ECM2423 - Coursework exercise

Deadline: 25th March 2021 12:00 (midday, Exeter time)

Lecturer: Dr. Federico Botta (f.botta@exeter.ac.uk)

This continuous assessment (CA) is worth 20% of the overall module assessment. This is an individual exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

Coursework submission The submission of this coursework is via **eBART**. The code needed to answer the questions must be written in Python and must run without errors. Make sure that your code is clear, commented, and that your answers are clearly labelled.

In **eBART**, you should submit a **single compressed (zipped) folder** which contains all the relevant files for this coursework exercise. The compressed folder can contain **either** a PDF file with the answers and your Python scripts, or a Jupyter notebook containing both code and answers if that is what you prefer using.

Marking criteria The three questions you will find below account for 75 marks. Each of the questions is further divided in sub-questions to guide your answers, and specific marks are provided for each of the sub-questions. Additionally, for each sub-question I have clearly defined what the deliverable should be, so that it is clear what you should produce for your answer.

A further 25 marks will be awarded according to the following three criteria:

1. code documentation and comments, with all the relevant information needed on how to run the code (including the names of scripts so that they can be clearly linked to a specific question): **5 marks**
2. clear presentation of your answers: **5 marks**
3. additional excellence, for example, further experimentation, depth of analysis or discussion, or any relevant additional figure: **15 marks**.

The sum of the marks that you can obtain in this coursework exercise cannot exceed 100.

In general, all questions will be marked according to the following criteria:

- For questions which require code (question 1.2 part 3; question 1.3; question 2.2. part 1 and 2; question 3): have you correctly implemented the algorithm? Does your implementation solve the problem as required in the question? Does your code run without errors? Is the code clearly structured?
- For questions requiring a written answer (question 1.1; question 1.2 part 1, 2 and 4; question 1.3; question 2.1; question 2.2 part 3; question 2.3; question 3): is your answer clear, coherent and well-structured? Does it give the correct answer to the correct question? If applicable, have you made the most of possible figures and plots to support your answers, and are those figures and plots appropriately designed and clear? Is your answer succinct and does it only contain information relevant to the question?

Question 1 | Implement a heuristic search algorithm: $A^\star$ and the 8-puzzle game.

The 8-puzzle consists of a 3 x 3 board. Eight tiles in the board are numbered from 1 to 8, and one tile is blank. Any tile adjacent to the blank space can slide into that space. The goal of the game is to reach a given goal configuration from the given initial state. Example initial and goal configurations are given in Figure 1.

In this question, you will be asked to phrase the 8-puzzle problem as a heuristic search and to implement the $A^\star$ algorithm to solve it.

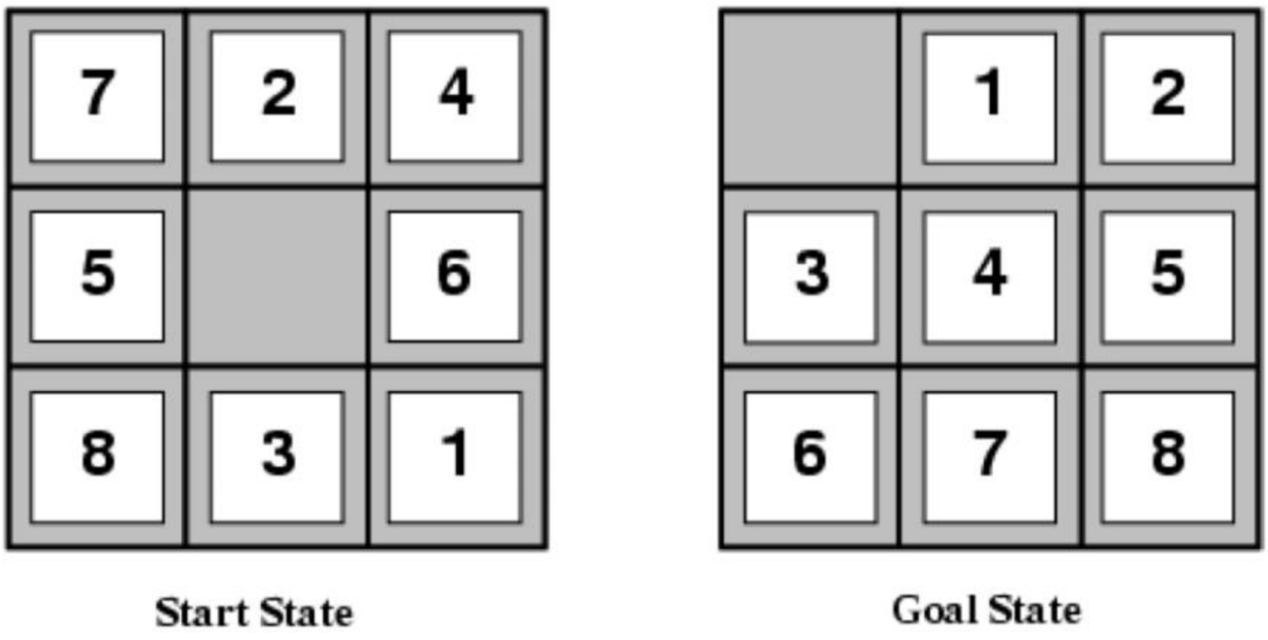

Figure 1: Examples of an initial and goal configuration of the 8-puzzle. Figure taken from Russell, S.J. and Norvig, P., 2016. Artificial intelligence: a modern approach

Question 1.1: Describe how you would frame the 8-puzzle problem as a search problem.

