Algo 2 matching, graphs, clustering: project description

NICOLAS LE HIR
nicolaslehir@gmail.com

1 DESCRIPTION OF THE PROJECT

The goal of the project is too analyze and process a dataset of your choice, using methods studied during the course.

1.1 Dataset constraints

You are free to choose the dataset within the following constraints :

— utf-8 encoded in a **data.csv** file

— several hundreds of lines

— a number n with $5 < n < 10$ (approximately) attributes (or columns, or features), the first being a unique id, separated by commas

— some fields must be quantitative (numbers), others categorical (not numbers).

— some fields could be correlated (for instance there is a correlation between temperature and pressure as seen in class)

It's nice if the dataset comes from a real example but you can also generate it, or even build a hybrid dataset mixing real data and generated data. The goal of the project is to apply some methods seen in class.

Example resources to find datasets :

— Link 1

— Link 2

— Link 2

— Link 4

You may **not** use a dataset that we studied during the course.

1.2 Processing

The goal of the processing is to build a distance between datapoints and use it to build a compatibility graph. Then a matching or a clustering should be performed on these data.

The processing must be made with **python 3** with 5 files :

1.2.1 *analyze.py*

Presents a quick analysis of the dataset. For instance :

— Histograms of quantitative variables with a comment on important aspects, such as mean, standard deviation, etc.

— Correlation matrices (maybe not for all variables)

— Any interesting analysis : if you have categorical data, with categories are represented most ? To what extent ?

If the dataset is very large you may also extract a random sample of the dataset to build histogram or compute correlations. You can discuss whether the randomness of the sample has an important influence on the analysis result (this will depend on the dataset).

1.2.2 *build_metric.py*

Generates of a metric (distance, or similarity) that represent the **compatibilities** between the datapoints (build edges between some of them). This allows you to build a compatibility graph.

You have to choose how to build the metric : as we have seen in class, there might be several relevant ways to do it. We saw for instance the euclidean distance, the edit distance, etc. **The important aspect of this part of the project is that you can build your own distance or similarity in order to assess whether datapoints should be considered "similar" or "dissimilar".**

For instance, in order to compute you distance, you may :

— 1) quantify the non-quantitatives variables,

— 2) normalize and/or weight the importance of the different fields,

— 3) remove useless variables,

— 4) combine several components of the dataset in order to compute your custom distance.

Then, using this distance, you might set a threshold and build edges between points that are separated by a distance smaller than the threshold, or between which the similarity is higher than the threshold.

This step allows you to build a graph representing connections between points. Please produce a visualization of the graph or of a part of the graph (this will be probably easier to read).

1.2.3 *(match.py + greedy.py) or (cluster.py + number_of_clusters.py)*

Depending on the dataset and on your personal preference, you can do either a matching **or** a clustering of the dataset. In both cases (matching or clustering), you should do some research in other to find a heuristic that you find interesting and relevant for your dataset.

MATCH.PY + GREEDY.PY **match.py** computes a matching in the created graph (extraction of the greatest possible number of pairs of compatible elements). For instance a maximal or maximum matching. Methods from third-party libraries are not allowed at this step.

In the case of the matching in a bipartite graph, we have seen in the course that there exists a polynomial algorithm to exactly solve the problem.

In the case of a non-bipartite graph, other algorithms exist, such as the blossom algorithm. You may also implement an heuristic of your choice, such as some degree-based heuristic.

The script must save your matching to a file containing the ids of the matched nodes. You may choose the file format (binary file with python object, csv, etc.)

greedy.py performs a simple greedy algorithm as studied in the course. You must use it to assess how much better your heuristic is, compared to the simple greedy algorithm. On top of a comparison between the greedy method and your heuristic, any quantitative analysis of the quality of your heuristic will be appreciated, for instance a dependence on some parameter of the algorithm.

CLUSTER.PY + NUMBER_OF_CLUSTERS.PY **cluster.py** clusters the data. You are free to choose the clustering algorithm. Methods from third-party libraries are not allowed at this step.

Note that a clustering may or may not be based on the edges of a graph. For instance, Spectral Clustering works with edges, unlike the kmeans. This means that if you cluster the data with a method not based on edges, such as kmeans, the graph produced by **build_metric.py** is only used for visualization. Many clustering algorithms exist, you are encouraged to do some research and to use one that you find relevant for your problem.

cluster.py must save your clustering to a file. You may choose the file format (binary file with python object, csv, etc.)

number_of_clusters.py performs a quantitative heuristic of your choice in order to justify the number of clusters used. This quantitative heuristic may be based on the knee criterion that we discussed during the class, but you are encouraged to explore also different heuristics. Any quantitative measure of the output of your method will be appreciated, as well as a quantitative comparison between hyperparameters of the algorithm.

1.2.4 *test.py*

If you did a matching, this file tests if your selection of edges is a correct matching and prints a boolean. In that case, you may use for instance methods of **networkx**.
`https://networkx.org/documentation/latest/reference/algorithms/generated/networkx.algorithms.matching.is_matching.html`

If you did a clustering, this file tests if your clusters form a clustering of the dataset and prints a boolean.

1.2.5 *Comments*

The usage of this dataset and its processing should be justified by a question of your choice. Thus, the approach should be explained and justified in a separate pdf file. The pdf file needs to contain explanations about :

— the nature of the dataset

— explanations about its processing, and in particular the construction of the distance or similarity.

— description of the matching or clustering used.

— comments on the results obtained.

If relevant, some comments on the distribution of the data, for instance by studying correlations between columns, or by analyzing the distribution of a given column, will be appreciated.

Do not hesitate to use networx or graphviz or another tool to illustrate the graph created, or parts of the graph if more relevant.

Please indicate how work was divided between students (each student must have contributions to the repository).

2 ORGANIZATION

Number of students per group : 2.

The deadline for submitting the project is **February 27th 2022**.

The project must be shared through a github repo with contributions from all students, that contains :

— the name of the student(s)

— the csv dataset **data.csv**

— the 5 python 3 files.

Please write "Algo 2 matching session 1" in the subject of your email.

You can reach me by email, I will answer faster if you use the gmail address rather than the Epitech address.

3 EXERCISES DONE DURING THE COURSE

The exercises we made during the class are available with correction here : `https://github.com/nlehir/ALGO2`. The repository contains example functions you can use to create graph images, such as **random_graph.py**. It is not mandatory to use this repo.

4 LIBRARIES

Unless specified, you may use third-party libraries. For instance for the matching/clustering part, you must implement the algorithm yourself. However, if you use libraries, for instance for loading the data or visualizing them, it is required that you present them shorty in your document and describe the functions that you use from the library. These comments on the libraries do not need to be long.