Anomaly-Based Intrusion Detection Using HMMs

Course: CMPT 318
Semester: Spring 2025
Group number: 2
Group members:

Kwong Hoi Chan	301608563
Duc Ta	301552746
Duc anh Nguyen	301576257
Andrew Yu	301420629

Table of Contents

Abstract **2**
Overview **3**
Problem Scope 3
Technical Background 3
Feature Scaling **4**
Feature Engineering **5**
HMM Training and Testing **8**
Anomaly Detection **11**
Conclusion **14**

Abstract

This technical paper explores the idea of anomaly-based intrusion detection systems using an unsupervised learning model approach. We implement a Hidden Markov Model (HMM) combined with Principal Component Analysis (PCA) to identify the most informative features for training our HMM based on provided normal electricity consumption data. Once trained, our model evaluates new weekly test data and calculates the deviation from its expected behaviour. A threshold approach is used to detect anomalies, which are identified through significant deviations. This method helps detect unexpected or abnormal energy usage.

Overview

Problem Scope

Supervisory Control and Data Acquisition systems, which manage critical infrastructure such as power grids, are vulnerable to cyberattacks. Intrusion Detection System plays a critical role in detecting malicious activities and preventing disruptions. We can split the detection into two classifications:

- Signature-based detection: Recognizes known attack patterns but will struggle with zero-day attacks. Zero-day attacks are essentially attacks that are first seen ever. It will take days to detect the malicious code and give out appropriate tackling solutions.
- Anomaly-based detection: Identifies deviations from normal behaviour, making it effective for detecting unknown threats.

Technical Background

We will be using Hidden Markov Models (HMMs) to model time-series data generated by our electricity consumption measuring system. The model will be evaluated using two metrics:

- Log-likelihood Comparison: Normalized log-likelihood values for training and test sets assess model fit and anomaly thresholds.
- Bayesian Information Criterion (BIC): BIC balances model complexity and accuracy, guiding the selection of the optimal number of HMM states.

Feature Scaling

Why are scaling features of a dataset necessary?

Feature scaling is necessary because many models, such as Hidden Markov Models (HMMs), are sensitive to the scale of the input data. If features have vastly different numerical ranges, the model may behave poorly. An HMM, for example, relies on probability distributions and transition matrices, so proper feature scaling can provide accurate probability estimations. Inconsistent feature scaling can distort parameter estimation, resulting in poor performance of the training model.

What do normalization and standardization do to the data and the noise?

Both normalization and standardization are considered to be important feature scaling techniques to transform input data to specific ranges and scores. Normalization rescales values to a fixed range, while standardization centers the value at a mean of 0 with a unit standard deviation.

Which feature scaling method do you choose for HMM training for anomaly detection purposes and why?

We choose standardization for anomaly detection purposes. Since we apply PCA before HMM training, it is sensitive to the scale of the input data. Normalization squishes all values into specific ranges, which hide the variance between features. If features with low variance, PCA is difficult to measure variance structure. As standardization ensures all features contribute equally by giving them the same scale, it can preserve the scale of variance between features.