In your own words, write a short description of how the 8-puzzle problem can be seen as a search problem. **(10 marks)**

Deliverable: written answer.

Question 1.2: Solve the 8-puzzle problem using $A^\star$.

This question is further divided in the four parts described next.

1. In this question, you should first briefly outline the $A^\star$ algorithm. **(5 marks)**

   Deliverable: written answer.

2. Then, you should describe **two** admissible heuristic functions for the 8-puzzle problem and explain why they are admissible. Make sure you also explain why you chose these two heuristic functions in particular amongst all the possible ones. **(5 marks)**

   Deliverable: written answer.

3. Then, you should implement **two** versions of the $A^\star$ algorithm in Python to solve the 8-puzzle using the two heuristic functions you have chosen. You can either implement the two versions in the same Python script, letting the user select which one to use before running the code, or you can have two different

scripts if you prefer. To test that it works, you can use the start and goal state of Figure 1 (however, note that this may take a few minutes to run depending on your computer, implementation and choice of heuristic functions), or you can specify your own initial and goal state. If you specify your own initial and goal state, select states which are **at least** five moves apart from each other and **write** these states in your report. You will **not** be penalised if you do not use the start and goal configurations of Figure 1, but you will not receive full marks if the configurations you select are too easy. **(10 marks)**

Deliverable: code and brief written answer if needed to discuss start and goal configuration.

4. Briefly discuss and compare the results given by $A^\star$ when using the two different heuristic functions in question 1.2. **(5 marks)**

   Deliverable: written answer (including figures if appropriate).

Question 1.3: General solution of the 8-puzzle using $A^\star$.

Write a general version of the $A^\star$ algorithm (using either of the two heuristic functions described above) to solve a generic version of the 8-puzzle where the user can input any start and goal state. **(5 marks)**
(Hint: can this be done for any generic pair of configurations...?)

Deliverable: code and brief written answer to discuss whether the code can solve any pair of configurations.

Go to the next page for question 2 of the coursework.

Question 2 | Unsupervised learning: k-means clustering.

This question focuses on the k-means clustering algorithm seen in the lectures.

Question 2.1: Describe k-means clustering and how it applies to the task of hand-written digit recognition.

Briefly outline the main concepts of the k-means clustering algorithm, and then describe how it could be applied to the task of hand-written digit recognition. **(5 marks)**

Deliverable: written answer.

Question 2.2: Using k-means.

You should provide a Python code which uses k-means clustering to recognise hand-written digits. The data for this task is available as part of the `scikit-learn` Python package as follows:

```
from sklearn.datasets import load_digits
digits = load_digits()
```

The dataset consists of 1,797 samples with 64 features. Each digit, originally an image with 8 x 8 pixels, is stored in `digits.data` with 64 features representing the brightness level of each pixel in the image.

This question is further divided in the two parts described next:

1. analyse the data using k-means clustering. **(5 marks)**

   Deliverable: code.

2. For point 1, you can use the built-in Python function for k-means. However, an extra 5 marks will be awarded if you implement your own k-means function. Note: if you provide this implementation, make sure this is clearly referenced in your submission. If I cannot find it, I cannot award the corresponding marks! **(5 marks)**

   Deliverable: code (but also include a brief written answer stating that you have implemented your own version of k-means).

3. present and discuss the results of the k-means clustering on the dataset. For instance, can you visualise and interpret the clusters found? What is the accuracy of the analysis (whilst this is an unsupervised learning exercise, the dataset also contains the labels of each image that you can use to determine the accuracy)? Note that these are only suggested questions to discuss, and you should think about what other interesting questions or results should be discussed. **(5 marks)**

   Deliverable: written answer (including figures if appropriate).

Question 2.3 Limitations of k-means.

Briefly outline what the assumptions and limitations of k-means clustering are. Draw (either manually or digitally) and discuss at least **two** examples of situations in which k-means clustering would not work even if there are very obvious clusters in the data. Why does this happen? **(5 marks)**

Deliverable: written answer, including at least two figures.

Go to the next page for question 3 of the coursework.

Question 3 | Classification of urban areas using decision trees

For this more open-ended question, you will be working with a real-world data set which I have used in my own research. The data set contains information on a variety of aspects of a city, such as the types of building, the types of points of interest (restaurants, cafes, museums, attractions, ...), and other interesting features which can be found in different areas of most cities. An interesting topic in the study of cities is what encourages different people to live in different parts of a city.

