Eu Shaun Lim, z5156345
Utkarsh Gulumkar, z5287077

Introduction

The aim of this assignment is to design a smart building that automatically turns on the lights in the room when a person walks in and at the same time minimizes the total electricity and productivity cost for a single work day. This is done through the application of Probabilistic Graphical Models which has been learnt throughout the course.

Data Description

The dataset given, *data.csv*, consists of the sensor readings in each room at each time step in 15 second intervals starting from 8:00:15 to 18:00:15. The sensor readings consist of 4 reliable motion sensors, 4 unreliable motion sensors, 4 door sensors and 2 robot sensors scattered across the building. There are also the ground truth values of the number of people at each room for every time step and the electricity cost.

Choice of Algorithms

Two algorithms were used in the design of the program, which are the Particle Filtering method and the Hidden Markov Model (HMM). These algorithms were chosen after a few rounds of discussions between group members.

During the initial discussion stage, we decided to first create a simple model which decides on the outcome for one room, and then build a more complicated model based on the initial model. Since there were a few rooms that were equipped with motion sensors, we decided that the HMM was the most appropriate model for these rooms. Since there were six rooms with motion sensors, we decided to build six HMMs, one for each of the six rooms.

The next step was to decide on the outcome for the rooms without motion sensors. The initial thought was to model the building using a Markov Network. This would rely on the outcomes of the six HMMs and calculate the probability of a person moving from the rooms with motion sensors to the adjacent rooms. However, we decided that it would be too complicated to implement and require assumptions that may not apply to the real world.

Instead, we decided to model the movement of the people in the building. Particle filtering was chosen as the main idea in particle filtering is to resample the particles from one state to another for each time step. This is similar to our problem, where each person can be thought of as a particle moving from one room to another at each time step. We will then turn on the lights in the rooms where a person has been moved to and turn off the lights for those without people.

We made the decision to combine both models by first simulating the movement of the people in the building using particle filtering and then running the HMM for the six rooms with motion sensors. The rationale behind this is that the HMM is more accurate than particle filtering since we are able to calculate the exact transition and emission probabilities from the training data, while the transition probabilities for particle filtering is not very accurate (more will be explained later). The outcomes in the HMMs will override the outcomes in particle filtering for those six

rooms. Importance weighting and resampling was not used in particle filtering since we already used the HMM to decide the outcome for the rooms with evidence.

Finally, we will look at the robot sensors as they are the most accurate given that they provide the ground truth values for the rooms they are in. If the robot sensors are in any of the rooms and it detects people, we will then turn on the lights no matter the outcome of the HMM nor particle filtering.

Program Description

The main program is written in Python 3.7.3 and consists of 4 files. The main file, `solution.py`, houses the main function, `get_action()`, which calls on the functions in the supplementary files and decides whether to turn the lights on or off in each room at every time step.

`state.py` stores the initial state of the floor plan. The floor plan is an undirected graph in the form of a dictionary where the keys are the names of each room and each room is another dictionary containing a list of people in the room and a list of adjacent rooms. This state is a global variable in solution.py and the number of people in each room will be updated at each time step.

`particle_filtering.py` consists of the functions needed to simulate the movement of people in the building using the particle filtering technique.

`hidden_markov_model.py` contains the calculated transition and emission probabilities and necessary functions needed to build a Hidden Markov Model for the rooms with motion sensors.

There is an additional file, `pickling.py`, which is not needed to run the program. This file calculates the probability of a person being in a room at a given time step and stores the values in a dictionary. It then uses the `pickle` library and stores the dictionary into a pickle file called `parameters.pickle`, which will then be loaded at the start of the main program. This reduces the number of operations needed in the main program, which in turn reduces the runtime of the main program by a large margin.

Analysis of Algorithms

Particle Filtering

Particle filtering was used to simulate the movement of every person in the building. In order to reduce the runtime of the program and increase efficiency, we first calculated the probabilities of being in each room at each time step and serialized the resulting probability using the `pickle` library. Before we start the simulation, we then load the probabilities and sample the number of people from a normal distribution with a mean of 20 and standard deviation of 1. We then assign all the people to the outside of the building at 08:00:00, which is one time step before the first time step in the training data.

