIPS conference:
https://nips.cc/Conferences/2020/PaperInformation/StyleFiles
Please aim for 2 pages without references. The abstract is not needed.

Final project (80% of the project grade) [By December 9, 11:59 pm PST]

- Blog posts (60% / 80%)
    - Blog structure and organization (20% / 60%)
    - Effective use of figures/gifs to explain the concepts/techniques (20% / 60%)
    - Technical soundness (20% / 60%)
- Google Colab (20% / 80%)
    - Code correctness (10% / 20%)
    - Documentation (10% / 20%)

Extra credit for PyG (up to 3% of total course grade)

- You will also be considered for up to an extra 1-3% of the total grade if you create an approved pull requests to OGB (https://github.com/snap-stanford/ogb), PyG (https://github.com/pyg-team/pytorch_geometric), GraphGym (https://github.com/pyg-team/pytorch_geometric/tree/master/torch_geometric/graphgym) that add useful functionalities or fix bugs, depending on the significance of the contribution.
- You are highly encouraged to participate in the open-source development of our own in-house GNN platform PyG, such as
    - finding bugs and creating issues

    - finding interesting methods/papers to implement, creating a feature request, discussing the overall interface and implementation, and implementing the final proposal
    - proposing missing functionality and implementing it
    - writing Google Colabs/tutorials to show-case specific GNN pipelines/architectures/training protocols via PyG
    - contributing useful modules to GraphGym pipeline (https://github.com/pyg-team/pytorch_geometric/tree/master/graphgym/custom_graphgym)
    - contributing your final project to the OGB leaderboard together with open-source code using PyG. This is particularly exciting if your model is able to achieve SoTA performance.
- We highly recommend consulting the TAs or PyG lead developers (Matthias Fey, matthias.fey@tu-dortmund.de and Jiaxuan You, jiaxuan@cs.stanford.edu) before contributions so that we can confirm the details of incorporating the contribution and granting the extra grade.