Large Scale Social and Complex Network: Design and Algorithms
ECE 232E Spring 2024
Prof. Vwani Roychowdhury
UCLA, Department of ECE

Project 4
Graph Algorithms

Due on Monday, June 10, 2024 by 11:59 PM PST

Introduction

In this project we will explore graph theory theorems and algorithms, by applying them on real data. In the first part of the project, we consider a particular graph modeling correlations between stock price time series. In the second part, we analyse traffic data on a dataset provided by Uber. Third part of the project asks you to define your own task.

Fourth part of the project is related to shortest paths, and is UNGRADED, OPTIONAL. However, it is suggested that you complete it before finishing part, especially if you struggle with the runtimes for questions 19-24.

1. Stock Market

In this part of the project, we study data from stock market. The data is available on this Dropbox Link. The goal of this part is to study correlation structures among fluctuation patterns of stock prices using tools from graph theory. The intuition is that investors will have similar strategies of investment for stocks that are effected by the same economic factors. For example, the stocks belonging to the transportation sector may have different absolute prices, but if for example fuel prices change or are expected to change significantly in the near future, then you would expect the investors to buy or sell all stocks similarly and maximize their returns. Towards that goal, we construct different graphs based on similarities among the time series of returns on different stocks at different time scales (day vs a week). Then, we study properties of such graphs. The data is obtained from Yahoo Finance website for 3 years. You're provided with a number of csv tables, each containing several fields: Date, Open, High, Low, Close, Volume, and Adj Close price. The files are named according to *Ticker Symbol* of each stock. You may find the market sector for each company in `Name_sector.csv`. **We recommend doing this part of the project (Q1 - Q8) in R**.

1. Return correlation

In this part of the project, we will compute the correlation among log-normalized stock-return time series data. Before giving the expression for correlation, we introduce the following notation:

- $p_i(t)$ is the closing price of stock i at the t^{th} day

- $q_i(t)$ is the return of stock i over a period of $[t-1, t]$

$$q_i(t) = \frac{p_i(t) - p_i(t-1)}{p_i(t-1)}$$

- $r_i(t)$ is the log-normalized return stock i over a period of $[t-1, t]$

$$r_i(t) = \log(1 + q_i(t))$$

Then with the above notation, we define the correlation between the log-normalized stock-return time series data of stocks i and j as

$$\rho_{ij} = \frac{\langle r_i(t) r_j(t) \rangle - \langle r_i(t) \rangle \langle r_j(t) \rangle}{\sqrt{(\langle r_i(t)^2 \rangle - \langle r_i(t) \rangle^2)(\langle r_j(t)^2 \rangle - \langle r_j(t) \rangle^2)}}$$

where $\langle \cdot \rangle$ is a temporal average on the investigated time regime (for our data set it is over 3 years).

QUESTION 1: What are upper and lower bounds on ρ_{ij}? Provide a justification for using log-normalized return ($r_i(t)$) instead of regular return ($q_i(t)$).

2. Constructing correlation graphs

In this part,we construct a correlation graph using the correlation coefficient computed in the previous section. The correlation graph has the stocks as the nodes and the edge weights are given by the following expression

$$w_{ij} = \sqrt{2(1 - \rho_{ij})}$$

Compute the edge weights using the above expression and construct the correlation graph.

QUESTION 2: Plot a histogram showing the un-normalized distribution of edge weights.

3. Minimum spanning tree (MST)

In this part of the project, we will extract the MST of the correlation graph and interpret it.

QUESTION 3: Extract the MST of the correlation graph. Each stock can be categorized into a sector, which can be found in `Name_sector.csv` file. Plot the MST and color-code the nodes based on sectors. Do you see any pattern in the MST? The structures that you find in MST are called Vine clusters. Provide a detailed explanation about the pattern you observe.

QUESTION 4: Run a community detection algorithm (for example walktrap) on the MST obtained above. Plot the communities formed. Compute the homogeneity and completeness of the clustering. (you can use the 'clevr' library in r to compute homogeneity and completeness).

