Dean Hou, z5163159
Tony Lu, z5204814

COMP9418 Assignment 2

Group: Tony and Dean

Introduction

In this assignment we use a probabilistic graphical model to make decisions on automatically turning lights on or off in a building, simulating a “smart building”. The goal of our model is to minimise the costs associated with the lighting of rooms in the building and the loss of productivity if people are in a room and the lights are not on for a single work day.

Data

The data given in data.csv contains the complete data for a single weekday at the building. The data has 2400 data points, with each data point including sensor data as well as the true number of people in the rooms, corridors and outside areas in 15 second intervals from the times 0800 to 1800. There are 4 reliable sensors, 4 unreliable sensors, 4 door sensors and 2 robot sensors in the building which we can use to tune the accuracy of our model. The data has no missing values or any “None” values. We aim to use this data to create a model that accurately determines whether there are people in the rooms and use this to turn the lights on or off.

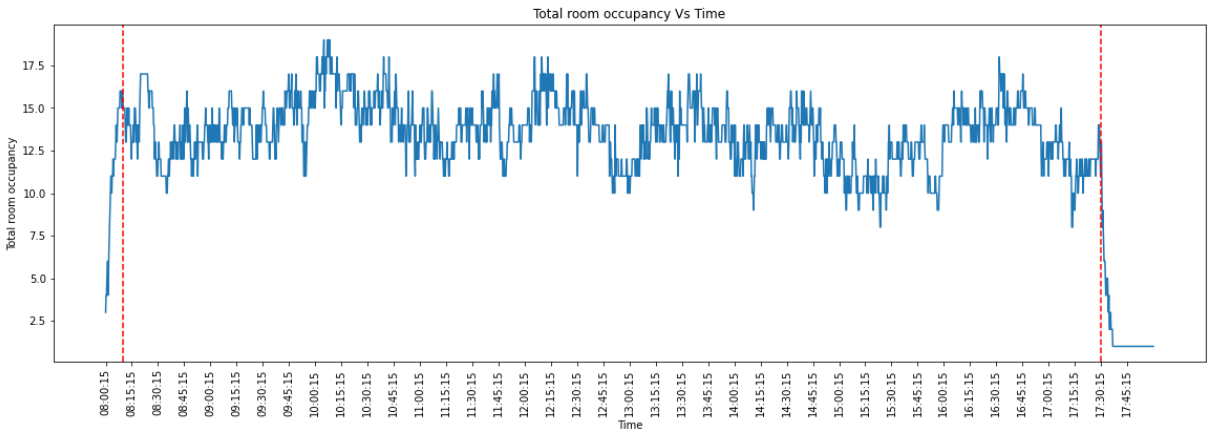
Choice of Model (Markov Chain)

In the initial planning stage, we first looked at the data and tried to determine the best approach to the problem. We noted that the problem required a dynamic solution as the way that testing was done was to create a new prediction for each time step, and at each time step the states of the rooms would change.

From the assignment specifications and the data we noted that:

- Sensors are only located in specific rooms and areas, and while it was possible to potentially use them to predict the probability of people being in rooms nearby this was not always the case, so they had limited use.
- From our own data analysis, some sensor information was much less accurate than others, so sensor reliability would need to be accounted for.
- There are many interconnected rooms and corridors, however many rooms are only connected to one or a few others, so we can assume that in a 15 second interval a person cannot move very far between rooms.
- We also noted that the number of people inside the rooms over time had sharp rises at the start and at the end of the day, representing people coming in for work and people leaving work, as seen in the graph below.

Dean Hou, z5163159
Tony Lu, z5204814



We also researched appropriate models and found that an effective intelligence lighting control system depends on a prediction model that was able to capture behaviour of occupants with high accuracy (Salimi et al., 2019). The researched model also had variations in occupants' movement due to temporal behaviour, and divided the day into different time periods.

From our analysis and research, we believe that an inhomogeneous Markov Chain prediction model based on occupancy data caters to the assumptions and data we are given. We chose this because Markov Chains are Dynamic Bayesian Networks that allow us to represent the problem in both time and space. Our Markov Chain model also takes the sensor data into account to correct any rooms where the sensor does exist based on the reliability of those sensors' true positives and false negatives (found from the given data).

We use an inhomogeneous Markov chain as the probability of going from one state to another is dependent on the time in which it is being made. This is because the probability of occupancy of each room varies within the day, such as at the beginning of the day, the transition matrix should show that there is a higher probability to go towards an office room rather than back outside as people are arriving for work. Thus, our model splits the day into 3 time periods with a transition matrix each, from 0800-0805, 0805-1730, 1730-1800.

Along with the assumptions made in our initial data analysis above, further additional model assumptions are discussed further below.

How our model works

Our Markov chain aims to track where people move from timestep to timestep as its states, thus predicting occupancy. Each state in a timestep contains all 41 rooms (rooms + corridors + outside areas). To create a Markov Chain we need 2 core components: the prior probabilities of the state and the transition probabilities.

The prior state in our model is chosen from the first row of the training data, with each room state having its corresponding ground truth values in the data.

