Detecting anomalies in electrical data using Hidden Markov Model

Technical Report

Simon Fraser University

CMPT 318: Cybersecurity

Spring 2022

Group 2

A. C. (ID: 301 448 884)

Prince, Nitafan (ID: 301 445 340)

Instructor:

Professor Uwe Glässer

Abstract

In this project, we perform cybersecurity intrusion detection by analyzing electrical data from automated supervisory system. Techniques for anomaly detection using R will be applied and discussed. By going through the important steps to build predictive model, the report outlines findings and interpretation of results from computation process.

Table of Contents

1. Introduction and Background

1.1. Introduction

With the unimaginably increasing cyber threats in today's world, critical infrastructures like electrical system, are more vulnerable to advanced persistent threats.

Cyber-physical systems like this relies on automated supervisory control which provides necessary real-time data for continuous data analysis. This can help to detect suspicious behaviors in its earlier stages and play an important step to mitigate the impact of attacks with immediate counter measurements.

To understand how this works, in this project, we explore methods to develop and perform cybersecurity intrusion detection by analyzing data from automated control processes. The analysis is built based on anomaly-detection approach.

1.2. Technical background

The given datasets consist of time series data of 2 independent variable (Dates & Times) and 7 distinct response variables as indicators of electrical consumption (Global active power, Global reactive power, Global intensity, Sub metering 1, Sub metering 2, Sub metering 3). There are 1 normal dataset of 3 years and 3 additional datasets with injected anomalies to test our trained model.

To meet the end goal of detecting anomalies from given dataset with R, we apply various techniques such as performing Principal Component Analysis; data partitioning to get training and testing datasets; train multivariate Hidden Markov Models; model selection using log-likelihood, BIC; confirm fitting of model with testing data; detect and illustrate anomalies with selected model.

More details of the techniques are discussed in Methodology section.

1.3. Team contribution

For the technical part of this project, A. C. works on Principal Component Analysis, cleaning up, filtering, and preparing training and testing datasets for model building. Prince works on model training and testing, as well as applying the model to detect anomalies.

For the report and presentation, each of us is responsible for documenting the process and findings of our own technical parts. Besides, A. C. is the main in charge to fine tune the report and slides, ensure consistency and completeness across the documents.

2. Problem

With given datasets on electrical consumption from supervisory control system, we need to come up with solution for anomaly detection by building a predictive model from normal dataset, and then using it to detect outliers (if any) of another dataset. In this case, we are given 3 anomalous datasets to test.

To tackle this, we must incorporate a thorough process to work with given data. It is important to go through careful data preparation process to make sure there is no corrupted data or missing values to use as input. Performing Principal Component analysis also supports our decision for choosing representative variables for our multivariate training model. From that, we can filter out unnecessary data and keep only relevant information to effectively interpret the data and trim down processing time. To train and select the accurate model, we also want to partition the data into training and testing datasets, select model using indicators like log-likelihood, BIC, and cross-check with testing data. The last thing is to detect the degree of anomalous in the input datasets and interpret the result.

After figuring out mentioned sub-problems, we would be able to detect anomalies with a good trained model.

3. Methodology

There are four main steps in our methodology in this project: (i) cleaning up data (ii) deciding response variables for model training; (iii) training and selecting appropriate multivariate Hidden Markov Models (iv) detecting anomalies using the trained model. Details of each step as specified below.

3.1. Clean up data

Before working with the data, we need to process the given raw data into workable dataset.

Firstly, it is important to check on N/A values to see how much values are missing. In the original dataset, there are 24 N/A values for Global active power alone, and 8348 N/A values for each of other response variables.

Date	Time	Global_active_power
0	0	24
Global_reactive_power	Voltage	Global_intensity
8348	8348	8348
Sub_metering_1	Sub_metering_2	Sub_metering_3
8348	8348	8348

Table 1: *Number of N/A values in each variable*

From this, we handle missing data points by replacing them using predictive mean matching method, which selects the closest observed values to our missing values.