4. Sector clustering in MST's

In this part, we want to predict the market sector of an unknown stock. We will explore two methods for performing the task. In order to evaluate the performance of the methods we define the following metric

$$\alpha = \frac{1}{|V|} \sum_{v_i \in V} P(v_i \in S_i)$$

where S_i is the sector of node i. Define

$$P(v_i \in S_i) = \frac{|Q_i|}{|N_i|}$$

where Q_i is the set of neighbors of node i that belong to the same sector as node i and N_i is the set of neighbors of node i. Compare α with the case where

$$P(v_i \in S_i) = \frac{|S_i|}{|V|}$$

QUESTION 5: Report the value of α for the above two cases and provide an interpretation for the difference.

5. Correlation graphs for weekly data

In the previous parts, we constructed the correlation graph based on daily data. In this part of the project, we will construct a correlation graph based on WEEKLY data. To create the graph, sample the stock data weekly on Mondays and then calculate ρ_{ij} using the sampled data. If there is a holiday on a Monday, we ignore that week. Create the correlation graph based on weekly data.

QUESTION 6: Repeat questions 2,3,4,5 on the WEEKLY data.

6. Correlation graphs for MONTHLY data

In this part of the project, we will construct a correlation graph based on MONTHLY data. To create the graph, sample the stock data Monthly on 15th and then calculate ρ_{ij} using the sampled data. If there is a holiday on the 15th, we ignore that month. Create the correlation graph based on MONTHLY data.

QUESTION 7: Repeat questions 2,3,4,5 on the MONTHLY data.

QUESTION 8: Compare and analyze all the results of daily data vs weekly data vs monthly data. What trends do you find? What changes? What remains similar? Give reason for your observations. Which granularity gives the best results when predicting the sector of an unknown stock and why?

2. Let's Help Santa!

Companies like Google and Uber have a vast amount of statistics about transportation dynamics. Santa has decided to use network theory to facilitate his gift delivery for the next Christmas. When we learned about his decision, we designed this part of the project to help him. We will send him your results for this part!

1. Download the Data

Go to "Uber Movement" website and download data of **Travel Times by Month (All Days), 2019 Quarter 4**, for Los Angeles area[1]. The dataset contains pairwise traveling time statistics between most pairs of points in the Los Angeles area. Points on the map are represented by unique IDs. To understand the correspondence between map IDs and areas, download **Geo Boundaries** file from the same website[2]. This file contains latitudes and longitudes of the corners of the polygons circumscribing each area. To be specific, if an area is represented by a polygon with 5 corners, then you have a 5×2 matrix of the latitudes and longitudes, each row of which represents latitude and longitude of one corner. **We recommend doing this part of the project (Q9 - Q18) in Python**.

2. Build Your Graph

Read the dataset at hand, and build a graph in which nodes correspond to locations, and undirected weighted edges correspond to the mean traveling times between each pair of locations (**only December**). Add the centroid coordinates of each polygon region (a 2-D vector) as an attribute to the corresponding vertex.

The graph will contain some isolated nodes (extra nodes existing in the Geo Boundaries JSON file) and a few small connected components. Remove such nodes and just keep the largest connected component of the graph. In addition, merge duplicate edges by averaging their weights [3]. We will refer to this cleaned graph as G afterwards.

QUESTION 9: Report the number of nodes and edges in G.

3. Traveling Salesman Problem

QUESTION 10: Build a minimum spanning tree (MST) of graph G. Report the street addresses near the two endpoints (the centroid locations) of a few edges. Are the results intuitive?

QUESTION 11: Determine what percentage of triangles in the graph (sets of 3 points on the map) satisfy the triangle inequality. You do not need to inspect all triangles, you can just estimate by random sampling of 1000 triangles.

Now, we want to find an approximation solution for the traveling salesman problem (TSP) on G. Apply the 1-approximate algorithm described in the class[4]. Inspect the sequence of street addresses visited on the map and see if the results are intuitive.

