

AI & ML Assignment 1

by Rhys Whatty

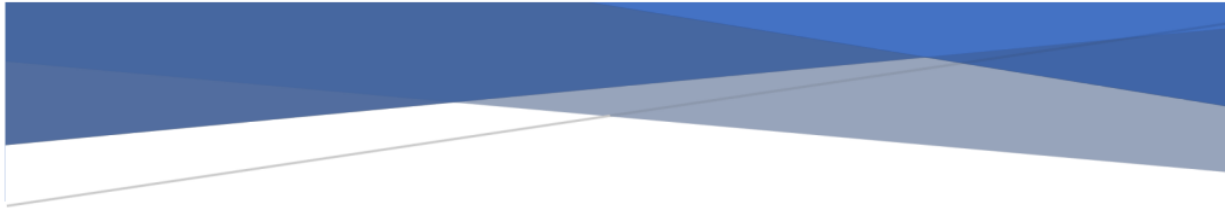
Submission date: 07-Apr-2022 12:26AM (UTC+0100)

Submission ID: 175692785

File name: 17862_Rhys_Whatty_AI__ML_Assignment_1_415582_1858108455.docx (117.88K)

Word count: 2292

Character count: 10405



SEARCH AND ML CLUSTERING

ACCA6011 – AI and Machine Learning

Rhys Whatty
[Email address]

Contents

Introduction 2

Task 1 3

Task 2 5

Task 3 6

Introduction

When dealing with large amounts of data, being able to search through and sort data is incredibly useful to make the handling of said data much more manageable. One such method of sorting large groups of data is through use of k-means clustering, which groups data points into clusters based on proximity to a determined central point known as a centroid.

In this report, the ability to search through data will be explored by creating a piece of code that performs a depth first search to navigate a player agent through a maze in order to obtain collectibles. This code will function through use of a map saved as a text file, and the code will call this text file as a reference. Following that, the report will look into the functionality of K-means algorithms, including their benefits and drawbacks. The algorithm will then be put to practice by using it to organise a series of pre-determined data points into one of two clusters, and then calculating it's new centroid. In this step, the data points will only be clustered once, however K-means algorithms will generally repeat until the clusters to not continue to change clusters based off of the centroids.

Task 1

As part of task 1, a piece of code was developed in which a user can explore a map contained within a text file through use of a depth first search. In order to achieve this, first a map must be provided to the code. The walls of the map are defined by use of asterisks (*), and the code has defined this as such meaning that the agent, who will be “travelling” through the map cannot move into a space occupied by a wall. The map is a 16 square grid and amongst all of the walls and paths, there are a number of objects, represented by letters, that the player can pick up as they travel around. The agent is marked by the letter Q and his starting position is marked on the map text file with the letter S.

In order to display the map, the code have two global variables of Width and Height set at a constant value of 16 and 16, indicating the total area size of the map. Using these values, a nested for loop is utilised counting from 0 to 16, or the respective height and width values, until the entire map is generated, one row at a time. In order to place the agent onto the map, a findStart structure is utilised which places the agent at the specified co-ordinates on the map (which in this instance, is located at the co-ordinate points of (1,1), being the top leftmost corner, as the border wall extends the x and y values of 0).

Within the main structure of the code, the user is offered a list of choices and asks them an input value of 1 to 4. The code then uses if statements to determine the player’s choice. Option 4 ends the program and is available at any time. Option 1 loads a file of choice and is also available at any time. Disregarding the option to end the program, opening the file is the user’s only option as the other options to view the map or to start navigating are unavailable until a file is loaded. Once a file is loaded successfully, if the player chooses to display the map, the display map structure is called, and the map is drawn. Should the player choose the option to navigate, this clears the system display and prints the map again before asking the player to input a direction. A switch statement is used to determine the choice of the player, with each input updating the agent’s position, assuming it is not trying to move into a wall space.

User Guide

Upon bootup, the user is given the option to open up a text file of choice, in which the user can select which file they wish to open. In scenarios where there are multiple maps linked to the same piece of code, this allows the user to load up a different map should they wish. From then, the user may choose from one of the other options or load another map. Their other options are to view the map they have loaded in, to navigate through it, or to end the application. Should the user choose the option to display the map, it will be printed to them in its original form. If they choose to navigate through the map, the map will be generated again with a letter Q indicating the position of the agent. Using the WASD keys (confirming their move each time with Enter), the player can navigate through the maze and obtain the collectibles that are marked as letters. The player can end the navigation at any point by inputting z into the system and will be returned to the options list.

