SI 671: Homework 2: Graph Mining

Due: Wednesday Nov 14th, 2018 by 11:59pm

Version 1.1 (Updated: Nov. 8th, 2018)

Summary

Have you ever wondered why someone is being recommended for you to “connect with” on a platform like Facebook or LinkedIn? Homework 2 will let you try your hand at this type of matchmaking on a large social network as you practice graph mining! Many online platforms include a social network, either explicitly through friendship links or implicitly through seeing two people communication or performing an action like co-editing. Social networks can serve as extremely valuable data for learning about people and studying society. Many people are similar to their friends in demographics (age, education level) and also in preferences (football teams, brand affiliations), which is formally known as homophily. Homework 2 focuses on mining a large social network to uncover how well homophily can predict identity as well as the network structure.

Task

Homework 2 consists of two tasks on the same large social network:

- **Task 1: Link Prediction**. You’ll be given a mostly complete social network, which has some (large) number of edges removed. Your task will be to predict up to 50K edges that your system predicts are most-likely. This task is a core component of many of friend-recommendation systems you see on real platforms like Facebook, LinkedIn, and Twitter. **This task is submitted through Canvas (not Kaggle).**
- **Task 2: Attribute Prediction**. For the same network, most of the nodes in the social network are provided with one more attributes, which can be drawn from different types (e.g., they might represent age, occupation, musical preference, etc.). Your task will be to predict the probabilities of attributes for a set of completely unlabeled nodes.

There are multiple ways to approach each task and no one way (or combination of ways) is expected to perfectly solve the task. Indeed, some approaches might try to solve both tasks *jointly* (which is harder). For Homework 2, you’re allowed to use any solution or combination of solutions to approach both tasks (though see details later).

Learning Objectives

Homework 2 has the following learning objectives:

- Knowledge of graph/network terminologies and basic operations that can be performed on them
- How to work with large network data (e.g., graphs with millions of nodes)
- Familiarity with one or more network/graph libraries
- Ability to break down a large problem/dataset into smaller, tractable subproblems
- Familiarity with different link prediction methods
- Familiarity with different attribute prediction methods

Data

Five data files are provided with the homework. The first file is `network.tsv`, which is a large social network specified in *edge list* format:

```
vertex1 vertex2
vertex1 vertex3
```

Which is essentially two columns where each column is the label for a node/vertex in the graph and the order in which the vertices are specified does not matter (this is an undirected graph). The above edge list specifies a graph with three vertices and two edges. In your data, nodes are assigned random numeric IDs that have no meaning. Nearly all major graph libraries support reading in edge list format, though you can definitely develop an approach that doesn't use such libraries.

The remaining files are `labeled-vertices.train.tsv`, `labeled-vertices.dev.tsv`, and `unlabeled-vertices.test.tsv.` These first two files contain examples of users with attributes, formatted as:

```
vertex1 T1:3 T7:1 T4:2
vertex2 T2:4
vertex3 T4:3 T3:1
```

Each attribute is specified as `AttributeType:Value`, where X and Y vary. You can think of each attribute type as specific identity attribute (e.g., movie genre preference) and its value as what that attribute is set to (e.g., "comedy" or "noir"). To better simulate real-world conditions, not all users will have their attributes listed (since not everyone reveals everything about themselves). However, the majority of users should have at least two attributes set. Having an attribute is a definitive sign the user has that attribute, while not having a value for attribute type only indicate they have not expressed that attribute (to draw an analogy from machine learning, the data has high precision, but less-than-perfect recall.)

The `unlabeled-vertices.test.txt` file simply has a list of vertices for which you should predict attributes and their values.

What To Do and Grading

This homework has four gradable components:

1. (25%) Building a software system that predicts the missing edges and node attributes
2. (25%) How well your system does at predicting missing edges
3. (25%) How well your system does at predicting attributes for unlabeled nodes
4. (25%) A written submission that includes the analyses described below

There are two tracks for how you can approach the overall homework, each with their own weighting on these three components.

For building your system, you can use existing libraries to predict edges and attributes. **Important Note:** If you use libraries, you need to list which you used and if you use example web pages that provide some code snippets/examples, these too need to be listed. Not all libraries and methods will scale to the dataset size in its original format. You will need to experiment to figure out how best predict given the available resources.

