Assignment 2

COMP9418 – Advanced Topics in Statistical Machine Learning

Lecturer: Gustavo Batista

Last revision: Monday 23$^{\text{rd}}$ October, 2023 at 20:23

Assignment designed by Jeremy Gillen

Edited by Gustavo Batista, Maryam Hashemi and Enver Clark with technical input from William Hales

Instructions

Submission deadline: Sunday, 19th November 2023, at 18:00:00.

Late Submission Policy: The penalty is set at 5% per late day for a maximum of 5 days. This is the UNSW standard late penalty. For example, if the assignment receives an on-time mark of 70/100 and is submitted three days late, it will receive a mark reduction of $70/100 * 15\%$. After five days, the assignment will receive a mark reduction of 100%.

Form of Submission: This is an **individual** or group of **two students** assignment. Write the names and zIDs of each student at the top of `solution.py` and in your report. **If submitted in a group, only one member should submit the assignment. Also, create a group on WebCMS by clicking on Groups and Create, including both group members**.

There is a maximum file size cap of 5MB, so make sure your submission files are at most this size.

You are allowed to use the libraries specified in the file `example_solution.py`. No other library will be accepted, particularly libraries for graph and probabilistic graphical model representation and operation. Also, you can reuse any source code developed in the tutorials.

You can submit your solution via WebCMS or using give. On a CSE Linux machine, type the following on the command line:

```
$ give cs9418 ass2 solution.py report.pdf *.csv *.py *.pickle *.json
```

You can submit zero or more csv/pickle/json files to store the parameters of your model, to be loaded by `solution.py` during testing. You may submit `data1.csv` and `data2.csv` with your files if your program needs them. You can include zero or more Python helper files in the submission to organise your code using multiple files.

Recall the guidance regarding plagiarism in the course introduction: this applies to this assignment, and if evidence of plagiarism is detected, it will result in penalties ranging from loss of marks to suspension.

Description

In this assignment, you will write a program that plays the part of a "smart building". This program will receive a real-time stream of sensor data and use this data to decide whether to turn on the lights in each room. Your goal is to minimise the cost of lighting in the building while also ensuring that the lights stay on if there are people in a room. Every 15 seconds, you will receive a new data point and have to decide whether each light should be turned on or off. The building has several types of sensors, each with different reliability and data output. You will receive two files called `data1.csv` and `data2.csv` containing two days of complete data with all sensor values and the number of people in each room.

You can approach this assignment in many different ways. We will not be giving any guidance on what algorithms are most appropriate.

Your solution must include a Probabilistic Graphical Model as the core component. This core component should be your implementation and can include any tutorial code, but it cannot simply be a sklearn implementation. You can use any algorithm as part of your approach, including any algorithm available in Python's sklearn library.

Start this assignment by brainstorming several different possible approaches. Make sure you understand what information is available, what information is uncertain, and what assumptions it may be reasonable to make.

Every area on the floor plan is named with a string: `r`, `c`, or `outside`. The strings `r` and `c` stand for room and corridor, respectively.

Data

The files `data1.csv` and `data2.csv` contain two complete data representative of a typical weekday in the office building. This data includes the output of each sensor and the actual number of people in each room. This data was generated using a simulation of the building, and your program will be tested against many days of data generated by the same simulation. Because this data would be expensive to collect, you are only given two sets of 2400 complete data points from two workdays. The simulation attempts to be a realistic approximation, including many different types of noise and bias. You should treat this project as if the data came from a real office building and will be tested on real data from that building. You can make any assumptions that you think would be reasonable in the real world, and you should describe all assumptions in the report. Part of your mark will be determined by the feasibility of your assumptions if applied to the real world.

The number of people who come to the office each day varies according to this distribution:

```
num_people = round(Normal(mean=20, stddev=3)).
```

This information was obtained from records of the number of workers present each day, and the empirical distribution of `num_people` was found to be identical to the above distribution.