Feature Engineering

Selection of response variables

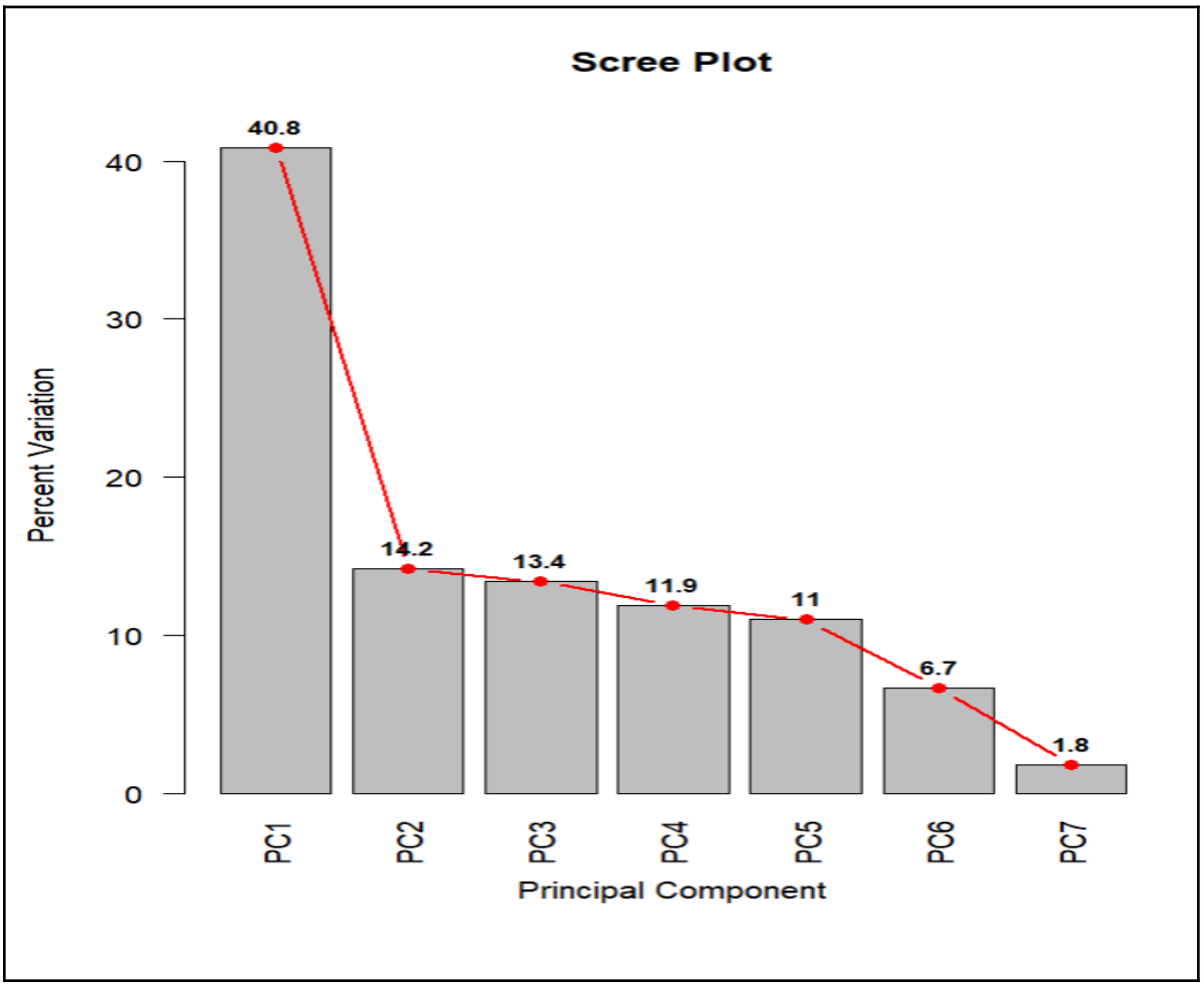
The question remains: How should we choose a subset of the response variables to train multivariate HMMs on normal electricity consumption data?

A few ideas are:

- Selecting the variables that explain the most variance in the data (using PCA). Why? We do this because we want the variables that represent the most standard behaviour of the dataset. We want a couple of variables that could work together to capture the significant distributions of observations
- Avoid using variables that have a high correlation to each other. Redundant variables don't add new information but increase model complexity.
- Variables that give good intuition on how they vary with changes in the system behaviour. We might have a better understanding of state transitions and detecting anomalies.
- We should also use domain knowledge to consider the problem.

To perform the required above task, we will be choosing a couple of variables that contain numerical data, named "Global_active_power", "Global_reactive_power", "Voltage", "Global_intensity", "Sub_metering_1", "Sub_metering_2", and "Sub_metering_3", respectively.

The scree plot shown below is the result of us running our R code.



As we can see from the plot, PC1 alone explains approximately 41% of the total variance. PC2 and PC5 explain 10% to 15% of the variance. We could not select PC6 and PC7 as they explain very little of the variance, and the difference in percentage between PC6 and PC7 likely introduces noise and redundant information if we include them in the training data. We want to select PC values that have the total sum of variation percentage of all of the variances of the dataset. Hence, we pick the first three PC values and observe which variables contribute the most. We have made a loading scores table, and it is displayed on the next page.