QUESTION 12: Find an upper bound on the empirical performance of the approximate algorithm:

$$\rho = \frac{\text{Approximate TSP Cost}}{\text{Optimal TSP Cost}}$$

QUESTION 13: Plot the trajectory that Santa has to travel!

[1] If you download the dataset correctly, it should be named as `los_angeles-censustracts-2019-4-All-MonthlyAggregate.csv`

[2] The file should be named `los_angeles_censustracts.json`

[3] Duplicate edges may exist when the dataset provides you with the statistic of a road in both directions. We remove duplicate edges for the sake of simplicity.

[4] You can find the algorithm in: Papadimitriou and Steiglitz, *"Combinatorial optimization: algorithms and complexity"*, Chapter 17, page 414

4. Analysing Traffic Flow

Next December, there is going to be a large group of visitors travelling between a location near Malibu to a location near Long Beach. We would like to analyse the maximum traffic that can flow between the two locations.

5. Estimate the Roads

We want to estimate the map of roads without using actual road datasets. Educate yourself about *Delaunay triangulation* algorithm and then apply it to the nodes coordinates[5].

QUESTION 14: Plot the road mesh that you obtain and explain the result. Create a graph G_Δ whose nodes are different locations and its edges are produced by triangulation.

6. Calculate Road Traffic Flows

QUESTION 15: Using simple math, calculate the traffic flow for each road in terms of cars/hour. Report your derivation.

Hint: Consider the following assumptions:

- Each degree of latitude and longitude $\approx$ 69 miles
- Car length $\approx$ 5 m = 0.003 mile
- Cars maintain a safety distance of 2 seconds to the next car
- Each road has 2 lanes in each direction

Assuming no traffic jam, consider the calculated traffic flow as the max capacity of each road.

7. Calculate Max Flow

Consider the following locations in terms of latitude and longitude:

- Source coordinates (in Malibu): [34.04, -118.56]
- Destination coordinates (in Long Beach): [33.77, -118.18]

QUESTION 16: Calculate the maximum number of cars that can commute per hour from Malibu to Long Beach. Also calculate the number of edge-disjoint paths between the two spots. Does the number of edge-disjoint paths match what you see on your road map?

[5] You can use `scipy.spatial.Delaunay` in `Python`, or `RTriangle` package in `R`.

8. Prune Your Graph

In G_Δ, there are a number of unreal roads that could be removed. For instance, you might notice some unreal links along the concavities of the beach, as well as in the hills of Topanga. Apply a threshold on the travel time of the roads in G_Δ to remove the fake edges. Call the resulting graph $\tilde{G}_\Delta$.

QUESTION 17: Plot $\tilde{G}_\Delta$ on actual coordinates. Do you think the thresholding method worked?

QUESTION 18: Now, repeat question 16 for $\tilde{G}_\Delta$ and report the results. Do you see any changes? Why?

9. Construct New Roads

Let us assume that the output of question 17 (graph $\tilde{G}_\Delta$) is the actual road network of LA. The government of LA wants to construct 20 new roads (each road is an edge). They ask you to help them decide where to construct the roads. In other words, you have to create 20 new edges in the graph.

QUESTION 19: Strategy 1 (geo_distance, static): Reduce the maximum extra travelling distance. The extra travelling distance between 2 locations is the difference of the shortest traveling distance and the straight line distance. i.e
extra_distance(v,s) = distance_of_shortest_path(v,s) - euclidean_distance(v,s)

Use the coordinates of v and s to get the euclidean_distance between them. Calculate the extra_distance between all pairs of points. The top 20 pairs with highest extra_distance will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy?

