ECE227 Project Guidelines

Spring 2024

1 Introduction

This document is prepared so that you can get an idea of what kind of projects are within the scope of this class. Using these guidelines as an inspiration, each group is supposed to define their own project. These notes only give you a head start and we expect you to define your own questions and make additional investigative steps.

2 Network Analysis and Visualization using NetworkX and Gephi

- Download the Collaboration (Erdös), Facebook and Enron networks.
- Analyze the nodes centrality in each graph using different notions of centrality (e.g. degree centrality, betweenness centrality, eigenvector centrality etc.). What are the most central nodes in each graph? Why might that be the case?
- How many nodes are among the top 10% in terms of degree centrality AND betweenness centrality? How does this overlap change for each of these networks?
- Perform community detection on these graphs. Investigate different community detection algorithms (e.g. "Louvain algorithm.")Experiment with bounded and unbounded approaches. Visualize results.
- Analyze the communities formed as a result of the community detection algorithms. Why do certain nodes belong to the same community? For example, for the Erdös graph, one can analyze the top 5 authors with highest degree from the largest communities and check their research areas to see if they indeed have shared research interests.
- Analyze the degree distribution of these graphs. Do they follow a power law or a Poisson degree distribution? Why or why not?

- Calculate the diameter and the average shortest path for these graphs. What can you notice?

- In terms of similarity and performance, how would you compare these graphs to Barabasi-Albert, Watts-Strogatz and Erdös-Renyi graphs? Justify your reasoning.

3 SEIR-Based COVID-19 Simulations

SEIR stands for Susceptible-Exposed-Infectious-Recovered/deceased and it is a dynamical network model that is used to create simulations and projections for pandemics like COVID-19. `https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology`. In a given population, an individual at a given time is in one of the four states of the SEIR model: Susceptible (S), Exposed (E), Infectious (I), Recovered (R). Each state comes with self-explanatory assumptions, such as a person in state S is healthy but does not have immunity against COVID-19, or a person in state R has a lower probability of re-infection. The goal of the SEIR model is to model the movement of individuals through these states at each time period, given a set of parameters.

The SEIR model uses a discrete-time state machine where each step is a day in the simulation. The state transitions are determined based on probability distributions. For example, an individual in state S has a probability of transitioning to state E during a day, according to a probability distribution. The same applies for each possible state transition. The probability distributions are convolved with the existing cases to estimate the new infections and new deaths for the day, where the number of infections is scaled by R0 (Basic reproduction number) and the number of deaths is scaled by the mortality rate of COVID-19.

The model uses various parameters. Fixed parameters are related to the properties of the COVID-19 virus and they are not likely to vary much across different countries or time frames. These parameters include latency period, infectious period, time between illness onset to hospitalization, time between illness onset to death, hospital stay time and time to recovery. Variable parameters are subject to change according to the country that the simulation is being run on. These parameters include R0 (Basic reproduction number), mortality rate, imports of positive cases per day, mitigation effects (i.e. post-mitigation R), lockdown fatigue factor, lifting of shelter-at-home orders (i.e. post-reopening R) and population. Determining optimal values for the variable parameters is the key to achieve successful projections with the SEIR model, which is where machine learning comes into play, to help with parameter search. The projections published on `https://covid19-projections.com/` uses an open-source SEIR model and it was one of the very successful projection websites for COVID19. You can download (clone) the simulator from here `https://github.com/youyanggu/yyg-seir-simulator` and upload the necessary files to datahub or run it on your local computer.

- Using this simulation model whose parameters are last updated in Oc-

tober 2020, predict the total number of deaths by the end of 2020 and compare the result with the actual number of deaths in the US (you can use John Hopkins website for the reference: `https://coronavirus.jhu.edu/us-map`). Is the model over-estimating or under-estimating the number of deaths? What could be the reason that the model is projecting the number of infections and deaths differently?