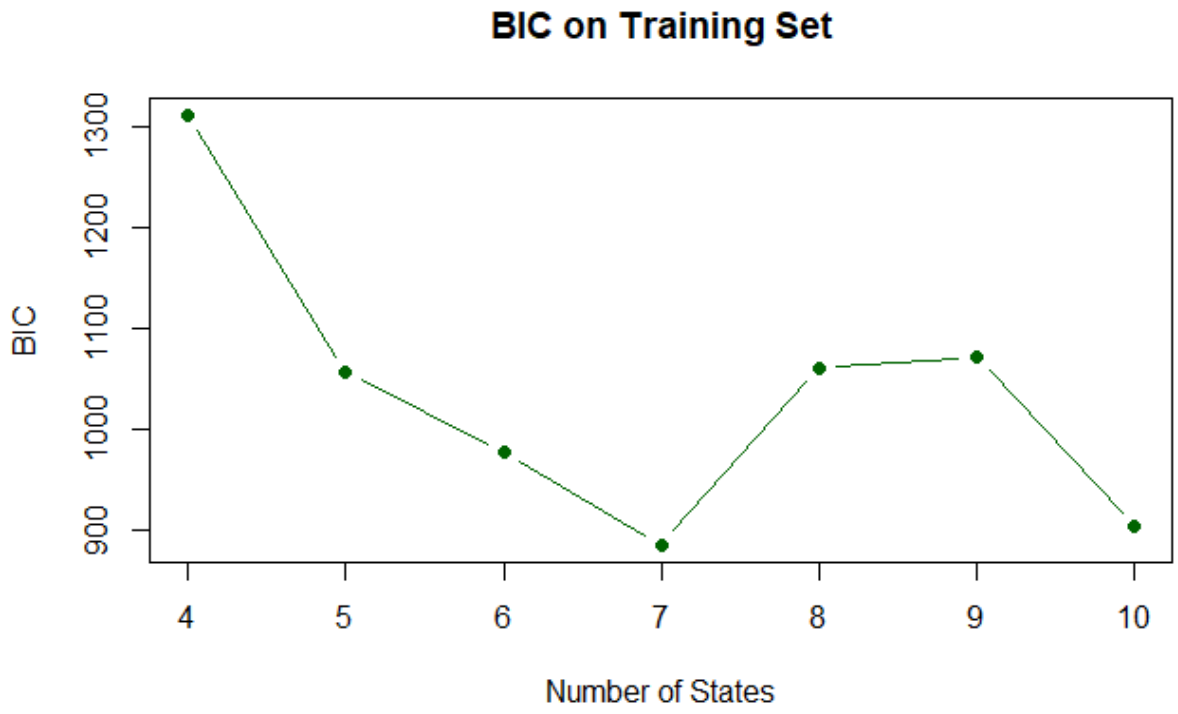
	Variable	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Global_active_power	Global_active_power	0.4688785	0.13622179	0.0880355908	0.07094699	0.25355859	0.77172439	0.29826548
Global_reactive_power	Global_reactive_power	0.1951502	0.74451033	0.1673110215	0.60604600	0.07267155	0.03410962	0.07675561
Voltage	Voltage	0.3289728	0.12852114	0.0374643513	0.14069809	0.91943574	0.07566917	0.05451435
Global_intensity	Global_intensity	0.5599063	0.02045491	0.0005624233	0.06473108	0.13852708	0.08217176	0.80990997
Sub_metering_1	Sub_metering_1	0.2990863	0.12718445	0.7279041221	0.47972092	0.04197444	0.23999384	0.27392710
Sub_metering_2	Sub_metering_2	0.2840618	0.41326932	0.6511671633	0.42663103	0.05737591	0.28660617	0.23885056
Sub_metering_3	Sub_metering_3	0.3875662	0.47248755	0.0948144367	0.43765570	0.24661956	0.50105871	0.33783821

