HVAC Hidden Markov Model

—approximated model using time shift

Project private github repository: https://github.com/z2862658714/hvac_hmm

Overview

The HVAC Hidden Markov Model (HMM) is a model that represents the status of a house being occupied by residences or not, observed by the motion sensors from the house. Theoretically, if there are residences in the house, their movements would trigger the motion sensors, and result in data we can observe. However, the data generated by the motion sensors are not always correct. Ventilation systems and other object movements may falsely trigger the motion sensors, and occupied house with residences staying still may fail to trigger the motion sensors. This resembles a hidden Markov Model because we are only able to acquire the observations from the motion sensors, and the true occupancy state is not directly available.

The Model

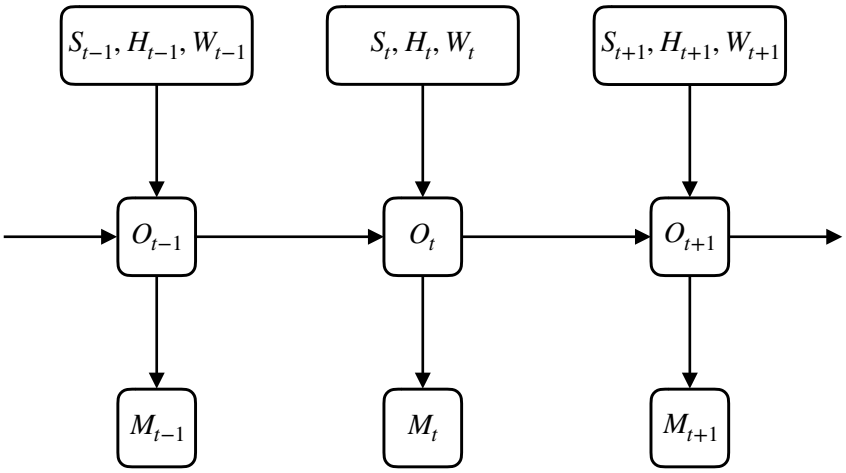

Diagram 1: HVAC Hidden Markov Model

The model is constructed on five (5) variable. Their transition probabilities and emission probabilities are formatted in two (2) matrices..

- Season (S): an observable state variable representing the season.
  S = {0: Summer, 1: Fall/Spring, 2: Winter}
- Hour (H): an observable state variable representing the time of the day.
  H = {0: 0:00:00, 1: 0:30:00, 2: 1:00:00, …, 47: 23:00:00, 48: 23:30:00}
  H increases by 1 with each half-hour increments in time.
- Week (W): an observable state variable representing if it is a weekday.
  W = {0: Weekend, 1: Weekday}
- Occupancy (O): a hidden state variable representing if the house is occupied.
  O = {0: Vacant, 1: Occupied}
- Motion sensor (M): an observation variable representing if motion is detected in the house.
  M = {0: No motion detected, 1: Motion detected}

There are four (4) state variables (S, H, W, H) and thus 3 * 48 * 2 * 2 = 576 possible state combinations. However, note that S, H, and W are observed state variables, and S and W also do not follow Markov transitions, therefore we do not consider the probabilities of transitioning to them in the transition matrix. The transition matrix has size 576 by 2, with each entry suggesting $P(O_t \mid O_{t-1}, S_t, H_t, W_t)$.

There are one (1) observation variable with 2 possible values. The emission matrix has size 576 by 2, with each entry suggesting $P(M_t \mid O_t, S_t, H_t, W_t)$.

Algorithm

While the rigorous way to train the model might be using the Baum-Welch algorithm, the algorithm proposed here serves as a "hack" way to quickly perform the training. In this study, we use the "time-shift" method to assume an initial guess of the hidden states, which is used to then generate an initial guess of the transition and emission matrices. After the initial guesses, we repetitively update the hidden state sequence using Viterbi algorithm, eventually reach our trained model.

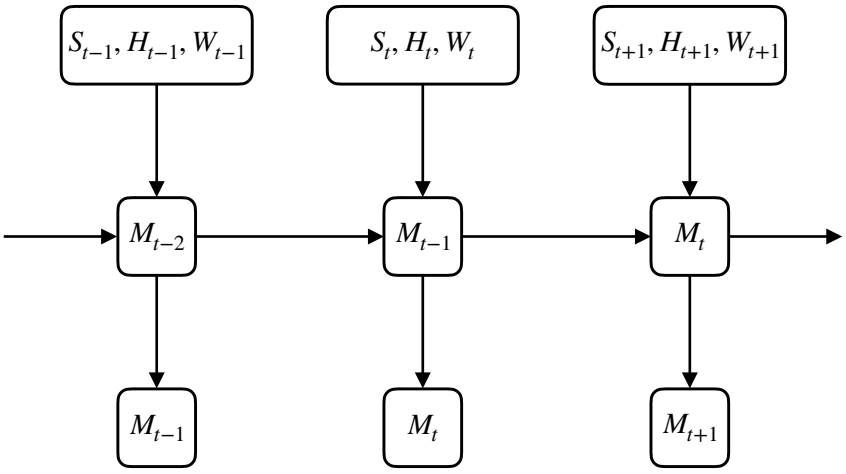

Diagram 2: HVAC HMM "time-shift" the observations to generate an initial guess of states