Secondly, to make the response variables comparable to one another, we need to scale the completed dataset. We then partition scaled datasets into 2 parts: training data and testing data. Specifically, we would use bigger portion to be training data to help in building a more accurate model.

The last step is to specify chosen time series to examine. Due to the scope of this project and time constraint, instead of looking at a whole dataset, we filter datasets to only look at one specific day in a week throughout the years within 3 hours time frame to examine.

3.2. Decide response variables for model training

Before building model, we want to select a minimal amount of response variables that still allow us to interpret representative information. To achieve this goal, we examine the correlation between response variables by performing a Principal Component Analysis (PCA) on the training dataset.

Since our dataset has non-numeric data (Date & Time), we create a new dataset filtering out these variables and use this updated dataset in PCA function with scaling to standardize the variance across variables. Without scaling, the high variance variables will be over-represented.

From PCA results which consist of 7 principal components (PCs), we will look at some indicators and examine them: (i) proportion of variance of each PC, scree plot and Kaiser-Guttman Criterion to decide which PC should be chosen for further break down (ii) absolute values of loading scores in each PC, and biplot to decide the response variables with greatest contribution to our dataset.

In the end, we choose only 2 variables to build our multivariate model since choosing 3 variables slowed down our computing process and can even crash RStudio.

3.3. Train and select appropriate Hidden Markov Models

In training the Hidden Markov Models for this project, the range of number of states for the models we are to simulate are 4 to 24. However, due to a long running time to calculate each model, we choose to run simulations on states 4, 8, 12, 16, 20, and 24. We also decided our model to be ‘gaussian()’ (variable 1) and ‘gaussian()’ (variable 2) for the depmix() function parameter ‘family = list(gaussian(), gaussian())’ rather than ‘multinomial(“identity”)’. While perhaps ‘multinomial(“identity”)’ may work better in calculating the fit of the data set, our decision to not use it was made because the parameter would crash our hardware, produce a different number of parameters between the test and training models, and largely increase the computation running time. Also, we will test multiple seeds to see which would have the most models converged.

We will utilize the fit() function of depmixS4 to fit the training models with the training data set. This will give us the log likelihoods and BIC values, along with the parameters that will be needed for the test models. However, due to different data lengths, we will normalize the log likelihoods and BIC values by dividing them by the length of the training data set. Then a dual-lined graph will be created by the log likelihood, and we will determine a range of candidate models from the graph with good correlations between their log-likelihoods and BIC values. The closer the log likelihood and BIC values are to each other, the better the model's fit could be.

In testing the test data set, we will create test models by copying the parameters from the fitted train models through the function setpars().We will then use the forwarbackward() function to test the test models and get the log likelihoods, in which we will normalize by dividing by the length of the test data set. To find the best model, a calculation of the difference of the test log likelihood and the train log likelihood will be done. The model with the smallest difference is the best candidate.

3.4. Detect anomalies with trained model

To prepare the anomaly data sets: (i) we replace NA values through predictive mean matching method, (ii) scale the data values, and then (iii) filter the predetermined time window.

In simulating the models with the anomaly data sets, the procedure we use is like the one we had done in testing the models. We create 3 test models for each anomaly data set and copy the best training model's parameters, use forwardbackward() function (the forward backward algorithm) to simulate the models for each anomalous data sets and get the log likelihoods. The normalization of the values is done by dividing by the lengths of their respective anomaly data sets. We will then calculate the differences of the log likelihoods between the anomaly and training models. We consider the model simulation with the highest log likelihood difference as the most anomalous data set.

4. Results & discussion

4.1. Principal Component Analysis for variable selection

- **Select necessary components (PCs) from PCA result:**

To interpret the result, first thing we want to do is reducing our dimension to focus on important PCs and omit less necessary PCs.

For that reason, we will first look at the **proportion of variance**.

```
Importance of components:
                          PC1    PC2    PC3    PC4     PC5     PC6     PC7
Standard deviation     1.7172 0.9984 0.9707 0.9194 0.81793 0.68520 0.35805
Proportion of Variance 0.4212 0.1424 0.1346 0.1208 0.09557 0.06707 0.01831
Cumulative Proportion  0.4212 0.5636 0.6983 0.8190 0.91461 0.98169 1.00000
```

Table 2: *Summary of Principal Component Analysis*

Each component is determined because it accounts for the most variance, and components are ranked by greatest to smallest portion of variance accounted for. For example, PC1 will account for the most variance (in this case: 42.1%) while PC2 will account for second most variance (14.2%) and so forth.