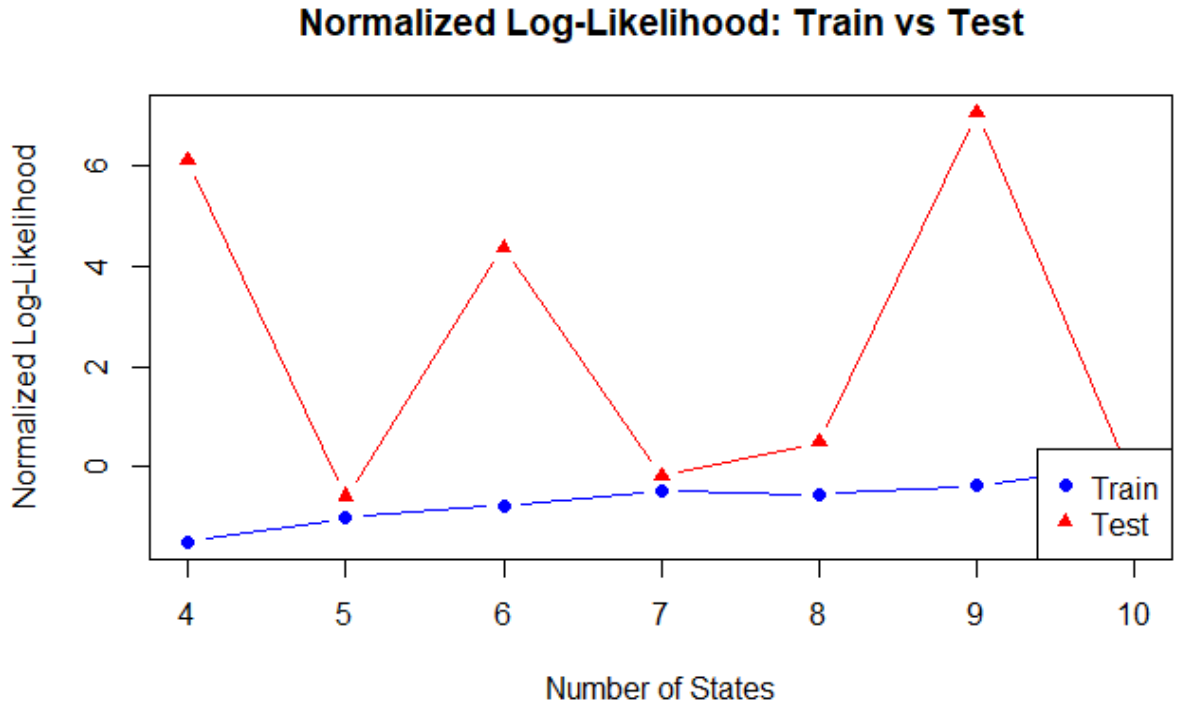
We select the variables that consistently contribute across the first three PC values. For PC1, we pick ‘Global_intensity’ and ‘Global_active_power’ as our variables for the HMMs. Similarly, in PC2, we can see that ‘Global_reactive_power’ is the variable that contributes the most; therefore, we also consider this to be the response variable. With these three variables, we can pretty much explain the original dataset by projecting every data point onto this three-dimensional plane. We added a few more variables named “Sub_metering_1” and “Sub_metering_2” for further testing. We can exclude “Voltage” from anomaly detection as its loading scores are consistently low.

Choosing an observation time window

Continuing with the feature engineering, we will choose an appropriate time window that captures the normal behaviour of household electricity usage. Intuitively, we can assume that the electricity consumption of an average household will be less on the weekdays than on the weekends, as people tend to work jobs on the weekdays and stay home on the weekends. We will analyze the behaviour of the window frame of random choice from 2 PM to 8 PM on a Sunday, July 1st, 2007.