1. Shift all observations one step ahead as the initial sequence of true occupancy states.

2. Construct the transition and emission matrices using Naive Bayes with Laplace Smoothing

$$P(O_t \mid O_{t-1}, S_t, H_t, W_t) = \frac{P(O_t \cap O_{t-1}, S_t, H_t, W_t) + 1}{P(O_{t-1}, S_t, H_t, W_t) + 2}$$

$$P(M_t \mid O_t, S_t, H_t, W_t) = \frac{P(M_t \cap O_t, S_t, H_t, W_t) + 1}{P(O_t, S_t, H_t, W_t) + 2}$$

Note: the transition matrix is 3 * 48 * 2 individual 2 * 2 matrices.
The transition matrix depends on the value of S, H, and W.

3. Using Viterbi algorithm (hvacviterbi.m)[1] to update the most probable sequence of true occupancy states given the sequence of observations, the transition matrix and the emission matrix.

4. Step 2 and 3 are repeated using the most probable sequence of true occupancy states generated from the last iteration, until absolute convergence or max number of iterations is reached.

Testing

To test the performance of the algorithm, prediction using forward algorithm is used and the predict accuracy is observed. A small sequence in each file of data is clipped, and the rest of the data is used to train the model (parameters in the transition and emission matrices). Given the trained matrices, forward algorithm calculates the most likely observation at the next half-hour, and compare result with the actual observations.

$$P(O_{t+1}) = P(O_{t+1} \mid O_t = 0)P(O_t = 0) + P(O_{t+1} \mid O_t = 1)P(O_t = 0)$$

where

$$P(O_{t+1} \mid O_t) = P(O_{t+1} \mid S_{t+1})P(S_{t+1} \mid S_t)P(S_t \mid O_t)$$

The result is tested in three ways:

- Cluster predictions: Each file is trained using the first 75% of data, and tested using the last 25% of data. The clustering of testing data may cause insufficient training for a specific season, which is likely to be tested on, and result in poor testing accuracies.

- Random predictions: Each file is trained using random 75% of data, and tested using the rest 25% of data.

- Days predictions: 20 entire days of data were picked and separated from the each file of data. The remaining data were used for training, and then tested on the 20 days of data.

Available Data

Data used for this study was formatted by Brent Huchuk into two (2) groups [tts_1, tts_2], each consists of 25 CSV files. Each CSV consists about one year of data collected from 2016-09-01, formatted every half-hour in $[M_t, S_t, H_t, W_t]$.

Two (2) different sets of 20 days of data in the first group [tts_1] are separated to create two (2) train-test-set data [test_1, train_1], [test_2, train_2] for the days predictions testing. One (1) set of 20 days of data is separated from the second group [tts_2] to create the train_test_set data [test, train] for the days predictions testing.

Due to privacy concerns, these data are not uploaded to the github repository.

Result

---

[1] The hvacviterbi.m function is a slight modification of the hmmviterbi.m function.

For each CSV file, there is a folder named with `FILEINDEX - TESTACCURACY` contain the probability distributions with respect to the values of S, H, and W. Inside each folder there are 5 files:

>> A `FILEINDEX - result.txt` file with the number of entries in the data, number of iterations to reach absolute convergence, and the prediction test accuracy

>> A `FILEINDEX - Motion Sensor Observations.png` figure showing the probability of observing a "true" in the original motion sensor observations (original data)

>> A `FILEINDEX - Trained Occupancy States.png` figure showing the probability of the hidden occupancy states being "true", where the hidden occupancy states are generated after the training of data

>> A `FILEINDEX - Discrete Predictions.png` figure showing the predictions generated with the prediction testing (0 or 1)

>> A `FILEINDEX - Forward Probabilities.png` figure showing the probability of the next motion sensor observation being "true" during the predictions, which also leads to the conclusions in the Discrete Predictions

Note that in all figures, the X-axis represents the hour of the day (0 to 48). The red line represents weekdays, and the blue line represent weekends.

Due to the numerousness of figures, they will not be included in this document. These figures can be found under the `HackModel/results` directory in the project repository.