We can also look at **scree plot** and see the same trend.

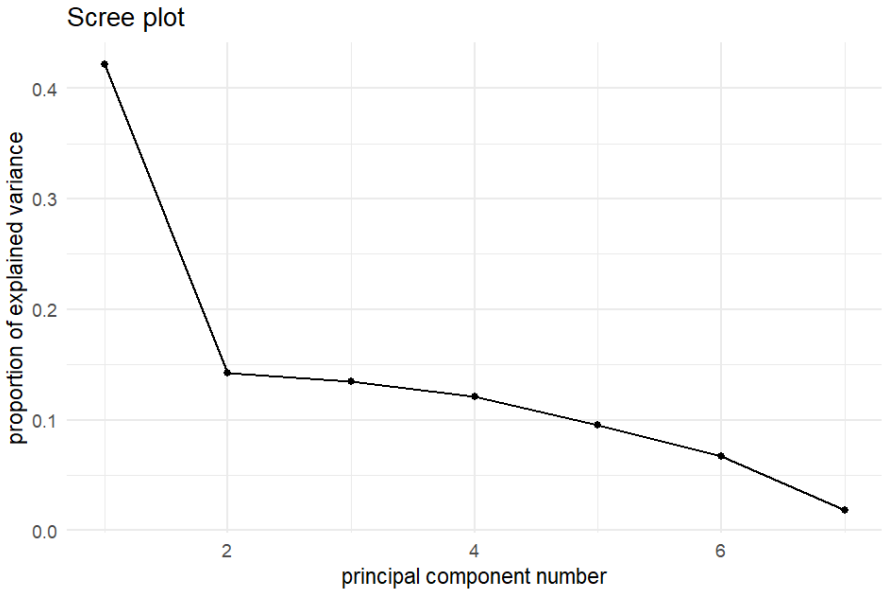

Figure 1: *Scree plot of Principle Component Analysis*

The scree plot above displays each PC on x-axis and importance of them on y-axis. After the first component (PC1), there is a significant drop of each additional component.

An additional evaluation we want to look at is **Kaiser-Guttman Criterion**, where we only keep components for which Eigenvalue is greater than 1. This is because if the Eigenvalue is less than 1, the component accounts for less variance than a single variable contributed.

To compute Eigenvalue per component, we will find standard deviation squared of each component as below. We can see that only PC1 is greater than 1.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Eigenvalue $(= s.dev^2)$	2.9486979	0.9968665	0.9423549	0.8453657	0.6690096	0.4695045	0.1282011

Table 3: *Eigenvalues of each principal components*

The above methods show us that it is enough to use the only first component, PC1, for further examination.

- **Select response variables based on its contribution**

After that, we consider the contribution of each variable in each PC by looking at **loading scores** as presented below. As selected in previous steps, we will examine only loading scores in PC1

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Global_active_power	-0.4677140	0.117268015	-0.083445684	-0.08627315	0.25054806	-0.76146126	-0.33239735
Global_reactive_power	-0.1765327	-0.780092300	0.147482368	0.57414552	0.01481733	-0.05532367	-0.07495206
Voltage	0.3760045	-0.170153791	-0.005137842	-0.12519058	0.89841831	0.01803809	0.08055061
Global_intensity	-0.5511229	0.008431065	0.013431388	-0.05498964	0.15059118	0.03709679	0.81788405
Sub_metering_1	-0.2817720	-0.115159510	0.758943154	-0.44826065	0.05561865	0.25167156	-0.25293968
Sub_metering_2	-0.2810601	-0.401908643	-0.621697452	-0.46985786	-0.01439783	0.32279958	-0.21861760
Sub_metering_3	-0.3828494	0.416999461	-0.092609596	0.47171317	0.32229214	0.49787182	-0.31096515

Table 4*: Loading scores of each response variables in each PC*

From the table above, we can see two greatest loading scores (consider only absolute values) are Global intensity (0.55), Global active power (0.46). This means these two response variables largely represents electrical consumption in our data set.

