Graph Counterfactual Explanations with GRETEL

December 26, 2023

Abstract

You will use GRETEL, a tool for counterfactual explainability in graphs. The goal of the project is either (A) to become an expert in using a well-defined tool to benchmark current state-of-the-art (SoA) methods in counterfactual explainability on some datasets or (B) to devise a new counterfactual explainer in graphs and assess its performances on three known datasets.

There mustn't be any documentation or report for the project. You'll get full marks if your code runs and produces the same/similar results as in your `output` folder.

General rules

1. *You might need a GPU and Amazon AWS credits ($300 should be given for free to students) to perform your experiments.*
2. You can form groups of at **least two** and **most four** people. Please register your group members here.
3. You can either do Task A or B. Not both.
4. Please fork the GRETEL project and pull changes frequently (e.g., it's advisable to pull at least once a week) to avoid having your fork outdated from the main project.
5. The delivery will follow these steps:
   - You fork the project into your repository on GitHub. Make this fork a private project.
   - Add `bardhprenkaj` as a collaborator in your fork of the project. Don't forget to add your other group members as well.
   - Keep pushing your changes (for either Task A or B) to your repository.
   - The project must contain an `environment.yml` that contains all the packages you installed. I can then use this file to recreate the same environment as your computers. This ensures that I can reproduce everything without worrying about different versions of python libraries. You can create this file by dumping your conda environment into it.
   - When you run the SoA explainers on the datasets you implement and oracles you optimize, you'll have `.jsonc` files that contain the output of the runs. **Don't delete these files**; I will rerun the code and see if it reproduces these results.
   - Put your `.jsonc` configuration file under a folder called `config/submission`.
   - Before you run your experiments, clear the folder `output`. Ensure that the `output` folder contains only your experiment folders, which for task A should be 25, and for task B should be 15.
   - When you want to deliver the project, write an email to `prenkaj@di.uniroma1.it` with the subject line `Delivery of ML Project`, putting your group members in CC. The content of the email must contain only the URL of your project on GitHub; nothing else.

A Task A

First, ensure that one of the group members books one dataset you will implement. **Only one of your group members** must use this this link for the booking. **Don't make more than one booking per group.** Once you have your choice (**watch out that the form doesn't return what you chose; make sure you save the link of the dataset a priori**), download the dataset, and place it under the folder `data/datasets`.

For your chosen dataset, train a Graph Convolution Network (GCN). You can use the configuration file in `config/TCR-1000-5k-0.3_GCN_RSGG.jsonc`, specifically, the `oracle` keyword, as a building block. Naturally, it would be best if you played around with the optimizer, the loss function and the parameters of `src.oracle.nn.gcn.DownstreamGCN`. It would be best if you also optimized the hyperparameters of the GCN. Unfortunately, GRETEL doesn't allow you to perform the hyperparameter search as is, so you must figure out a way to do it without breaking the flow of how things currently work in the framework. Once you have satisfactory performances, freeze the weights and leave the oracle be. You'll use it for the explainers.

You need to write 5 (the number of explainers) $\times$ 5 (for the number of folds; from 0 to 4) = 25 different configuration files, each for the following explainers on the dataset you ported into GRETEL:

- `src.explainer.generative.gcountergan.GCounteRGAN`
- `src.explainer.generative.clear.CLEARExplainer`
- `src.explainer.generative.cf2.CF2Explainer`
- `src.explainer.search.obs.ObliviousBidirectionalSearchExplainer`
- `src.explainer.search.i_rand.IRandExplainer`

The keyword `fold_id` for the oracle will always be -1, and that of the explainer changes from 0 to 4. All of these explainers have hyperparameters. Choose a sound number of hyperparameters (e.g., CLEAR has many, so choose a subset) and optimize them using a hyperparameter search on your chosen dataset. Run the optimized explainers with the configuration files you wrote, and report the results in the `output` folder.

B Task B

I expect only 5% of the groups to choose this task. If you don't choose it, you won't be punished. This task suits students who are more curious about research than software engineering. **If you choose to implement an ad-hoc explainer that solves one of the research questions in `lab6` (see last slide on GitHub), we can think of publishing it with all the group members as authors (shall you want to pursue this).**

You will implement a SoA counterfactual explainer by extending GRETEL's base classes (e.g., `Explainer` and/or `Trainable`). One of your group members must use this link to book your explainer. You will use three datasets to train your oracle and explainer: i.e., TreeCycles (see `config/TCR-150-100-0.3+TCC+pRand.jsonc` for an example of the dataset snippet), Autism ASD (see `config/ASD+SVM+Graph2Vec_DCES.jsonc` for an example of the dataset snippet), and BBBP (see `config/BBBP+GCN+DCES.jsonc` for an example of the dataset snippet). The TreeCycles dataset should be generated with `"parameters": { "num_instances": 5000, "num_nodes_per_instance": 28, "ratio_nodes_in_cycles": 0.3 }`. The other datasets have default parameters and will be read from the file system.

For each dataset, train a Graph Convolution Network (GCN). You can use the configuration file in `config/TCR-1000-5k-0.3_GCN_RSGG.jsonc`, specifically, the `oracle` keyword, as a building block. Naturally, it would be best to play around with the optimizer, the loss function and the parameters of `src.oracle.nn.gcn.DownstreamGCN`. It would be best if you also optimized the hyperparameters of the GCN. Unfortunately, GRETEL doesn't allow you to perform the hyperparameter search as is, so you must figure out a way to do it without breaking the flow of how things currently work in the framework. Once you have satisfactory performances, freeze the weights and leave the oracle be. You'll use it for the explainers.

You need to write 5 (the number of folds; from 0 to 4) $\times$ 3 (the number of datasets) = 15 different configuration files for the explainer you chose to implement. The keyword `fold_id` for the GCN oracle will

always be -1, and that of the explainer changes from 0 to 4. Your explainer might have hyperparameters which you need to optimize for each dataset separately.

Run the optimized explainer with the configuration files with the configuration files you wrote, and report the results in the `output` folder.