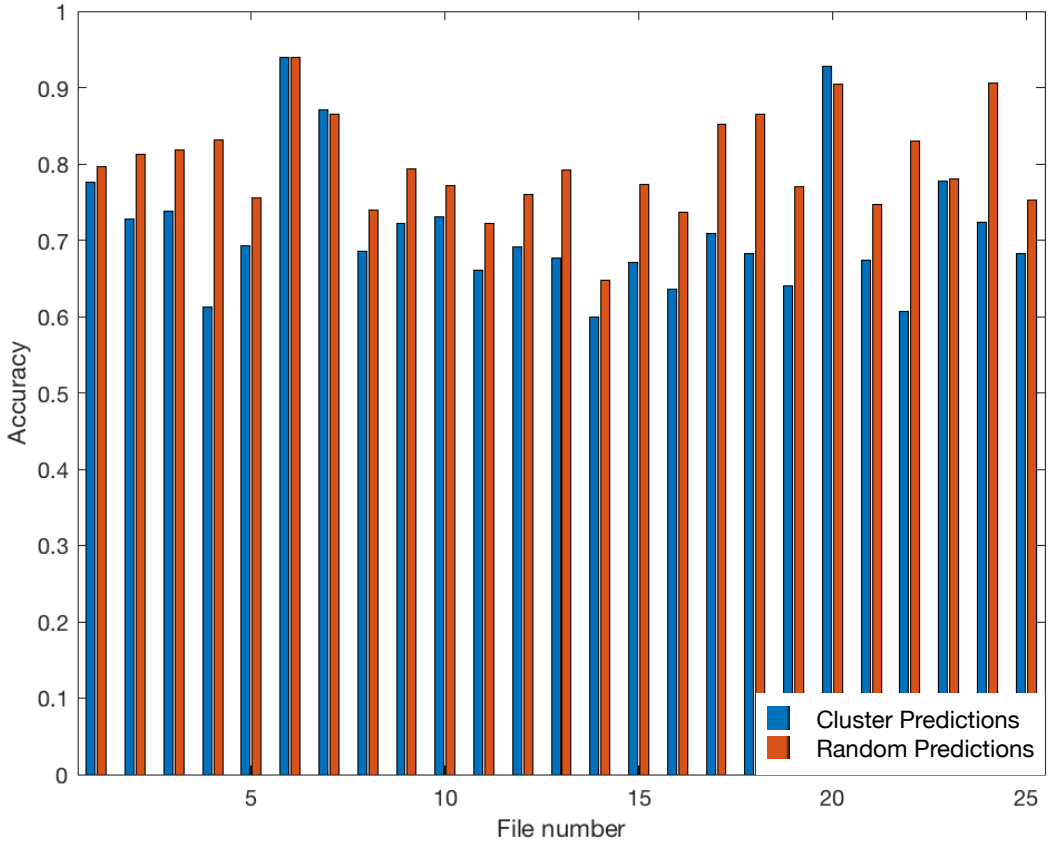

Figure 1: Accuracy of Prediction Testing using cluster predictions and random predictions in the first group of data [tts_1]. Average accuracy for cluster predictions is 71.41%, for random predictions is 79.84%.

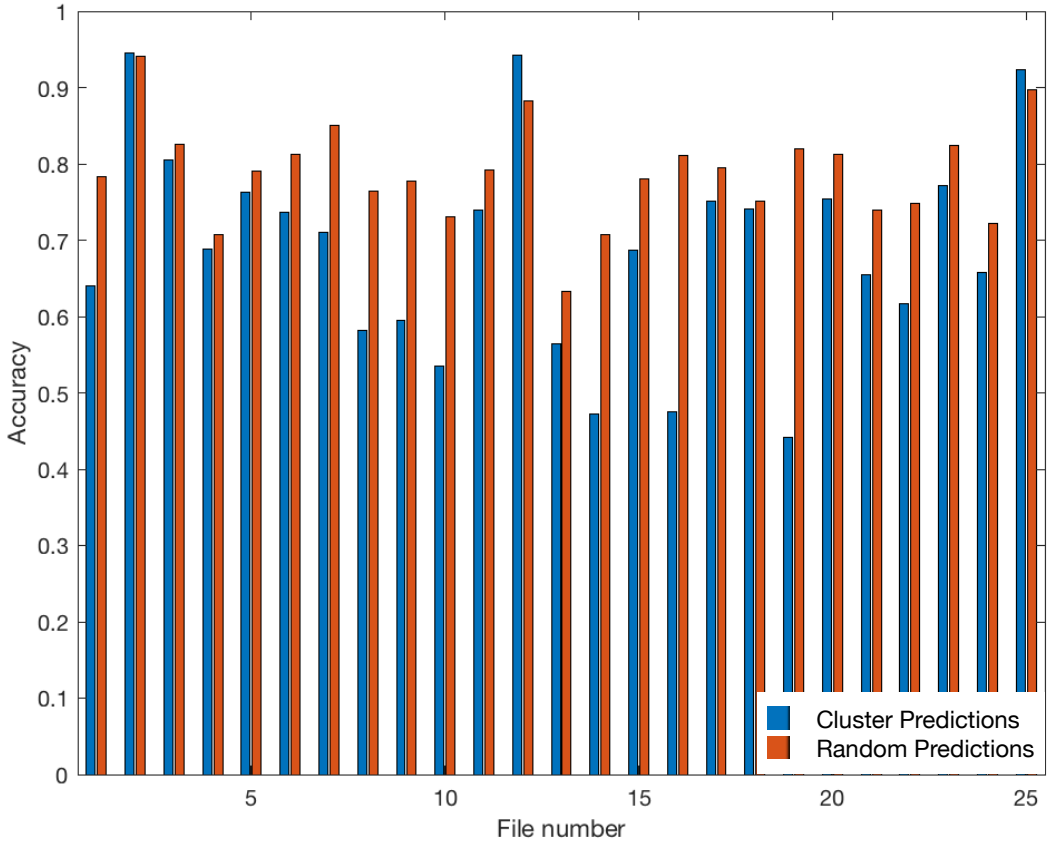

Figure 2: Accuracy of Prediction Testing using cluster predictions and random predictions in the second group of data [tts_2]. Average accuracy for cluster predictions is 68.76%, for random predictions is 78.78%.

