Project Report: EECE2140 COMPUTING FUNDAMENTALS FOR ENGINEERS

Group 7: Marc Jalkh, Maria Olano, Shrishti Thakur
Northeastern University
College of Engineering
Department of Electrical and Computer Engineering
Course Title: EECE 2140: COMPUTING FUNDAMENTALS FOR ENGINEERS
Instructor: Fatema Nafa

March 17, 2024

1 Information

Iteration *02*
Group 7: *KNN (k nearest neighbors)*
Date: *March 17, 2024*

2 Objectives & Deliverables

KNN or K Nearest Neighbors is a machine learning algorithm essential in the domain of AI. It allows to predict the behavior of a defined entity in a framework. This algorithm is used in a multitude of tasks such as regression analysis and classification.

For the purpose of the project, the focus will be predicting the class of a new element whose position is known, that is part of a set of points with given classification and position.

This can be applied to any data type as long as it is classified into a set of points with respective positions. Meaning the characteristics of every point are to be conformed to a set of coordinates for its position.

With the right algorithm, any data type can be implemented with few modifications. They can range from a Cartesian system to a shape and color identification, or sorting individuals into categories, and so on. The amount and scope of implementation are to be determined based on available data and is a flexible matter that can be changed.

Then, the core of the project is the KNN algorithm itself, which is expected to classify a new unknown element into a specific category based on the range of data available to it. The proceedings will be elaborated in the following section.

3 Technologies & Tools

As previously mentioned, the algorithm requires some data as basis of functionality. A data that includes a set of subjects in the same framework, each with classifiable variations of the same attributes. An example could be a classroom of students with different majors, origin country and/or state, hairstyles, clothing styles and so on.

Each attribute's variant has to be classified in order to conform each individual to a set of coordinates, hence positions. There are many ways data sets of this sort can be treated:

1. Reading data from tables such as Excel with the **Python Pandas and CSV tools** and inserting them into Python as a data type, and then automatically classify data into coordinates based on a dictionary of attributes as keys with respective classification as values.

2. Directly insert the data into Python, by manually structuring them into a data type and automatically classify them with the same above method.

3. Manually insert and classify

Note: *The group is opting for the second option, awaiting approval of professor on decision. Again, this is an adjustable feature that would not affect the Objectives & Deliverables of the project.*

4 Plan & Timeline

- 03/17: Iteration 02 report due (HERE NOW)
- 03/18 - 03/22: Code development + Selecting Data type
- 03/24: Iteration 03 report due
- 03/25 - 03/29: Wrapping up code + Start presentation prep
- 03/31: Iteration 04 report due
- 04/01 - 04/04: Final revisions + Practice presentation
- 04/05: Presentation day

Note: *All team members will be working equally on all and same aspects and milestones listed above, regardless of specific skills, expertise, and interests.*

5 Basic Functionalities

5.1 Pseudo-code

1. Evaluate the distance separating a new element x from each and every other point available in the framework. Every point in the framework is assigned an index i.
2. Stock these distances in a data set preferably a nested list of this type $[[d0, p0], [d1, p1], \ldots, [di, pi]]$ where d is the distance separating x from p. It is important to keep track of both distance and coordinate to be able to trace back origin and classification
3. Choose the k nearest neighbors of x based on the distance d
4. Assign a class to the new element x based on the majority of classifications represented between the chosen k points.

Note: *point coordinates stored as tuples, each coordinate representing a classification of each attribute in question. There can be infinitely many attributes.*

5.2 Implications

The above documented procedure implies the use of functions, and the consideration of the object-oriented style of the code:

- A classification method that takes a set of attributes and turns it into coordinates
- A distance method to calculate the distances following basic math formula $\sqrt{(x0 - x1)^2 + (y0 - y1)^2}$ in the case of two attributes and applicable to more
- A method that determines the k nearest neighbors and sorts them from least to greatest distance. This can be done with a bubble sort $k(k - 1)/2$
- A method to trace back the classification of the k nearest neighbors, and identify which is more abundant, which the new element x will be classified with.
  This would assign a classification of the new element x according to the majority of the classes represented among the chosen k points

6 Progress & Next steps

So far, our group has completed all requirements listed in the Iteration 02 prompt, and have started code developments, completing almost half of the listed Implications and Pseudo-code.

The next step is continuing and ideally completing all aspects of the algorithm, and finding a suitable data type, with the approval of professor. This also implies consulting the professor regarding the data analysis method.

7 Git

Link to group repository:
https://github.com/marcjalkh/EECE-KNN-project