For creating transition probabilities, we create 3 transition matrices using the data split into 3 time periods (0800 - 0805, 0805-1730, 1730-1800). Each of these datasets goes through the transition matrix generator individually but the calculation of these matrices remains the same. The transition matrix is a 41 x 41 matrix where each value along the diagonal is the probability of people staying in the room, and values not on the diagonal represent the probability of the room column moving to the room row.

To calculate the probability of staying in the room, we count the number of times the number of people in the room has stayed the same between time steps. This is divided by the total number of people in that room for the entire time period. To calculate the probability of people moving from room A to room B, we find the number of times people from room A has decreased and the number of people in room B has increased and divide that by the total amount of people in room A over the time period. Note that for each room, the possibility of people moving from room A to room B is limited by the fact that people can only move between rooms they are nearby, which is defined in a graph in the transition matrix generator. Thus it is impossible to teleport to the other side of the building. This is done room by room for all three split sets of data to create 3 matrices.

The sensors are used to correct the room states for the rooms the sensors are relevant to (i.e. the rooms the sensors are situated in or the two rooms either side of the door sensors). This can also be used to avoid the vanishing probability problem. We mainly focus on 2 probabilities of each sensor: the true positive rate and the false negative rate. The true positive rate is found with the following formula:

$$TP = \frac{times\ sensor\ detected\ motion\ and\ room\ had\ people}{total\ number\ of\ times\ sensor\ detected\ motion}$$

The false negative rate is found by the following formula:

$$FN = \frac{times\ sensor\ detected\ no\ motion\ and\ room\ had\ people}{total\ number\ of\ times\ sensor\ detected\ no\ motion}$$

For the door sensors we do the same for reliable and unreliable sensors but for counting people on each side of the door.

We correct the room states by replacing the corresponding room with the sensor true positive if the room state value is lower than the threshold and with the false negative if the room state value is lower than the threshold. In this case the threshold is 0.15 (found by trial and error to minimise cost). For the robot sensor, its accuracy is 100% so we always replace the room data with the number of occupants detected by the robot. We try to correct these values accordingly as if we see that a sensor says there is motion but it is highly inaccurate then we will replace the value with a lower one, whereas if it is accurate then it will keep the state at a higher value. If the sensor detects no motion, and if the sensor is unreliable at detecting no motion, we will replace it with a high value, else the room will remain at a low state value. This essentially biases the model to predict turning the light on.

At each timestep or passage of time in the Markov Chain, we perform matrix multiplication of the previous room state with the appropriate transition matrix based on the input time. With three transition matrices, the Markov Chain model can be considered as 3 Markov Chains, where the first chain uses the prior state that we have provided, and the subsequent Chains using the last state of the previous Markov Chain as its prior probabilities. Then we correct the room states based on the reliability of the sensors using the sensor logic above. “None” values in the sensor data are ignored and the room state is left as is after the matrix multiplication. After these adjustments we use a threshold of 0.15 to turn the room lights on or off. Each room light corresponds with an index in the state array, and if it is above the 0.15 threshold then re-turn on that room's light, otherwise we turn the light off.

Efficiency

The total time complexity is O(nX), where X is the number of all rooms (in this case 41), n is the length of the training set. The reliability of the sensors is calculated in O(nS), where S is the number of sensors (14), n is the length of the training set. The time complexity Markov Chain for one whole day is O($n\,|X|^2$), where X is the number of rooms and n is the length of the total data for the data. The transition matrices and reliability matrices are all precalculated and stored into CSVs to speed up the Markov chain process.

Model Assumptions

- Future states are independent of past ones given the current state (Markov property)
- The transition probabilities are the same for all states in the time (for the same time period in the model, in this case we have effectively 3 Markov chains as we use 3 transition matrices).
- We assume that a change in the number of people in one room is related to the occupancy of nearby rooms and that no change of occupancy in a room implies that all people stayed in that room. Nearby rooms are defined in a graph in the transition matrix generator.
- It is more expensive to have lost productivity (not turning on the lights when someone is in the room is 4 cents per person in that room) compared to wasted electricity (turning on the lights when nobody is in the room is 1 cent per 15 second interval), so our model is designed with that bias in mind, hence why the threshold is near 0.
- The sensor information only affects the rooms that they are in, and door sensors only indicate presence in the rooms either side of the door.

Results

The initial model submitted on the 15th of November scored 79261 on the leaderboard with a total runtime of 40 seconds. This initial model contained an early version of the transition matrices and no reliability metrics were taken into account, with initial testing on example_test.py scoring around 74000. A second updated model was submitted on the 19th of November. This model included some reliability metrics and modifications to the transition matrices to try to lower cost, with the example_test.py testing scoring around 67000. This model scored 78786 on the second leaderboard with a runtime of 40 seconds, a very minor improvement to our first model with the same speed. From this we can conclude that our model is fast to run but it is insufficiently accurate.

References

Salimi, S., Hammad, A., & Liu, Z. (2019, 01 29). *Occupancy prediction model for open-plan offices using real-time location system and inhomogeneous Markov chain*. https://spectrum.library.concordia.ca/id/eprint/984991/1/Hammad-2019.pdf