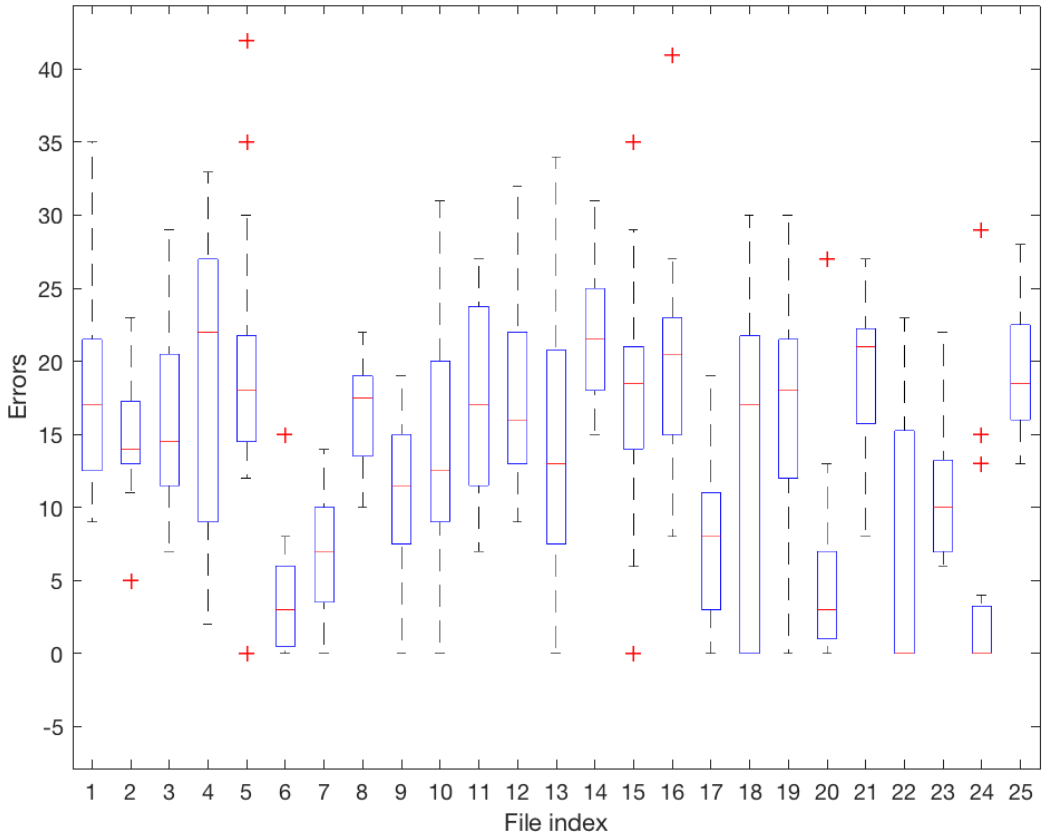

Figure 3: Number of Prediction Errors per testing day using days predictions in the first group of data [tts_1] with the first train-test-set [test_1, train_1]. Average prediction errors is 13.97.