HMM Training and Testing



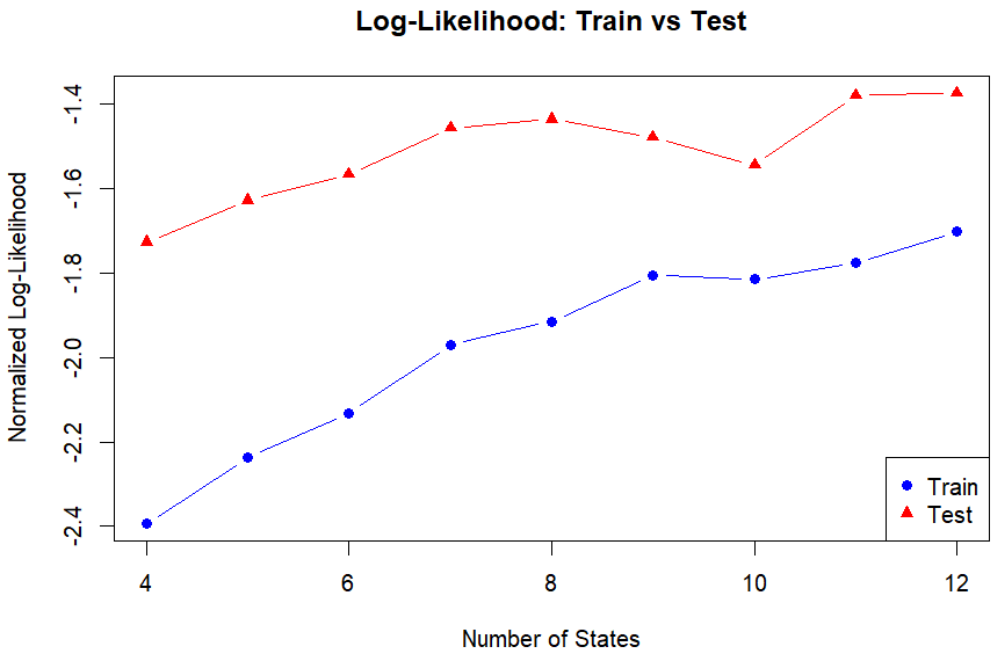


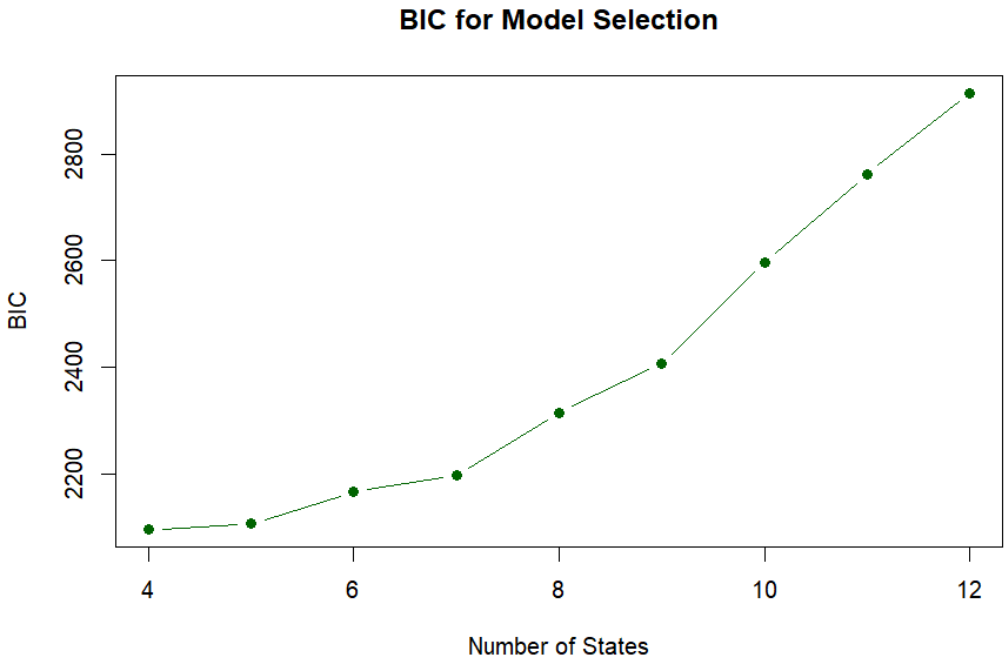
We have separated the training and testing datasets by year: 2006, 2007, and 2008 for the training dataset; meanwhile, we reserved the year 2009 for testing. We then randomly choose a time window between 2 PM and 8 PM for a random weekend day, July 1st, 2007, for training, and July 1st, 2009, for testing. Based on the PCA result, we used a loop to train multivariate HMM models with three features from PC1 and PC3 (Global_active_power, Global_intensity, and Global_reactive_power), accounting for most of the variance. From 4 to 10 states, we obtained the best result at 7 states according to the normalized log-likelihood. We only run from 4 to 10 states because the model tends to become too complex when trying to run for 11 or 14 states, resulting in an error.