A **biplots** below also illustrates the contribution of response variables on our data and visualize how the data is spread out in dataset.

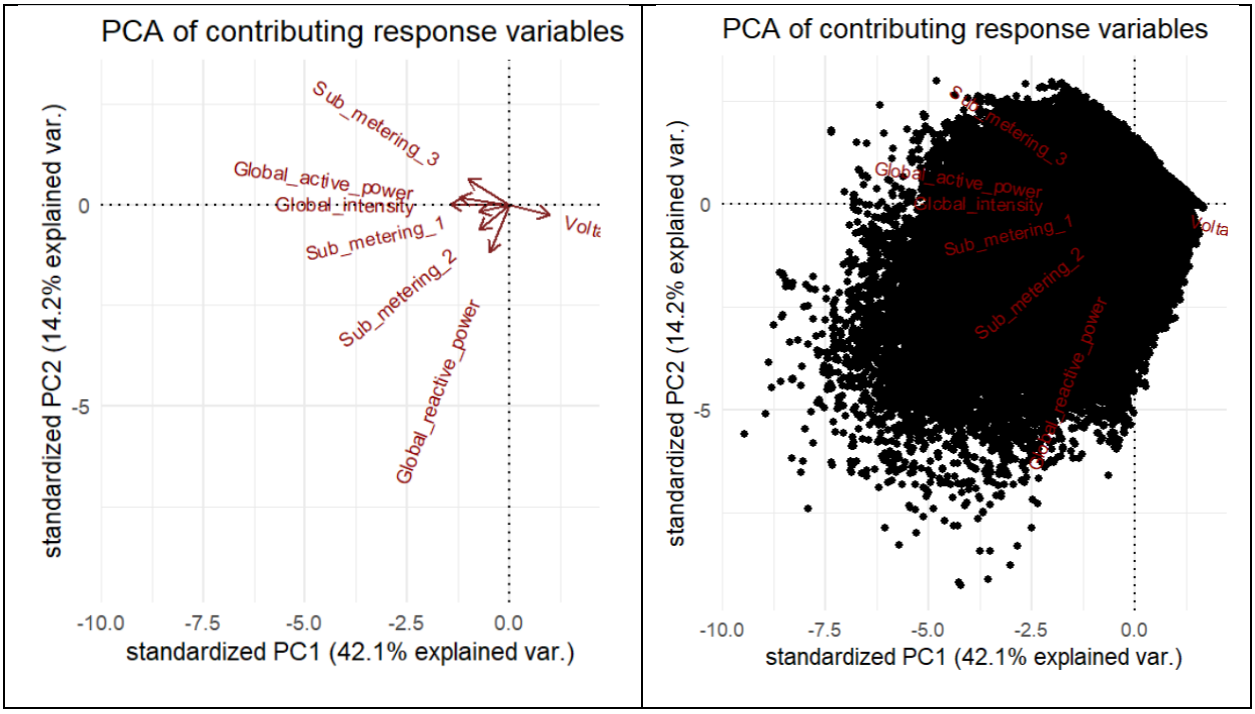

Figure 2 & 3: *Biplots of PCA without data points (left) and with data points (right)*

Again, Global intensity, and Global active power are two dimensions strongly influences a principal component. The modulus of each of these two vectors are also the greatest among all vectors.

At first, we picked the greatest three variables from PCA result. However, after using all three variables to create multivariate model, we realized that it slowed down our computing process, and could possibly crash the program when training HMM.

Therefore, we narrowed down to pick two response variables only: **Global intensity** and **Global active power**. These variables can represent the changes and trends happening to our data.

4.2. Time window selection & data partition

- **Time window selection:**

As mentioned earlier, in the scope of our project and resources, we choose to look at only a specific day within a time frame instead of the full dataset. In this case, we select every Wednesday from 9:00am to 11:59am to observe a normal weekday electrical consumption, although we acknowledge that Monday and Tuesday both have one more day of data points compared to Wednesday and the rest of the week.