Data format specification

Sensor data

Your submission file must contain a function called `get_action(sensor_data)`, which receives sensor data in the following format:

```
sensor_data = {'motion_sensor1': 'motion', 'motion_sensor2': 'motion',
'motion_sensor3': 'motion', 'motion_sensor4': 'motion', 'motion_sensor5': 'motion',
'motion_sensor6': 'motion', 'motion_sensor7': 'motion', 'motion_sensor8': 'motion',
'motion_sensor9': 'motion', 'motion_sensor10': 'motion', 'door_sensor1': 0,
'door_sensor2': 0, 'door_sensor3': 0, 'door_sensor4':0, 'door_sensor5':0, 'door_sensor6':0,
'door_sensor7':0, 'door_sensor8':0, 'door_sensor9':0, 'door_sensor10':0, 'door_sensor11':0,
'camera1': 0, 'camera2': 0, 'camera3': 0, 'camera4': 0,
'robot1': ('r1', 0), 'robot2': ('r8', 0), 'time': datetime.time(8, 0)}
```

We have four different kinds of sensors for this building:

- The first one is the motion sensor. The output of this sensor is two states: `motion` and `no motion`. `motion` means the sensor captured a motion in the room, and `no motion` means the sensor did not capture anything. We know that the performance of these sensors is not very accurate, but these sensors are cheap and accessible.
- The second sensor is the door sensor. These sensors count the number of people in or out of the room (either way). So, the output of this sensor is a positive integer. Same as motion sensors, door sensors are not 100% accurate either.
- The third sensor is the camera sensor. These sensors are more accurate than the other ones and more expensive. As before, sensors in the real world are only partially accurate. The output of this sensor is the absolute number of people it counts. Since these sensors are more expensive, very few rooms have them.
- We also have two robots in the building. The robots wander around the building and count the number of people in each room. The value is a 2-tuple of the current room and the number of people counted. For example, if the robot goes into r4 and counts eight people, it would have the value ('r4',8). If it goes into corridor 'c2' and no one is present, it would have a value ('c2',0).

The value of `time` is a datetime.time object representing the current time. Datapoints will be provided in 15-second resolution, i.e., your function will be fed data points from 15-second intervals from 8 am - 6 pm.

Training data

Both `data1.csv` and `data2.csv` have the same structure. They contain a column for each of the above sensors and columns for each room, which tell you the current number of people in that room. The columns of data files are the following and can be divided into two groups:

1. Columns representing sensor readings, as described in the previous section.
2. Columns that are present **only** in the training data and provide the ground truth with the number of people in each room, corridor, and outside the building: `r1`, `r2`, `r3`, `r4`, `r5`, `r6`, `r7`, `r8`, `r9`, `r10`, `c1`, `c2`, `c3`, `outside`. This ground truth data provides the instantaneous count of people per room (i.e. every 15 seconds, a snapshot of each room is magically taken at exactly the same time, and the number of people in each room is counted. If someone passes through multiple rooms within 15 seconds, they will not increment the count in multiple rooms, only in one room).

Note that the first column of data files is the index and has no name.

You should use this data to learn the parameters of your model. Also, you can save the parameters to csv files that can be loaded during testing.

Action data

`get_action()` must return a dictionary with the following format. Note that every numbered room named `r` in the building has lights you can turn on or off. All other rooms/corridors have lights that are permanently on, which you have no control over and do not affect the cost.

```
actions_dict = {'lights1': 'off', 'lights2': 'off', 'lights3': 'off',
'lights4': 'off', 'lights5': 'off', 'lights6': 'off', 'lights7': 'off',
'lights8': 'off', 'lights9': 'off', 'lights10': 'off'}
```

The outcome space of all actions is (`on`,`off`).

In the provided `example_solution.py`, there is an example code stub that shows an example of how to set up your code.

Figure 1 shows the floor plan specification. Please notice a door between room 3 and the outside area (entrance door), but there is no sensor in this door.