We tried discretizing the data, but the results were not desired. The BIC values increase as the number of states increases. This can be explained as when we discretize the continuous data into its corresponding numeric value, the information has been lost considerably. That leads the model to be overfitting with the new discrete patterns. The BIC value is calculated as

$$BIC = -2 * loglikelihood + k * log(n)$$

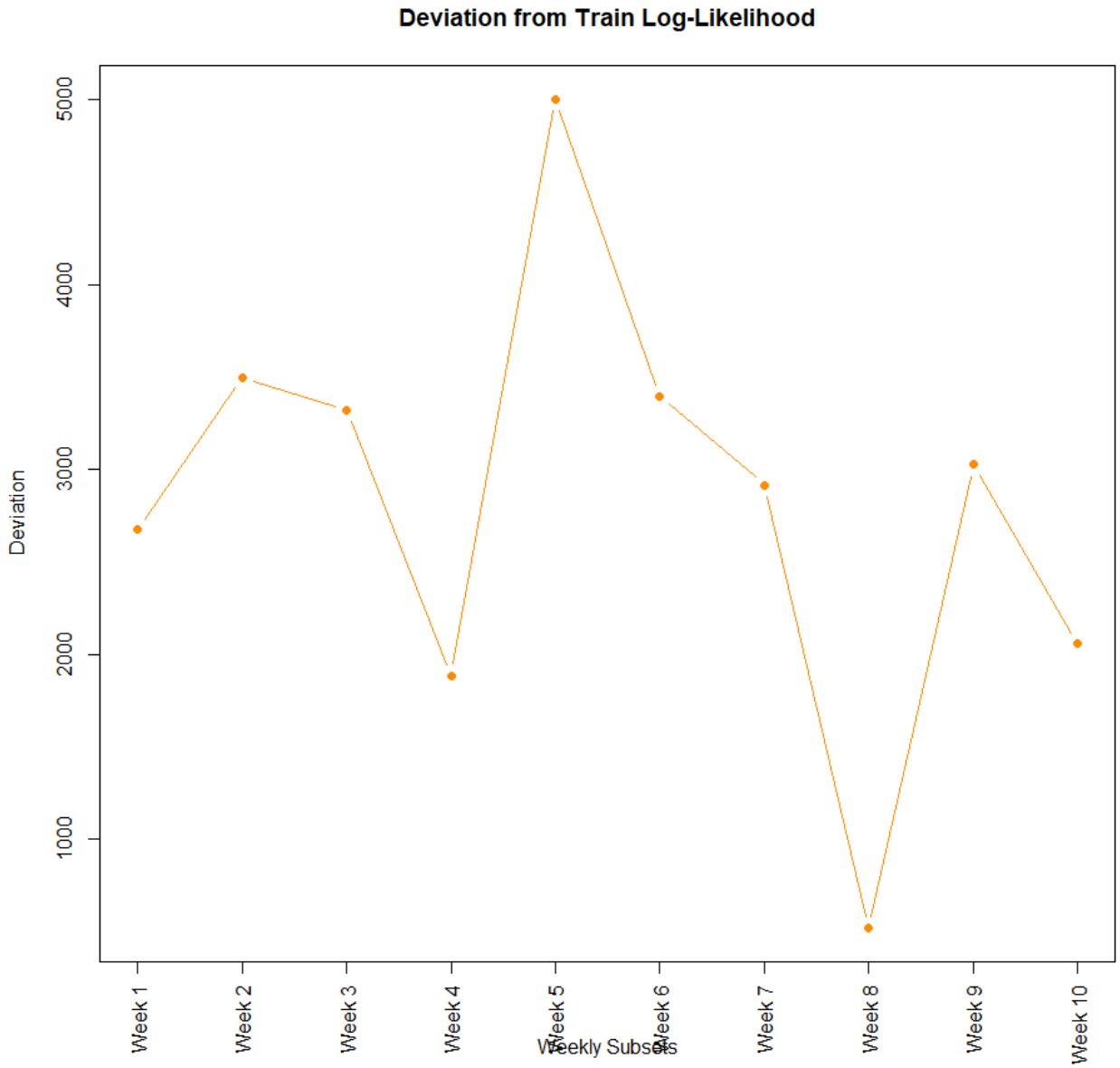
The results we get are as following graphs in the next page. We will be training our HMMs on continuous data.