Written Component

All submissions, regardless of which Option chosen, need to include a written component that covers the following items:

- **Your details:**
  - What is your username on Kaggle and what is your uniqname (if we can't link you to your submission, we have to give you a 0 😥).
  - Which libraries you used (if any)
  - Which code snippets you used (if any)
- **Your Approach:**
  - The description of your final submission's methods and discussion of why you decided on these methods. What algorithm did you implement? What parameters did you use? The description is intended to be human-readable such that another person in class could read it and feasibly understand how you implemented your system and *why* you made the choices you did. Learning to justify your design and choices is a critical part of the discussion and something you will need to do as a practicing data scientist. Your description can include some pseudocode but should not include lots of code.
  - What other approaches you tried and how well they did, presented in a Table. You can include a short description of each if you want to comment. Different parameter variations should count in this table. Your work here shows effort and will be counted towards the overall grade.
- **Your Analyses:**

   - How much does homophily hold in this dataset? Define a metric to measure how similar are two user's attributes and report on the distribution of how similar all (or a large sample) of users are.
   - Which attribute types are hard to predict? Using the development data, perform an error analysis and report your expected error rate for each type.

Evaluation

For link prediction, we treat this as a ranking task where you rank the most likely edges for the whole network. The network that was provided had 50,000 edges removed. Here, we'll measure performance in terms of precision@50,000, where calculates how many of your system's predicted edges are within the 50,000. Note that this is a hard problem and it's not expected that any system would have perfect performance!

For attribute prediction, we treat this as a multilabel classification problem and use the Mean F1 score over all attributes.

Each task uses a separate Kaggle InClass submission (since they are different tasks).

When you upload your predictions, you'll receive a score for a publicly-visible 10% of the test data that determines the submission's ranking. The instructors will see how well you do on the hidden 90% of the test data, which will determine the final grade.

Submission Formats and Procedure

Note the submission for edge information has changed in Version 1.1:
For link prediction, you will submit your predictions as a .txt file to Canvas. Your file should contain the 50,000 edges your system thinks are most likely to be in the network. The file should contain one edge per line formatted as
`vertex1 vertex2`
Edges should be sorted with the most probable edges first.

Your Kaggle InClass submission file for **attribute prediction** should be **a csv file** with two columns: `id` and `attr`. The `attr` column should contain a space-delimited list of the attributes you think the user with that `id` has. Your submission file should have the following format:

```
id, attr
123, T0:0 T1:1
```

Academic Honesty Policy

Unless otherwise specified in an assignment all submitted work must be your own, original work. Any excerpts, statements, or phrases from the work of others must be clearly identified as a quotation, and a proper citation provided. Any violation of the University's policies on Academic and Professional Integrity may result in serious penalties, which might range from failing an assignment, to failing a course, to being expelled from the program. Violations of academic and professional integrity will be reported to Student Affairs. Consequences impacting assignment or course grades are determined by the faculty instructor; additional sanctions may be imposed.

Submission Procedure

Your assignment is complete when you have submitted

- at least one file of valid predictions to the **node attribute** task in Kaggle InClass
- **Uploaded your edge predictions file to Canvas.** This should be a file with the 50,000 most probable edges in the graph, as predicted by your system (note this was formerly on Kaggle InClass, but now is on Canvas)
- uploaded your code for **both** systems as a .zip or .tar.gz to Canvas
- uploaded a PDF copy of your written report to Canvas

All of these must be submitted by the deadline or the submission will be counted late. You have three total late days to use for this project and submissions received after that time will not be graded.

Notes and Hints

- This dataset is big! Although the working with this network may seem challenging at first, learning to overcome this challenge is a main learning objective--and importantly, is a skill you'll face in real-world scenarios when given large datasets that you'll need to mine. The objective is to learn how to make sense of them. A few ideas:
    - When struggling to get a system running, try using less data. You can still make predictions with 1% of the data, though they won't be as good. Then scale up when you find (and fix) the bottleneck.
    - Think about which data you need. A random sample of users or edges might work, but what kinds of problems might that lead to? Is there a different way you could sample or select parts of the network to work with?
- Some attributes could be correlated (e.g., if you like metal music, your hair might be more likely to be a mullet). How could you use this to your advantage? How would you learn these associations?

- Since friends often have similar attributes, how might this help you predict edges? Homophily often extends beyond just those people a person is friends with. Could you combine this fact with the network structure somehow to improve predictions?
- Is there a way you could represent each node so that it is predictive of both its attributes and which edges it has? (Hint: think about recommender systems)
- You can solve these problems using many of the different representations we've talked about (item sets, matrices, sequences, etc.). However, not all will be computationally efficient on this type of data, nor will all perform as well. Start with *simple* techniques designed for networks and work from there.
- You're allowed to submit as many times as you want, so feel free to try out various ideas to see what works best.
- Remember: this project is feasible! If you're having trouble working with the data seems to crash your system, take a deep breath and see if there's another way to represent or work with the data that might be more efficient. If you're stuck, the instructors are always here to answer questions and help strategize!