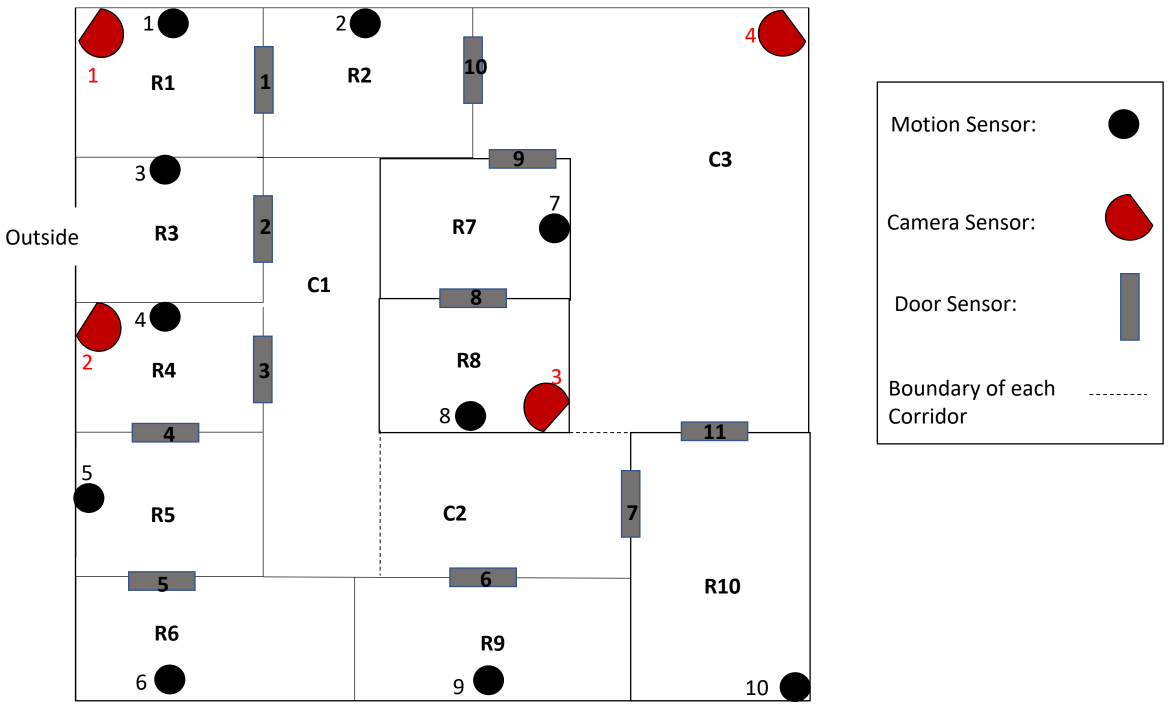

Figure 1: Floor plan.

Cost specification

If a light is on in a room for 15 seconds, it usually costs 2 cents. If people are in a room with no light on, it costs 4 cents per person every 15 seconds because of lost productivity. The cost can be calculated exactly using the complete training data, so it is also based on an instantaneous count of the number of people in each room.

Your goal is to minimise the total cost of lighting plus lost productivity, added up over the whole day. You do not need to calculate this cost. The testing code will calculate it using the actions returned by your function and the true locations of people (unavailable to you). The file `example_test.py` shows how the cost is calculated.

Testing specification

Your program must be submitted as a Python file called `solution.py`. During testing, `solution.py` will be placed in a folder with `test.py`. A simpler version of `test.py` has been provided (called `example_test.py`), so you can confirm that testing will work. A more elaborate version of `test.py` will be used to grade your solution.

There is a strict upper time limit for the final submission. If your submission runs for longer than 1800 seconds for 10 days (180s/day), it will be cut off, and you will receive 0 marks for the programming section of the assignment.

The file `solution.py` will be reloaded for each new day (using importlib.reload), so you may do any daily setup in that file outside of the `get_action` function, and it will still work.

We will run a test evaluation one week before the deadline and release the cost and time of each student's model. This is to help you confirm that your model has no major errors and works with the evaluation system. To participate, submit the assignment one week before the final deadline. We will announce this to remind you. To ensure you have something to submit, try to implement at least a minimum working model by this time.

Report

Your report should cover the following points:

- What algorithms did you use? A brief description of how they work and their time complexity.
- Briefly justify your methods (if you tried different variations, describe them).
- Any assumptions you made when creating your model.

The report must be less than 2000 words (around four pages of text). The only accepted format is PDF.

Marking Criteria

This assignment will be marked according to the following criteria:

1. 50% of the mark will be determined by the cost incurred by your code after several days of (hidden) simulated data.
2. 30% of the mark will be determined by the description of the algorithms used and a short justification of the methods used.
3. 10% of the mark will be determined by a description of the assumptions and/or simplifications you made in your model and whether those assumptions would be effective in the real world.
4. 10% of the mark will be determined by the quality and readability of the code.

Items 2 and 3 will be assessed using the report. Items 1 and 4 will be assessed using Python files.

Please include the code you used for learning your parameters, even if that code is never called during test time and parameters are simply loaded from files.

Bonus Marks

Bonus marks will be given to the top 10 performing programs (10 percentage points for 1st place, 1 percentage point for 10th place). If you score 98% on the assignment and come fifth in the final ranking, you will receive 103%.