The reason why we keep this time series is because we already had the same time window in previous assignment, so it would be a good opportunity to compare our findings with previous experience.

- **Data partition:**

In the data partitioning process, we choose training dataset consists of data running from 16/12/2006 to 16/12/2008 (104 Wednesdays), and the testing data is from 17/12/2008 to 1/12/2009 (50 Wednesdays). Ideally, we would want to separate data with approximately 80:20 ratio with training data as a bigger portion.

But in the context of our data based on time series, preserving the data as a whole year would help to keep the context right. Thus, we decided to have the first 2 years of data for training dataset, and the last 1 year (with the last 1 weeks and 5 days missing to be a full year) for testing dataset.

4.3. Model selection using log-likelihood and BIC values of different numbers of HMM states

While following our methodology in training Hidden Markov Models, we tested seeds 1 to 15 to run simulations before deciding on seed 5 since it converged models with number of states 4, 8, 12, 16 and 20. However, we found that the model with 24 states would not converge despite the numerous runs we conducted with various seeds. Therefore, we tried simulating training models with states above 20, but unfortunately, models 21 to 23 would not converge with seed 5.

Since the length of the training data set consist of 104 Wednesdays, we normalized the produced log likelihoods and BIC values by dividing by 104. We then constructed a graph based on the chart "Graph Values" below. After examining the graph, we decided that the models with 6-8 states are our testing candidates since they are the nearest to the intersection of the graph where the loglikelihood and BIC values are closest to each other.

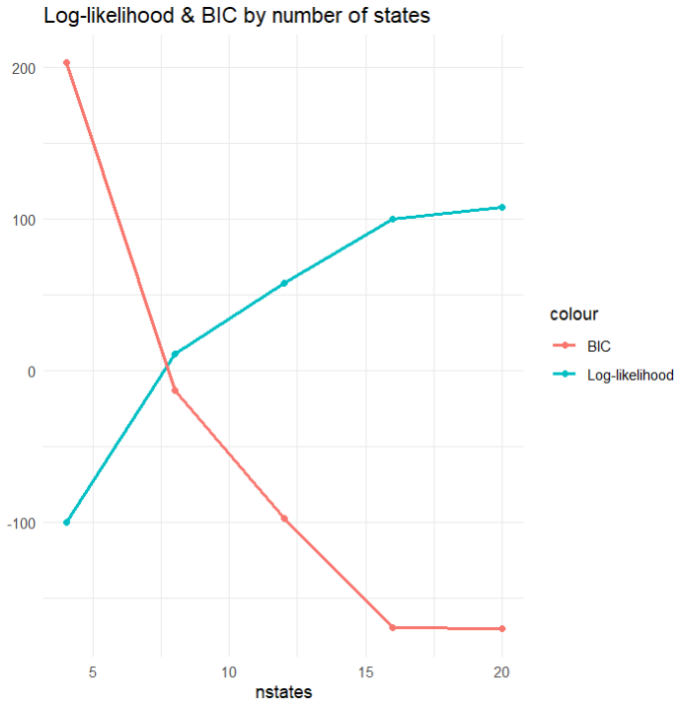

Figure 4: *Log-likelihood & BIC by number of states*

n_states	log_likelihood	BIC_value
4	-99.99519	202.92266
8	11.08848	-13.19092
12	57.59502	-97.12338
16	99.84789	-169.52162
20	107.61025	-169.91194

Table 5: *Graph values*

n_states	log_likelihood	BIC_value
6	-39.61315	84.80711
7	-14.92983	37.04850
8	11.08848	-13.19092

Table 6: *Candidate values*

4.4. Confirm fitting of model to data by comparing normalized log-likelihood of training and testing data

Since the test data set contains 50 Wednesdays, we normalize the log likelihoods produced from the test models by dividing by 50.