- If the initial R0 of the model was increased by 20% how many more deaths would have occurred compared to the model's original prediction in the US?
- If the opening date in the US was increased by 10 days or decreased by 10 days how would the death toll change compared to the model's original prediction in the US?
- According to this model, how many fewer deaths would have occurred if the social distancing in the US had started just 2 days earlier? How about 4, 7, and 10 days?
- Can you modify the SEIR model to incorporate vaccinations? The effect of vaccinations could be incorporated into the model by changing the R0 of the disease to some $R_{effective}$ for the vaccinated population [1]. Vaccination data is also available on `https://coronavirus.jhu.edu/us-map` and various projection methods could be used if we assume that we are running the model on a certain date in the past such as June 2021.

4 Tonnetz Graph

In this project we explore the Tonnetz Graph (`https://en.wikipedia.org/wiki/Tonnetz`). The Tonnetz Graph is simply a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, equipped with a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E}$, enconding relationships in a musical composition. In the literature, there are several interpretations, in some use cases the set of vertices $\mathcal{V}$ represents notes of a melody, and in other cases the vertices represents a set of chords. The set of edges $\mathcal{E}$ encode a relationship between notes/ chords. Generally, the set of edges $\mathcal{E}$ represent *tonal* distances, but they can also encode a diferent meaning depending on the use case. Given a melody, of some length T, we will extend the melody by predicting the most optimal note at the next time step by relying on the Tonnetz Graph.

- Construct a Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Given the chord A_m(A minor, for example), which consists of the following notes: A, B, C, D, E, F, G. We have that $\mathcal{V} = \{A, B, C, D, E, F, G\}$. The set of edges $\mathcal{E}$, encodes some relationship between $\mathcal{V}_i$ and $\mathcal{V}_j$ for $i \neq j$, for $\mathcal{V}_i, \mathcal{V}_j \in \mathcal{V}$. The objective of this project is to explore the Tonnetz graph to analyze melodies and to generate new melodies using AI methods as well as network properties.
- Given the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, identify the most "central" node in the graph by using one of the following methods:

  - The more connected node on average.
  - "Betweeness" centrality
  - Eigen-vector centrality

- Since the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ gives rise to an Adjacency matrix A, maybe initialize transitional probabilities via some random initialization scheme. Methods to explore (*Xavier* initialization, Zero-mean and Identity covariance initialization from a *Gaussian*, Uniform initialization). These are inspired from Deep Learning. We can explore MIDI Data-Sets and use melodies from there to learn the matrix A.

- Given some melody of length T, extend the melody further by generating a new note at a time step $\tau + 1$ by relying on the Tonnetz Graph and not worry too much about the melody sounding good. Generate new note based on "proximity" to the most central node in the graph.

There are several ways of approaching this problem when it comes to data. You can start from raw audio and convert the raw audio into discrete tokens which can later be useful for note representation. A MIDI Data Set can also be an alternative:

- Lakh MIDI Data Set: `https://colinraffel.com/projects/lmd/`

- Audio:
`https://towardsdatascience.com/40-open-source-audio-datasets-for-ml-59dc39d48f06`

Additional references that can be useful for this project are:

- `https://guichaoua.gitlab.io/web-hexachord/`

- `https://morenoandreatta.com/software/`

5 Cooperation of Multi-Agent systems

Consider a set of N agents or point robots positioned in Euclidean Space ($\mathbb{R}^2$). Denote the set of agents as $\mathcal{V} \in G(\mathcal{V}, \mathcal{E})$. The set of vertices $\mathcal{V}$ identify a position $(x, y) \in \mathbb{R}^2$ and possibly orientation θ. The set of edges $\mathcal{E}$ encode a relationship between the N agents ($|\mathcal{V}| = N$). The goal here is to guide a differential - drive robot to the most "central" agent in $\mathcal{G}$ by exchange of information between other agents. Denote the differential - drive robot agent to guide as $\mathcal{V}_0$ and the central agent as $\mathcal{V}_C$. The set of weights in $\mathcal{E}$ can represent a level of trust to the information that $\mathcal{V}_0$ receives from the other agents in $\mathcal{V}$. For example, you can define a set of weights $w_{(i,j)}$, where $w_{(i,j)} \in [0, 1]$. Here 0 can represent a "poor level of trust", and 1 can represent a "high level of trust". The agent $\mathcal{V}_0$ needs to move according to a precise motion model. A good start is to use the $Euler - Discretization$ model, which is given by:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t) = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \boldsymbol{x}_t + \tau_t \begin{pmatrix} v_t cos(\theta_t) \\ v_t sin(\theta_t) \\ \omega_t \end{pmatrix} \tag{1}$$

