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 25, 2024

Information

Iteration *03*
Group 7: *KNN (k nearest neighbors)*
Date: *March 25, 2024*

1 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.

2 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. For this project, the group has agreed on the following prompt.

Arriving in the cantina of the planet Tatooine, Han Solo decides to give Luke Skywalker instructions so that he does not provoke the dangerous aliens upon his arrival. He then spots some characteristics drawing them into a table and asks us to help him provide information to Luke so as not to create problems and therefore be able to define a dangerous alien.

Color	Height	Weight	Eyes	Dangerous
Yellow	Average	Light	Single	No
Yellow	Tall	Normal	Pairs	Yes
Green	Short	Normal	Pairs	Yes
Yellow	Short	Normal	Single	No
Red	Average	Heavy	Single	No
Green	Tall	Heavy	Single	Yes
Green	Average	Heavy	Single	Yes
Yellow	Short	Light	Pairs	Yes

Han Solo's table: Each row represents the attributes of a single alien in the cantina. Here the framework is clearly defined as the aliens inside the cantina with each identified with variations of the same attributes, also possessing the to be determined Dangerous attribute which has been previously determined by Han Solo for the aliens showcased above. If a new alien is to enter the cantina, it would have the apparent attributes such as Color, Height, Weight, and Eyes but the hidden Dangerous characteristic which is to be determined based on this data and the KNN algorithm. More details on this process later.

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. The group has opted for reading the above dataset from a CSV Excel table using Pandas, processing it, and inserting it into a Python datatype. That would require the inclusion of the Pandas library in Python as well as a CSV file containing the table.

3 Basic Functionalities & Pseudo-code

3.1 Setup

The first step of the KNN algorithm is processing the data. If the basis in question is a set of numeric data then, no classification is needed. But in the Tatooine cantina case, the data is a set of non-numeric attributes with non-numeric variations. So, the group has to settle on a classification to turn this data numeric, and generate a set of coordinates each coordinate representing an attribute and its value being the classification of the variation of this attribute. Below is the template used to classify each attribute.

Color	Height	Weight	Eyes	Classification
Yellow	Short	Light	Single	1
Green	Average	Normal	Pairs	2
Red	Tall	Heavy	N/A	3

Note:For example based on the template, the first row alien with attributes Yellow, Average, Light, Single has as position $(1, 2, 1, 1)$.

The classification is implemented as a Python dictionary for subsequent use following this fashion: $"Yellow" : 1, "Green" : 2, "Red" : 3, "Short" : 1, "Average" : 2, "Tall" : 3, "Light" : 1, "Normal" : 2, "Heavy" : 3, "Pairs" : 2, "Single" : 1$
Now that the classification norm has been set, we can proceed with the second data processing step of reading the CSV file.

The Pandas library's built in functions facilitates by swiftly transposing the data as a table from the file, and can be treated as such with its rows and columns using other Pandas functions. This in Python Table or Matrix can be separated and treated into usable lists. So, it would be convenient to separate the known attributes columns (Color,Height,Weight,Eyes) and the to be determined new point's unknown attribute columns (Dangerous).

After that, the latter is to be treated row by row, and then using a loop each attribute is to be classified as numbers (based on above dictionary) into a list, and then with another loop, separated 4 by 4 into tuples creating a list of positions each one representing the classification of the attributes of one alien in the cantina.

Note:*All the above is to be done in main() and is independent of the OOP part of the project, per professor approval.*

Now that a list of points with their coordinates is available, the setup is complete and we can proceed with the main KNN code which will be in Object Oriented style.

But first, a required functionality is a distance calculator function, independent of KNN, that should compute the distances between each set of coordinates and the coordinates of the new point using basic math formula $\sqrt{(x0 - x1)^2 + (y0 - y1)^2}$. The function is designed for two or more coordinates based on a loop, in our case each point has 4 coordinates.

3.2 Object Oriented Main code

The main code or "cantina" class takes several arguments in its initialization that are required to perform the KNN algorithm:

- The new point x being the new alien entering the cantina and its unclassified attributes as a list
- The classification dictionary
- The list of points as tuple of coordinates
- The list of the separated Dangerous attribute

Onto the OOP methods in this order:

1. Classify the attributes of the new point and store them in a tuple.
2. Use this tuple of coordinates and calculate the distance between all points in coord list and the new point.
    (a) Stock these distances in a nested list of this type $[[d0, p0], [d1, p1], \ldots, [di, pi]]$ where d is the distance separating x from p.
    (b) It is important to keep track of both distance and coordinate for subsequent method.

3. Use this nested list to sort it from smallest to greatest distance to new point using a bubble sort $k(k-1)/2$.
    (a) Then cut it based on a user given argument of k which will cut the list at the kth index.
    (b) Keeping the distance associated to its point in the nested list index is convenient as it allows to use the distances to find the closest k neighbors of the new point and maintaining the tuple.
    (c) After cutting the list, the distances can now be discarded as we now have identified the k nearest neighbors and stored them in a list.
4. With narrowed down list based on k proximity scope, trace back the selected nearest point to their own attribute in the to be determined separated list.
    (a) Finding the coordinate index in the original coordinate list used as class argument
    (b) Use this index to find attribute in separated list
    (c) Insert into a list
5. Proceed to voting: The variation with the most counts in list is the one which the new point 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

4 Plan & Timeline

- 03/25: Iteration 03 report due
- 03/25 - 03/29: Start presentation prep
- 03/31: Iteration 04 report due + Individual Git Commits
- 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 Progress & Next steps

In addition to the previous iteration's progress, our group has completed all requirements listed in the Iteration 03 prompt. This includes finishing all the code as well as settling on the final data type of the Tatooine cantina as a CSV file and its processing method using Pandas.

The next step is starting the in-class presentation script and PowerPoint as well as tracking progress with frequent and regular Git Commits.

6 Git

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