QUESTION 20: Strategy 2 (geo_distance, static, with_frequency): In strategy1, we are using the geographical distance between points to decide which new road to create. However, in the real world, some pairs are more in demand than other pairs. Assume that you know the frequency of travel between every pair. Use the frequency as the weight to multiply the difference. i.e

weighted_extra_distance(v,s) = extra_distance(v,s) * frequency(v,s)

where frequency(v,s) is a random integer between [1,1000]. Use the coordinates of v and s to get the euclidean_distance between them. Calculate the weighted_extra_distance between all pairs of points. The top 20 pairs with highest weighted_extra_distance will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy?

QUESTION 21: Strategy 3 (geo_distance, dynamic): In the above strategy, we are creating all roads at the same time. This time, we will create the roads one by one.

Repeat 20 times: i) Compute extra_distance between all the pairs in the graph.
ii) Create a road between the pair with highest extra_distance and update the graph.
iii) Print the source and destination of this new edge.

Report the source and destination of the 20 new edges. plot the final graph on actual coordinates. What is the time complexity of the strategy?

QUESTION 22: Strategy 4 (Travel time, static): We want to optimize to reduce the maximum extra travelling time. The extra travelling distance between 2 locations is the difference of the shortest traveling time and the straight line travel time. i.e

extra_time(v,s) = travel_time_of_shortest_path(v,s) - euclidean_distance(v,s)/travel_speed(v,s)

travel_speed(v,s) = distance_of_shortest_path(v,s) / travel_time_of_shortest_path(v,s)

Use the coordinates of v and s to get the euclidean_distance between them. Calculate the extra_time between all pairs of points. The top 20 pairs with highest extra_time will be the new edges we suggest. Print the source and destination of these pairs and write them in your report. Create these new edges in the graph, and plot the resultant graph on actual coordinates. What is the time complexity of the strategy? What is the time complexity of the strategy?

QUESTION 23: Strategy 5 (Travel time, dynamic): Similar to strategy 3, this time we will create roads one by one while optimizing the travel time.

Repeat 20 times:
i) Compute extra_time between all the pairs in the graph.
ii) Create a road between the pair with highest extra_time and update the graph.
iii) Print the source and destination of this new edge.

Report the source and destination of the 20 new edges. plot the final graph on actual coordinates. What is the time complexity of the strategy?

QUESTION 24: Strategy comparison. Compare the following strategies:

a) 1 vs 2 - compare and analyze the results. which is better? why?
b) 1 vs 3 - compare and analyze the results. which is better? why?
c) 1 vs 4 - compare and analyze the results. which is better? why?
d) statics vs dynamic - Considering we want to improve the overall road network by constructing new roads, is either of them the optimal strategy?
i) if yes, which one and why?
ii) if not, what would be a better strategy to construct roads and is it optimal? What is the time complexity?

(assume that you want to reduce travelling distances and you know before hand that we have to construct 20 new roads).

e) Open ended question : Come up with new strategy. You are free to make any assumptions. You don't have to necessarily optimize traveling time or distance, you can come up with any constraints. justify your strategy.

10. Define Your Own Task

Brainstorm with your group, then define and implement a new interesting task based on the dataset at hand. *We recommend you to download and use data of other time periods. The*

duration of this data should be at least three months. For example you may use January 2020 to March 2020 in which the major changes happened to daily's commutes.

This part is open-ended and you'll be graded based on your creativity. You may also wish to go beyond, and use any other navigation- / traffic- related dataset and define your favorite task. This part is worth 20% credit of the project. You may discuss your ideas with the TA in designated office hours.

11. Shortest Paths (UNGRADED)

Write a code to implement Dijkstra's algorithm from scratch. What does the algorithm take as input and what does it output? What is the time complexity?

Write a code to implement Bellman-ford algorithm from scratch. What does the algorithm take as input and what does it output? What is the time complexity?

Write a code to implement floyd-warshall algorithm from scratch. What does the algorithm take as input and what does it output? What is the time complexity?

Submission

Please submit your report to Gradescope. In addition, please submit a zip file containing your codes and report to BruinLearn. The zip file should be named as "Project4_UID1_..._UIDn.zip" where UIDx are student ID numbers of team members. If you have any questions you can post on piazza.