When comparing the log likelihood outputs from training ("train_logLik") and testing ("test_logLik"), we decided to choose the model with 7 states as it has the smallest difference out of the 3 candidates on training and testing, while being close to the intersection in the graph.

n_states	test_logLik	train_logLik	difference
6	-116.01960	-39.61315	-76.40645
7	-84.65469	-14.92983	-69.72485
8	-69.80528	11.08848	-80.89376

Table 7: *Differences of each model's training and testing log likelihoods*

4.5. Detect the degree of anomalous with the chosen model in 3 anomalous datasets

Each anomalous data set contains 52 Wednesdays, so we normalize the log likelihoods produced from the anomaly models by dividing by 52.

After following our anomaly testing methodology and calculating the differences, we found that anomalous data set 3 has the highest difference between its log likelihood and the train loglikelihood. Therefore, data set 3 can be considered the most anomalous as, in comparison to the other 2 anomalous data sets, its loglikelihood is around 21% larger. The least anomalous is data set 1 as it has the least difference value, while data set 2 is the second most anomalous, albeit just slightly more than the first set. The table below illustrates the outcome of the anomaly models.

anomaly_set	anomaly_logLik	train_logLik	diff_subtract
1	-313.0941	-14.92983	-298.1642
2	-313.3981	-14.92983	-298.4682
3	-380.7981	-14.92983	-365.8683

Table 8: Differences of anomaly data set's log likelihoods and training log-likelihood

5. Limitations

One major challenge that can affect the accuracy of our anomaly detection method is dealing with unlabeled data. Since we can only make implicit assumption that a normal dataset contains very few anomalies, so we want to use it for model training. However, if this is not true, we could yield very high false alarm rate. There is also a lack of historic data to better understand the context.

Besides, the raw data also contains missing values that we had to populate with approximation. And the last year in our dataset doesn't cover a full year. These could somehow impact our solution as well.

On the other hand, due to limitation of long computation time, we could not consider a larger amount of data points or take more variable parameters for better accuracy. For example, we must narrow down to 2 response variables for our training model to run; and choose a specific time window instead of look at the whole dataset. For the same reason on computation issue, we had to choose Gaussian distribution as parameter for model training instead of Multinomial distribution, since selecting Multinomial does not give us the result in a reasonable amount of time.

6. Conclusion

In this project, we have examined basic techniques for unsupervised anomaly detection using R.

This includes learning how to prepare data for analysis, perform and interpret PCA result to minimalize the amount of necessary response variables, build and select multivariable Hidden Markov Models using time-series datasets extracted from supervisory control system, as well as testing the result with anomalous datasets.

We also encountered challenges in working with real-life data in this project due to imperfections of original data (unlabelled, missing values, etc) and long computation process. A larger scope of this project with more time, data, and computation resources can partially help to tackle these issues and potentially produce a more accurate solution to our problem.

7. References

AGRON Stats. "Biplot of PCs Using Ggbiplot Function." *AGRON Stats*, 4 July 2021, https://agroninfotech.blogspot.com/2020/06/biplot-for-principal-component-analysis.html.

Alto, Valentina. "PCA: Eigenvectors and Eigenvalues." *Towards Data Science*, Towards Data Science, 13 July 2019, https://towardsdatascience.com/pca-eigenvectors-and-eigenvalues-1f968bc6777a.

"Ggbiplot: Biplot for Principal Components Using GGPLOT2." *RDocumentation*, https://www.rdocumentation.org/packages/ggbiplot/versions/0.55/topics/ggbiplot.

Madiraju, Nischal. "Splitting a Dataset." *Towards Data Science*, Towards Data Science, 14 June 2020, https://towardsdatascience.com/splitting-a-dataset-e328dab2760a.

Michy, Alice. "Imputing Missing Data with R; Mice Package." *DataScience+*, 14 Mar. 2018, https://datascienceplus.com/imputing-missing-data-with-r-mice-package/.

"PRCOMP: Principal Components Analysis." *RDocumentation*, https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/prcomp.

Wood, Robert. "Learn Principal Component Analysis in R." *Towards Data Science*, Towards Data Science, 21 May 2020, https://towardsdatascience.com/learn-principle-component-analysis-in-r-ddba7c9b1064.