Where (x_t, y_t) denotes the position of the robot at time t and θ_t denotes the orientation of the robot at time t. The linear velocity of the robot at time t is given by v_t and it is obtained from the Wheel Encoders. The angular velocity ω_t, on the other hand, is obtained from IMU measurements. Finally τ_t represents the time difference between two consecutive time stamps.

You can explore different types of Data-Sets which provide IMU data and Wheel Encoder data and other type of data. A good start could be to explore the Data-Set at `https://www.uma.es/robotics-and-mechatronics/info/132852/negs-ugv-dataset`. Use NetworkX when creating your graph and make sure to use it for this problem `https://networkx.org/`. An important note is that the motion model defined in (1) relies on the type of robot that collected the data which is referenced in the paper attached. If you decide to use a different Data-Set, whose robot that collected the data has a different motion model, then it is oppurtune to modify the motion model in (1).

The initial setting is that given $\mathcal{V}_0$ and $\mathcal{V}_C$ with (x, y) locations in $\mathbb{R}^2$, we have a graph $\mathcal{G}$ with external point agents as a set of vertices connected to $\mathcal{V}_0$. The weights on the edges, encode a level of trust, or it can encode something else that can be meaningful in guiding $\mathcal{V}_0$ towards $\mathcal{V}_C$. Given the setting above do the following:

- How can you model the following problem as a graph problem? In other words, how can you use the "trust weights" (which arise from an adjacency matrix for $\mathcal{G}$) in an un-directed graph such that the differential-drive robot can move according to the motion model in (1). The information that the agents in $\mathcal{V}$ can provide is orientation information, linear velocity information, direction of movement etc.

- What is the most "central" node in the graph problem you have formulated?

- The robot needs to move over T time stamps. At time $t = 0$ the robot needs to take in information from the agents in $\mathcal{V}$ such that it moves to a new location at time $t = 1$. How can you formulate this as an optimization problem over the graph $\mathcal{G}$ by learning the weights $w_{(i,j)}$ or via another method (this obviously depends on how you formulated this as a graph problem).

- Make sure that you plot the trajectory of the robot, by clearly indicating the starting point and end point.

- At the end of the robot trajectory visualize the graph with final weights.

The following references for the modeling of this problem can be useful:

- Paper on the Data-Set suggested above : `https://www.mdpi.com/1424-8220/22/15/5599`
- Poses in Robotics: `https://natanaso.github.io/ece276a/ref/ECE276A_3_Rotations.pdf`
- Robot motion and Observation models : `https://natanaso.github.io/ece276a/ref/ECE276A_4_MotionAndObservationModels.pdf`
- Gradient Descent using Numpy: `https://stackoverflow.com/questions/17784587/gradient-descent-using-python-and-numpy`
- Projected Gradient descent algorithm : `https://angms.science/doc/CVX/CVX_PGD.pdf` (this can be useful if you decide to learn the weights and making sure that they satisfy some constraint)

6 Network models of racial segregaton

Census data:
https://www.washingtonpost.com/graphics/2018/national/segregation-us-cities/
The Schelling model:
https://en.wikipedia.org/wiki/Schelling
http://nifty.stanford.edu/2014/mccown-schelling-model-segregation/
https://ncase.me/polygons/
https://www.sciencedirect.com/science/article/pii/S0264275124000520
Research alternative models, propose policies for integration, explore the issue of maintaining cultural diversity, compare with data.

References

[1] Phitchayapak Wintachai and Kiattisak Prathom. "Stability analysis of SEIR model related to efficiency of vaccines for COVID-19 situation". In: *Heliyon* 7.4 (2021).