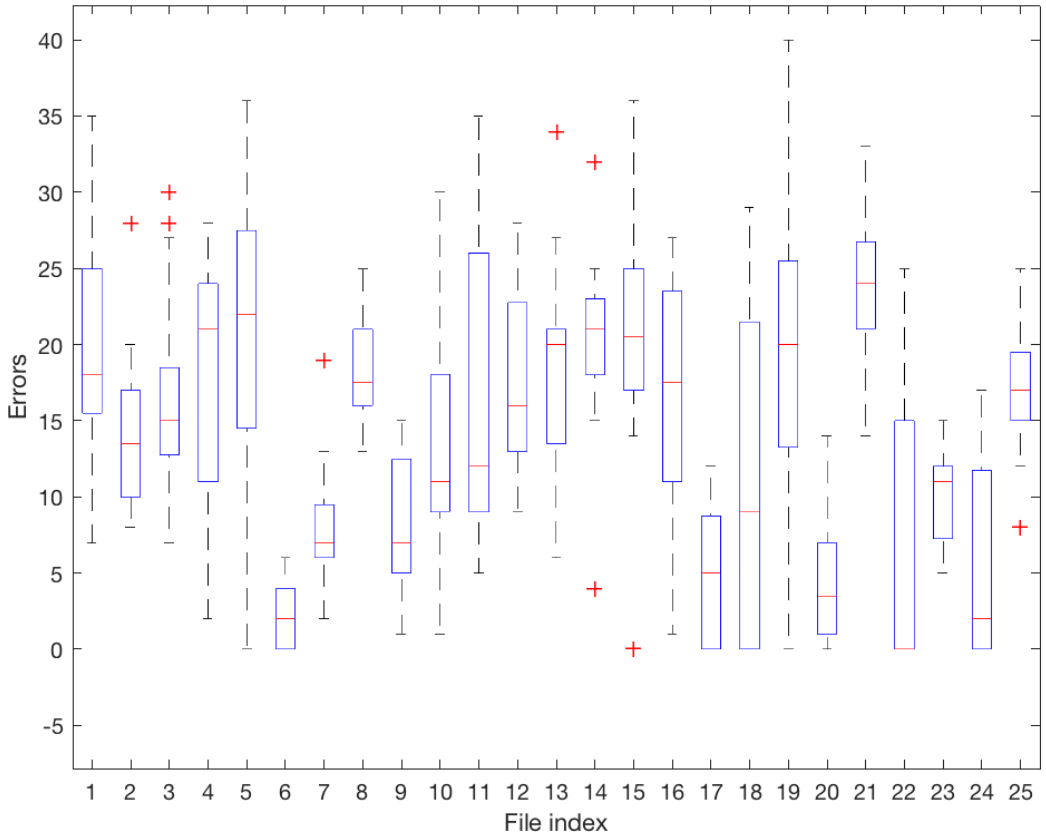

Figure 4: Number of Prediction Errors per testing day using days predictions in the first group of data [tts_1] with the second train-test-set [test_2, train_2]. Average prediction errors is 14.16.

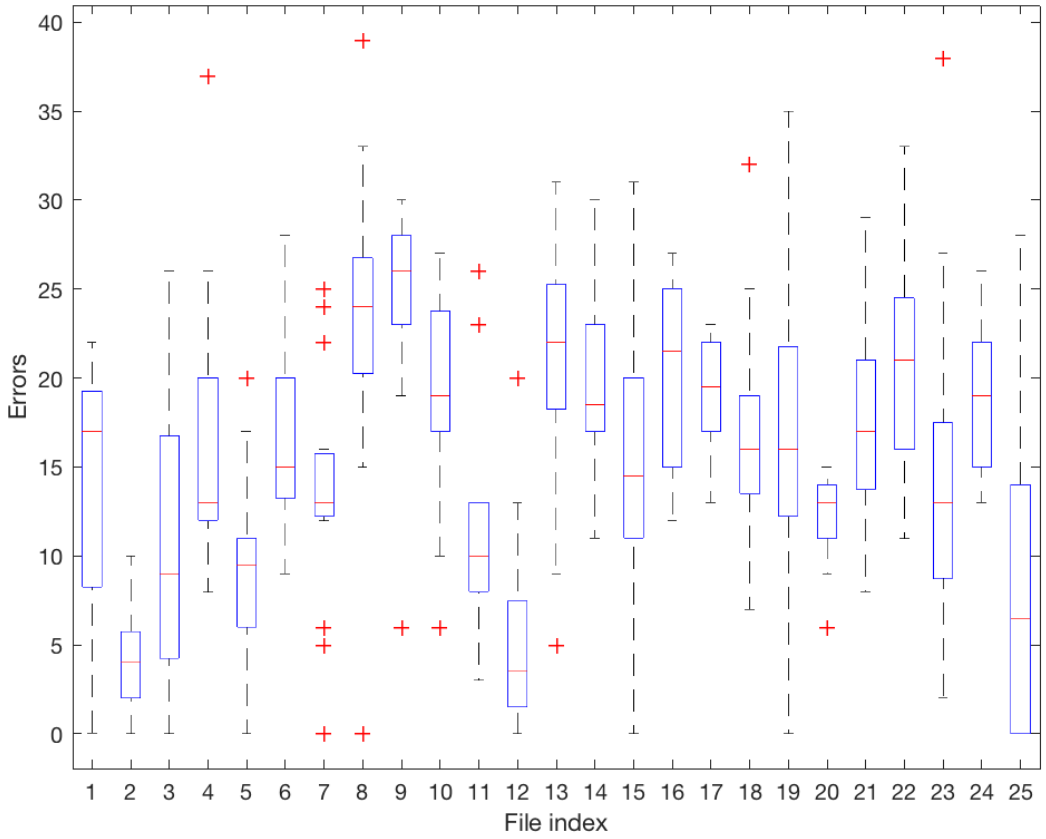

Figure 5: Number of Prediction Errors per testing day using days predictions in the second group of data [tts_2] with the train-test-set [test, train]. Average prediction errors is 15.53.

Analysis

How S, H, W affects M and O

Figure 6 shows a figure of the trained occupancy states probability distribution of an average household with respect to S, H, W. $P(O = 1 \mid S, H, W)$.

Figure 7 shows a figure of the probability distribution of the motion sensor observation in an average household with respect to S, H, W. $P(M = 1 \mid S, H, W)$.

These figures suggests that M and O have similar trends. H and W are the dominant factors that affects the M and O trend lines, while S has little to no effect on the trend lines.