Log-Likelihood: Train vs Test
Normalized Log-Likelihood
-1.4
-1.6
-1.8
-2.0
-2.2
-2.4
4
6
8
10
12
Number of States
Train
Test


BIC for Model Selection
BIC
2800
2600
2400
2200
4
6
8
10
12
Number of States

Anomaly Detection

Anomaly detection that utilises log-likelihood values shows how unlikely data can be under a normal statistical model. This allows the observer to detect deviations compared to the model's originally learned behaviour.



```
converged at iteration 27 with logLik: -2892.856
converged at iteration 50 with logLik: -3712.11
converged at iteration 53 with logLik: -3535.42
converged at iteration 46 with logLik: -2100.521
converged at iteration 62 with logLik: -5218.719
converged at iteration 53 with logLik: -3612.231
converged at iteration 101 with logLik: -3131.112
converged at iteration 74 with logLik: -733.8576
converged at iteration 37 with logLik: -3241.564
converged at iteration 59 with logLik: -2272.178
Train Log-Likelihood: -214.1301
Weekly Test Log-Likelihoods:
 [1] -2892.8561 -3712.1104 -3535.4205 -2100.5206 -5218.7193 -3612.2306 -3131.1118  -733.8576 -3241.5635 -2272.1778
Deviations from Train Log-Likelihood:
 [1] 2678.7260 3497.9803 3321.2904 1886.3905 5004.5892 3398.1005 2916.9817  519.7275 3027.4334 2058.0477
Anomaly Threshold (Max Deviation): 5004.589
```

We assume that the training data represents normal electricity consumption behaviour. When training with our best number of states, which is 7, we get a train log-likelihood as shown in the image above as -214.1031. Then, we partition our test data into 10 consecutive weeks, each week consisting of 7 days, and each day having the same time window from 2 PM to 8 PM, which is the same as our training dataset. The observed weekly log-likelihood values vary significantly, resulting in different deviation values.

We can calculate the deviation using the formula $deviation = |\ell_{train} - \ell_{test}|$ where ℓ_{train} is the training log-likelihood and ℓ_{test} is the weekly subset testing log-likelihood. The maximum deviation is computed by absolute value because we want to evaluate the difference on both upper and lower sides to capture the full range of what is considered normal behaviour, as in our given dataset. As stated above, we can use this formula to calculate the maximum deviation to find the anomaly threshold, which was found to be 5004.589 at week 5 and was chosen as the anomaly threshold.

The reason why the maximum deviation is chosen as a threshold is that by assuming the given data represents normal behaviour, anything that falls beyond that threshold

will be abnormal to our given data, thus considered an anomaly. This threshold now serves as the reference point for any unseen observations. A deviation exceeding this value would be considered an anomaly.

The plotted deviation values over ten weeks highlight fluctuations in observed deviations. Notably, week 5 shows the highest deviation at 5004.589, which may suggest a significant anomaly in the dataset in comparison to the other weeks. In contrast, week 8 has the lowest deviation at 519.7275, so we can assume that this week remains well within the normal behaviour range.

Conclusion

Over the entire course, we have learned to use the Hidden Markov Model to analyze time-series data, gaining insights into the normal behaviour of household electricity consumption data. The data collected influences a lot of how our model works as it is the variables that directly contribute to the effectiveness of the system.

Lessons Learned

- Picking an appropriate matter to the effectiveness of the model.
- Choosing an anomaly threshold for the assumed normal data set.

Problems Encountered

- **PCA Interpretation**
    - We encountered the problem of understanding the meaning of Principal Component Analysis (PCA) values generated for each variable.
    - To deal with the situation, our idea was to look for external resources that explain the use cases of principal component analysis in-depth.
    - The scree plot shows the variance explained by each PCA. It is used to determine how many PCAs we should retain to filter out the desired input variables. We keep enough PCAs to explain a high cumulative variance.
    - We then have a table with loading scores for each variable corresponding to every PCA value. We observe which variables consistently contribute most to their corresponding PCA value.

- **Anomaly Detection**
    - During our anomaly detection, we ran into cases where our model, despite using similar data, would output different numbers and graphs depending on who ran them and on what machine.
    - We had some trouble finding out the meaning of the extreme points of deviation shown in the graphs.