Test Schedule

The code was able to function as intended. Below is a table of the tests conducted in order to locate any bugs in the system.

Test No.	Description of Test	Intended Outcome	Actual Outcome	Functioning?
1	Attempting to display map before loading	Options should be re-offered to the user if they haven't loaded a map in yet.	Options get reoffered	Yes
2	Attempting to navigate a map before loading	Same process, but with the navigate map option	Options get reoffered	Yes
3	Loading in a map of choice	Loads in the map that is typed into the prompt when selecting option 1	Loads the map successfully	Yes
4	Display the map after loading	Renders the entire map to the player in the command line	Successfully displays the map	Yes
5	Begin Navigation	Generates agent in map and allows user to enter direction	Agent is successfully generated	Yes
6	Move Right	Moves the agent a space to the right	Agent moves correctly	Yes
7	Move Left	Moves the agent a space to the left	Agent moves correctly	Yes
8	Move Down	Moves the agent a space down	Agent moves correctly	Yes
9	Move Up	Moves the agent a space up	Agent moves correctly	Yes
10	Move into a wall block	If a movement option that would cause the agent to move into a wall block, the agent will not move	Agent does not move	Yes
11	Exit Navigation	Upon entering the input Z, the player will return to the options	Player returns to the options	Yes

Task 2

A K-means algorithm is a form of clustering algorithm which aims to group large concentrations of data into clusters, with each group containing data as similar to one another as possible.

This algorithm is achieved by first selecting the number of desired K clusters, and then randomly selecting a K point for each centroid. Next, each datapoint will be assigned to their closest centroid. This can be achieved through use of Pythagoras. Once this has been achieved, calculate the mean of the points within each cluster and re-assign the centroid based on the mean distance from every point within the cluster. Repeat the process of calculating the distances and re-assigning the centroid until the centroid stays in the same position.

There are advantages and also drawbacks to this method. K-means algorithms are well suited for handling large numbers of irregular data points that do not possess the means to be differentiated through other methods [1]. It is also a quick and simple method, which lends to its wide usage. It can also regularly lead to good data classification, although this will hinge on knowing the number of datasets in a cluster. [2]

Meanwhile, there are three main drawbacks of k-means algorithms. They are:

- 1) The clusters must be previously known and fixed.
- 2) The results of k-means algorithm depend on initial cluster centres (initial seed-points).
- 3) The algorithm contains the dead-unit problem. [3]

Dead units are problem in concerned underutilization. When calculating centroids, they are always going to be contained within the centre of clusters, meaning that data points that have been placed far away from this rough centre will never be factored into utilisation, rendering a dead unit. [3]

Task 3

Below is a table containing 10 individual Data points with an X and Y co-ordinate. Using Pythagoras theorem, each data point has been determined to whether it is closer to centroid 1, located at (1.80, 4.72), or closer to centroid 2, located at (2.66, 2.14), and has been stated which cluster that it will belong to, based on proximity to its respective centroid.

Data Point x	Data point y	Distance to C1	Distance to C2	Cluster
5	5	3.21	3.69	1
0	5	1.82	3.91	1
2	3	1.73	1.08	2
2	2	2.73	0.67	2
5	1	4.91	2.60	2
3	3	2.10	0.92	2
1	4	1.08	2.49	1
5	3	3.63	2.49	2
1	2	2.84	1.67	2
4	2	3.50	1.35	2

Using the information provided above, the new positions of each centroid can be calculated by taking the co-ordinates of each data point linked to it and calculating its average to provide the co-ordinates of the new centroid.

3 Data points have been clustered to Centroid 1, which were (5,5), (0,5) and (1,4). Taking the averages of the X co-ordinates and the Y co-ordinates individually, the new centroid location calculates out to be located at (2, 4.67).

7 Data points have been clustered to Centroid 2, which were (2,3), (2,2), (5,1), (3,3), (5,3), (1,2) and (4,2). As with Centroid 1, the averages of the X co-ordinates and the Y co-ordinates were calculated to place the new centroid location at (3.14, 2.28).

Bibliography

- [1] Y. e. a. Karaca, "Stroke Subtype Clustering by Multifractal Bayesian Denoising with Fuzzy C Means and K-Means Algorithms," *Scholarly Journal*, 2018.
- [2] F. W. J. J. Chong lun, "K-Means algorithms for clustering analysis with frequency sensitive discrepancy metrics," *Pattern Recognition Letters*, vol. 34, no. 5, pp. 580-586, 2013.
- [3] K. R. Žalik, "An efficient k'-means clustering algorithm," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1385-1391, 2008.