At each time step starting from 08:00:15, we assign new rooms to a person by first getting the list of possible rooms to move to from his original room. The possible rooms consist of the

original room, the neighbouring rooms which are connected to the original room and all the rooms connected to the neighbouring rooms. We then obtain the probabilities of moving to all the possible rooms from the pre-calculated probabilities. These probabilities are then normalized and a number is sampled from the uniform distribution which is then used to determine the next room that the person will be in the next time step. This process is repeated until every person in the building has been assigned a new room. The global state variable is then updated with the newly assigned rooms for everyone.

Since we have to calculate the probabilities of every room at every time step, the time complexity of calculating the probabilities is O(mn), where m is the number of rooms in the building and n is the number of time steps. Since this process was done before the simulation, the main program does not incur this cost. The time complexity for the main program is O(nlog(m)), where m is the number of people sampled and n is the number of time steps.

Hidden Markov Model (HMM)

For the HMM, we first need to calculate the transition and emission probabilities for the six rooms with motion sensors which can be obtained from the training data. From the training data, we can assume that the light is always on when there is someone in the room and always off when nobody is in the room. The transition probability, P(X_t | X_t-1), where X_t is the lights being on or off at time step t, for a room can then be calculated by counting the frequencies of the lights being on or off happening given that the lights were on or off in the previous time step. This number is then divided by the probability of the probability of the light being on or off. The emission probability, P(E_t | X_t), is calculated similarly, where E_t is the state of the motion sensor in the room.

Once the transition and emission probabilities are calculated, we can start building the Markov chain. The initial chain starts from 08:00:00 to 08:00:15 and the first evidence is seen at 08:00:15. At each time step, we add on to the chain with evidence from the motion sensor and the outcome at each time step is determined using the Viterbi algorithm. The Viterbi algorithm works by joining the prior probability with the transition and emission probabilities and choosing the outcome with the highest probability at each time step. The probabilities are normalized at every time step to prevent underflowing probabilities. At every time step, we obtain new evidence from the motion sensors and this evidence is then fed into the Markov chain and new outcomes are obtained.

The time complexity for calculating the transition and emission probabilities are both O(n), where n is the number of time steps in the training data. Since this was calculated beforehand, this cost was not incurred in the main program. The time complexity of the Viterbi algorithm is O(n|X|^2), where n is the number of time steps and X is the number of states.

Results

The model was submitted to the 20 Nov leaderboard for evaluation and achieved an average cost of 90,765 with 77 seconds of total runtime. Using the supplied example_test.py, the model achieved an average cost of 61,785 with a total runtime of 76 seconds in 10 test runs. This shows that the model has overfitted to the training data. We attributed this to the incorrect transition probability for the particle filtering algorithm. Since it was impossible to track the movement of the people in the training data, we could not obtain the proper transition probability of moving from one room to another. Instead, we decided to use the probability of being in a room at a given time step. This probability is too specific for this dataset and does not generalize well, and thus overfitting has occurred.

Another method of obtaining the transition probability was to first simulate one day of movement using our current particle filtering method, however this time tracking the movement of all the people. This way, we can then calculate the probability of moving from one room to another and this may have prevented overfitting. However, due to time constraints we were unable to implement and evaluate this method.

Assumptions / Future Works

One assumption made is that the motion sensors have a range spanning the whole room it is located in and is able to detect motion as soon as a person walks into the room. In the real world, the motion sensors may only detect motion up to a certain radius of the sensor and therefore may not be as reliable. This assumption was made in order to simplify the design of our model.

Another assumption made is that every person can only move as far as two rooms away from his original room. This assumption was made to determine the most likely room the person will be in for the next time step and to prevent the particle filtering from moving everyone across the whole building. The thought process behind this is that it is very unlikely for a person to move to a room very far away from his original room in 15 seconds, hence the decision to limit the possible rooms. An indirect assumption made due to this is that the distance of every room to another is equal, which obviously does not reflect well in the real world as seen in the floor plan.

The program could be improved in the future by using better transition probabilities for particle filtering as mentioned in the results section. If we were able to track the movement of the people in the building, we could then obtain a more realistic transition probabilities and potentially reduce overfitting. The assumption of limiting the possible rooms would also not be needed as we will have exact probabilities of moving from one room to another.

Conclusion

In conclusion, we have designed a program that uses Probabilistic Graphical Models such as the Particle Filtering technique and Hidden Markov Models to determine whether to turn on or off the lights for the rooms in a smart building.