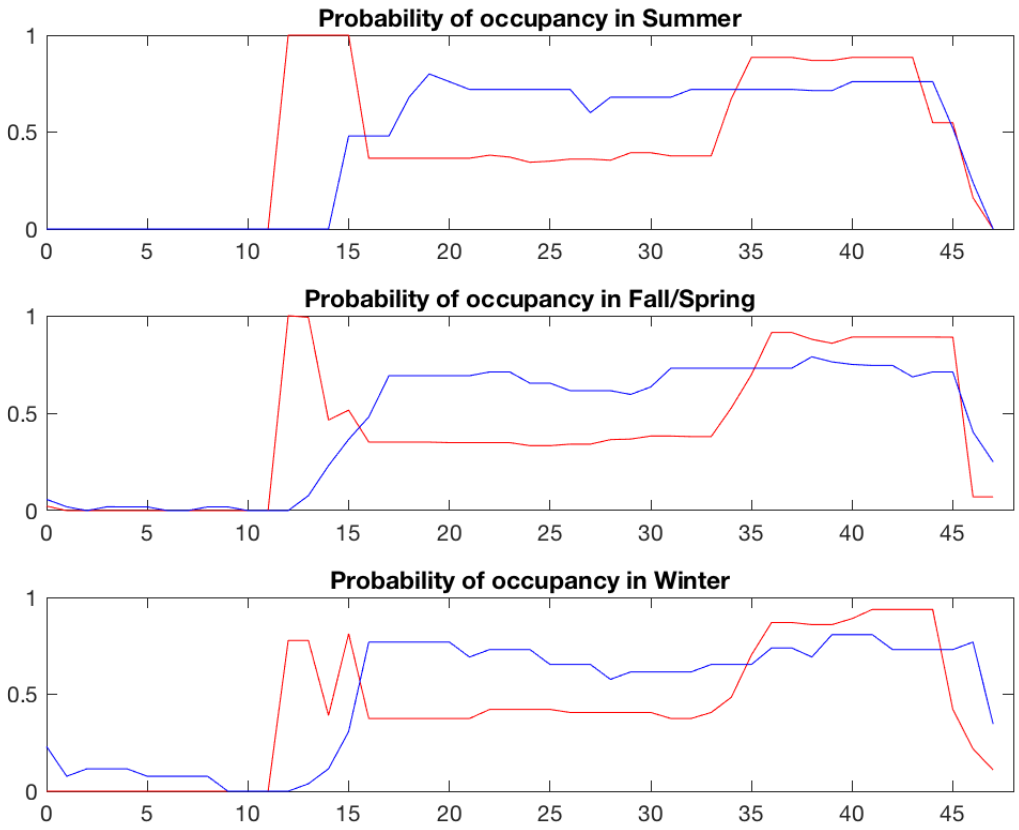

Figure 6: Trained Occupancy States probability distribution of an average household with respect to S, H, W.

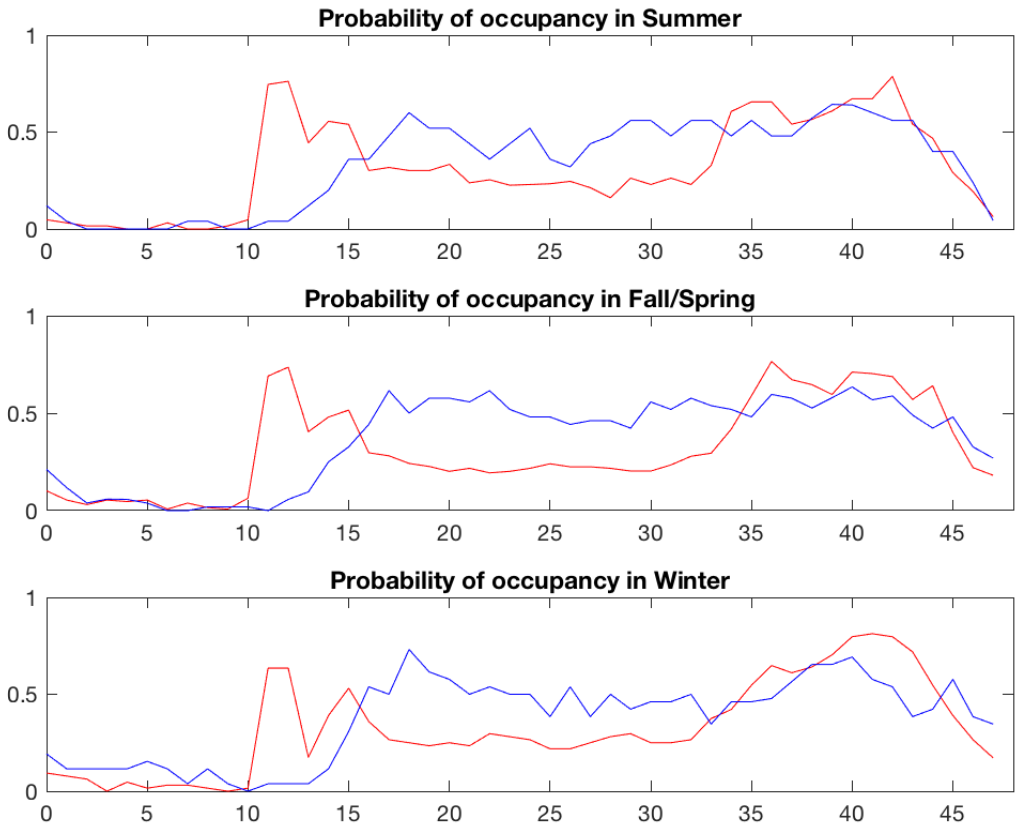

Figure 7: Probability distribution of the Motion Sensor Observations in an average household with respect to S, H, W.

The trend lines in Figure 6 and 7 show similar shapes, with the difference being the trained states probabilities are more “sharp” (or discrete). This is expected since the trained states are generated from the trained matrices, and with a given set of S, H, W the results are less stochastic compare to the sampled observations.