In this question, you will focus on trying to understand what features of urban environments can be used to predict which age group is the most present in each area. To do so, I have already extracted and prepared the data for small areas across the city of Rome, Italy. The data was retrieved from the following two sources:

- *OpenStreetMap*: *OpenStreetMap* (OSM) is a voluntary, crowdsourced mapping platform which has been created over the years and which collects a whole range of geographical information about cities and countries worldwide. You can explore it here: `https://www.openstreetmap.org`. Data from OSM is publicly available, and is often used by researchers interested in studying urban environments. In particular, OSM data contains roads, buildings, building types, points of interest (such as cafes, restaurants, museums), parks, and many other features of urban environments
- Italian 2011 census: most countries carry out a national-level census (typically every ten years) to gather data about the population living in the country. Census data contains information on the number of people living in so-called *census sections*, which are small units in which each country is divided into. The data on the population also contains information on the age of people, their income, education levels, and so on.

On the ELE page of this module (ECM2423), you will find the data in a file named `data.csv`. A complete description of the variables included in the data set is provided in Appendix A. Make sure you read this carefully, and do get in touch if you have any questions about the data.

In this question, you should use a decision tree classifier to try and predict the `most_present_age` variable in each of the small areas. You should then answer the following questions:

1. which features are most important for the analysis? Is there any feature that could be omitted? If so, which ones and why? **(5 marks)**

   Deliverable: code for the analysis and written answer (including figures if appropriate).

2. what is the accuracy of your classifier? What happens to the accuracy if you vary the depth of the decision tree? Briefly discuss your results. **(5 marks)**

   Deliverable: code for the analysis and written answer (including figures if appropriate).

Note that this is a good question where you can try and get some excellence marks by going beyond the required analysis and showing independent thinking about a real-world data science and machine learning problem.

Go to the next page for the Appendix with the description of the urban features.

A Urban features description for question 3

In the data file you will find several columns containing each of the different features needed for the analysis. For the purposes of this analysis, the city of Rome is divided into 843 cells, each representing a different area of the city. Cells are unequal in size. Each row in the data file contains data for a different cell. Each column, together with its label in the data file, is as follows:

- `cell_id`: this is simply a label which refers to an identifier for each of the cells in which the city of Rome is divided.
- `Roads:number_intersections`: the density of road intersections
- `Roads:diversity`: an index measuring the diversity of types of roads (major roads, small roads, ...)
- `Roads:total`: the density of roads
- `Buildings:diversity`: an index measuring the diversity of types of buildings (residential, offices, ...)
- `Buildings:total`: the density of buildings
- `LandUse:Mix`: an index measuring the mixture of land usage, such as industrial, residential, natural..
- `TrafficPoints:crossing`: the density of crossing points of roads, such as pedestrian crossings
- `poisAreas:area_park`: the percentage of area in the cell which is covered by a park
- `poisAreas:area_pitch`: the percentage of area in the cell which is covered by a pitch, such as a sports pitch
- `pois:diversity`: an index measuring the diversity of points of interest (POIs), such as bars, museums, attractions or churches
- `pois:total`: the density of points of interest
- `ThirdPlaces:oa_count`: the density of places categorised as *organised activities*
- `Thirdplaces:edt_count`: the density of places categorised as *eating, drinking and talking*
- `ThirdPlaces:out_count`: the density of places categorised as *outdoor places*
- `ThirdPlaces:cv_count`: the density of places categorised as *commercial venues*
- `ThirdPlaces:diverstiy`: an index measuring the diversity of *third places*[1]
- `ThirdPlaces:total`: the density of third places
- `Vertical_density`: the vertical density of buildings, i.e. the average height (in number of floors) of buildings in the cell
- `buildings_age`: the average age of buildings in the cell
- `buildings_age:diversity`: an index measuring the diversity in the age of buildings of the cell
- `most_present_age`: a label corresponding to the age group which has the largest population (according to the census) in the corresponding cell; the possible age groups and labels are:
    1. `age_under_18`: people aged under 18 years old
    2. `age_20_30`: people between 20 and 30 years old

[1]The notion of third places comes from the urban studies literature; broadly speaking, it refers to places which are not home (first place) nor work (second place), and are typically places where people spend their free time to socialise.

3. `age_30_40`: people between 30 and 40 years old
4. `age_40_50`: people between 40 and 50 years old
5. `age_50_60`: people between 50 and 60 years old
6. `age_over_60`: people over the age of 60 years old

Note: some of the features above refer to *densities*. For instance, `Roads:total` is defined as the density of roads. In this case, the density is calculated as the total number of roads divided by the surface area of the corresponding cell in which the roads are. Analogous calculations apply to every feature which is described as a density. This is to ensure that the features correctly account for the fact that cells are different in size. If the values had not been divided by the area, larger cells would naturally result in larger values simply because they are bigger in size.