Appendices

Appendix A: Task 1 Code

```
#define WIDTH 16
#define HEIGHT 16
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
struct position {
    int x;
    int y;
};
position current = { 1,1 };
position findStart(void);
char text[WIDTH * HEIGHT];
int loadFile(string);
void printArray(void);

//Main Code that allows players to choose options
int main(void)
{
    int x = 0;
    float a, b, z;
    int fileLoaded = 0;

    string fname;
    while (x < 4)
    {
        //Options for the player to choose
        cout << endl;
        cout << "1:Load a file" << endl;
        cout << "2: Display Map Layout" << endl;
        cout << "3: Navigate Map" << endl;
        cout << "4: End program" << endl;
        cout << "Please enter a choice (1 to 3): ";
        cin >> x;

        //Loading a file
        if (x == 1)
        {
            cout << "Please Enter File name, followed by .txt: ";
            cin >> fname;
            if (!loadFile(fname)) exit(0);
            fileLoaded = 1;
            cout << "File loaded successfully" << endl;
        }

        //Displaying Map Layout
        if (x == 2 && fileLoaded)
        {
            printArray();
        }

        //Navigating Map
        if (x == 3 && fileLoaded)
        {
            char key = 'a';
```

```

        current = findStart();
        cout << "start position is: (" << current.x << " , " << current.y
<< " )" << endl;
        while (key != 'z')
        {
            system("cls");
            printArray();
            cin >> key;

            //User inputs a direction, code calculates if the
            direction is possible, and if it is, makes it
            //switch key tracks which direction was inputted - d =
            right, a = left, s = down, w = up
            switch (key) {
                case 'd':
                    if (text[WIDTH * current.y + (current.x + 1)] !=
                    '**') {
                        text[WIDTH * current.y + (current.x)] = ' ';
                        //overwrite my avatar with a space as I am leaving this position
                        current.x = current.x + 1; // move
                        text[WIDTH * current.y + (current.x)] = 'Q';
                    }
                    break;
                case 'a':
                    if (text[WIDTH * current.y + (current.x - 1)] !=
                    '**') {
                        text[WIDTH * current.y + (current.x)] = ' ';
                        //overwrite my avatar with a space as I am leaving this position
                        current.x = current.x - 1; // move
                        text[WIDTH * current.y + (current.x)] = 'Q';
                    }
                    break;
                case 's':
                    if (text[HEIGHT * (current.y + 1) + current.x] !=
                    '**') {
                        text[HEIGHT * current.y + current.x] = ' ';
                        //overwrite my avatar with a space as I am leaving this position
                        current.y = current.y + 1; // move
                        text[HEIGHT * current.y + current.x] = 'Q';
                    }
                    break;
                case 'w':
                    if (text[HEIGHT * (current.y - 1) + current.x] !=
                    '**') {
                        text[HEIGHT * current.y + current.x] = ' ';
                        //overwrite my avatar with a space as I am leaving this position
                        current.y = current.y - 1; // move
                        text[HEIGHT * current.y + current.x] = 'Q';
                    }
                    break;
            }
        }

        return 0;
    }
}

```

```

//Structure for loading the file from the text menu
int loadFile(string fileName)
{
    string line;
    ifstream myfile(fileName);
    if (myfile.is_open())
    {
        int count = 0;
        while (getline(myfile, line))
        {
            cout << line << endl;

            line.copy(&text[WIDTH * count], WIDTH, 0);
            count++;
        }
        myfile.close();
        return 1;
    }
    else return 0;
}

//Structure for printing the Array file that contains the map into the command line as
a visual display
void printArray(void)
{
    for (int y = 0; y < HEIGHT; y++)
    {
        for (int x = 0; x < WIDTH; x++)
        {
            cout << text[WIDTH * y + x];
        }
        cout << endl;
    }
}

//Structure that determines the start position of the agent
//Changing the values of position changes the start location, regardless of the
location of Marker 'S' on the map
position findStart(void)
{
    return position{ 1,1 };
}

```

AI & ML Assignment 1

GRADEMARK REPORT

FINAL GRADE

53/100

GENERAL COMMENTS

Instructor

DFS code. Implementation attempted. Maze loads and can be manually navigated via keyboard but sadly no dfs in place.

K-means described with some critical evaluation.

First iteration manually calculated and documented. Correct - Well done.

Refs used appropriately.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

RHYS WHATTY

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.