It is observed that in both figures, the three subplots are nearly the same, indicating that Season (S) has little effect on the occupancy state (O) and the motion sensor observations (M).

The Hour (H) and Week (W) affects the state (O) and observations (M) expectably. On a weekday (red trend line), motion is detected in the morning (residence wake up), not detected during work time (residence leaves for work), and detected again in the evening (residence return from work). On a weekend, motion is detected later in the morning 9residence wake up later in morning) and persist throughout the day (residence is not leaving on schedule for work).

Cluster Predictions vs. Random Predictions

It is able to observe from Figure 1 and 2 that random predictions has a significantly better performance comparing to the cluster predictions.

The factor that greatly reduces the accuracy of cluster predictions is that all data in Summer was used for testing. Most of the data was collected between 2016-09-01 to 2017-08-31, thus

the last 25% of the data was mostly in June, July and August, where they are the only months that S = 0 occurs. With these data removed from the training set, it is possible that the parameters in the transmission and emission matrices that correspond to S = 0 were underfit, or never fit at all. This is especially a problem in the second group of data, where 24 out of the 25 files have all S = 0 data removed from the training set, and the predictions in Summer resulted in a 50/50 guess. This make cluster predictions a very bad testing protocol, and it will not be used for other

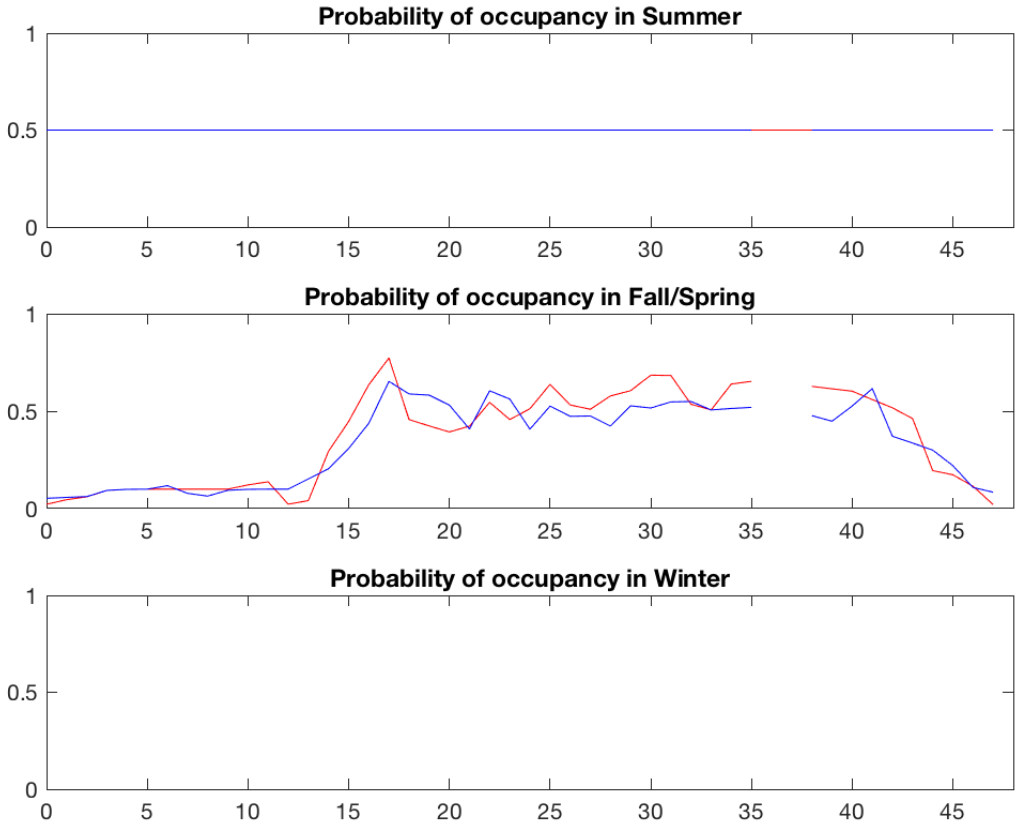

Figure 8: Next observation prediction probability distribution using cluster predictions of a file with all Summer (S = 0) data in the testing set (removed from training set). All probabilities in Summer was never trained and a default uniform distribution was used for testing, which is very random and inaccurate.

While the random predictions result in significant better performance, the scattered sampling created a lot discontinuities in the data. These discontinuities cause disconnections in the model and may result in false conditional probabilities, and will be discussed in the next topic.

Cluster predictions and days predictions always predict a sequence of observations. In such cases when predicting $P(O_{t+1})$, the probability of $P(O_t)$ is often carried on by a previous prediction and is not discrete. The uncertainty of O_t may affect the predicted value of O_{t+1}. In random predictions, most times the value of O_t is known and the probability of $P(O_t)$ is discrete, which could lead to more certain predictions.

Data Discontinuities Cause Errors in Model Structure

In Diagram 2, we showed how “time-shift” is used to generate the initial guess of the hidden state sequences. All motion sensor observations are shifted one (1) time step ahead, so that the model can still resemble its conditional probabilities about one state being dependent on the state in the previous time step.

However, in the CSV files, there exist periods of times where data are unavailable. For example, the following data are observed:

Table 1: Data with discontinuity

2016-09-01 03:00:00	0	1	6	1
2016-09-01 03:30:00	0	1	7	1
2016-09-01 07:00:00	1	1	14	1

When “time-shift” on data like this, the result will be as shown in Diagram 3.

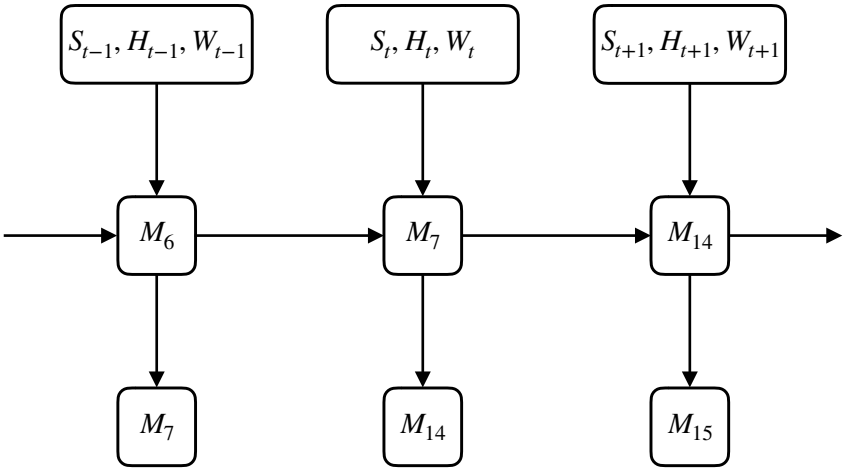

Diagram 3: “Time-shift” motion sensor observations with discontinuous data

Note that while M_{14} should have depended on M_{13}, it is now depended on M_7. This is problematic because our HMM only models $P(M_{14} \mid M_{13})$ but never $P(M_{14} \mid M_7)$, thus training data like this could affect the result of training.

Data discontinuity was not dealt with yet in this project[2], since continuous data overwhelmingly outnumbers discontinuous data. However, it is suggested to be dealt with as a further step of the project.

Data Structure Can Affect Test Accuracy

<to be completed>

---

[2] Data discontinuity was dealt with in an earlier version of the project. The code was removed due to a change in the model from $P(O_t \mid S_{t-1}, H_{t-1}, W_{t-1})$ to $P(O_t \mid S_t, H_t, W_t)$. The code is included in the Further Steps section as a suggestion.

Further Steps

Allow Training of Discontinuous Data

As discussed before, discontinuous data can create holes in the HMM, and causing errors in the model structure. If discontinuities can be detected and dealt with properly training accuracy should increase slightly.

An earlier version of the project has the following code to detect discontinuous data, and the code is displayed here as a suggestion.

```
datacontinuous = true;
if i == 1
    datacontinuous = false;
elseif mod(data(n, 3) - data(n - 1, 3), 48) ~= 1
    datacontinuous = false;
elseif data(n, 3) ~=
    if data(n, 2) ~= data(n - 1, 2)
        datacontinuous = false;
    elseif data(n, 4), ~= data(n - 1, 4)
        datacontinuous = false;
    end
end
```

If data continuity is detected, we made an assumption to guess the S, H, W of the previous time step by keeping S and W the same and decrease H by 1, since season and week does not change as often as hour. This assumption may result in around 1.0% decrease in prediction accuracy.

```
if datacontinuous
    s = data(n - 1, 2);
    h = data(n - 1, 3);
    w = data(n - 1, 4);
else
    s = data(n, 2);
    h = data(n, 3) - 1;
    w = data(n, 4);
end
```

The main problem is to guess O and M of the previous time step. While discontinuous data may be omitted to ensure the model structure is reliable, some data files have more discontinuities that omitting all of them may result in insufficient training data. It is suggested to find a way of modelling S, H, W, O, M in the missing time steps of a data discontinuity.

Increase Code Efficiency and Potentially Utilize the HMM Toolbox

Prior to the study, it was suggested to use the HMM toolbox for Matlab[3] to carry out various HMM algorithms. This was found challenging due to the model has non Markov transitions for the observable state variables S, H, W. As a result, the transition matrix used in this model has

[3] The toolbox is available at https://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html.

size 576 by 2, while the toolbox only supports square transition matrices (i.e., all state variables should possess a Markov transition). Although the toolbox was found superior in performance, its lack of documentation and code complexity makes it harder to be modified to fit this specific HVAC HMM.

The hmmviterbi.m function included within MATLAB_R2018a was found easier to modify, and thus was used as the backbone to create the hvacviterbi.m function used in this model. MATLAB_R2018a does not include a prebuilt forward algorithm function, and since the algorithm is rather simple, the prediction function was built upon an original forward algorithm that may show significant room of improvement.

Conclusion

This project studies the performance of modelling HVAC of individual households with hidden Markov model, train the model using a “time-shift” method, and predict further observations using forward algorithm upon the trained model. The prediction accuracy averages about 70% when predicting a sequence of observations and 79% when predicting individual next observations. This model/method makes around 14-15 errors (~70%) when predicting a day sequence of observations.