

Anomaly Data Detection

CMPT 318
Term Project Report
Group 19
Spring 2022

Yihao Wang - 301354603
Jordan Yee - 301370132
Wesam Shalabi - 301392844

The main problem: Cybersecurity is an ever-growing industry with many everyday systems under constant threat. To combat these lurking threats we must first figure out when our systems have been breached. The faster we find out the less damage that can be done. Keep in mind that systems will never be completely safe as breaches are inevitable. In this report, we try to determine if the electrical grid has been affected by anomalies and which of them have had the greatest effect.

Abstract: In order to do this we must figure out the normal behavior of a previously observed set of data. First, by partitioning our data into 2 sets, train and test. Then we create a hidden Markov model using the train data which will represent the normal behavior. Lastly, we use this model on the test data set to determine if the model gives an accurate representation of the normal behavior. When we have determined that the model is a good fit we can use this model to predict against new incoming observations to determine if anomalies are present.

Table of Contents

Table of Contents	2
1. Part1	3
2. Part2	5
3. Part3	7
4. Conclusion	8
5. References	11

Table of Figures

Table of Figures	2
PCA	3
Plot of log-likelihood and BIC	6
Log-likelihood of the test set	7
Log-likelihood of three sets	8

1. Part1

Comparing each response variable relative to the other ones as the result of performing PCA and illustrating the rationale for our final choice of variables:

Before we start PCA, we first need to scale the raw data. Because scale function cannot work with NA values, we choose to replace NA values with predicted values (made by mice function). That is because considering median/mean can probably be an outlier which will have an unexpected impact on our result of PCA. Using predicted values can reduce this potential risk so that the original NA values don't have observable weight on PCA.

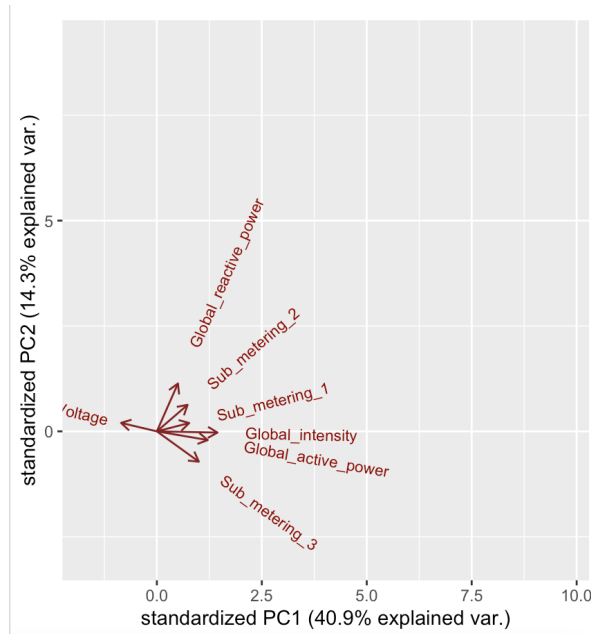
```
> summary(pcaM)
Importance of components:
               PC1    PC2    PC3    PC4    PC5    PC6    PC7
Standard deviation  1.6873 0.9965 0.9672 0.9109 0.8756 0.6858 0.35457
Proportion of Variance 0.4086 0.1425 0.1343 0.1191 0.1100 0.0675 0.01804
Cumulative Proportion 0.4086 0.5511 0.6854 0.8044 0.9145 0.9820 1.00000
```

We find that the first PC occupied about 40.86% of the total variance in the dataset. And the rest of the PCs are substantially less important than the first PC.

```
> print(pcaM$rotation)
               PC1    PC2    PC3    PC4    PC5    PC6    PC7
Global_active_power  0.4698787 -0.13566565 -0.087860425 -0.06938224  0.26329840 -0.76831712  0.29772177
Global_reactive_power 0.1945370  0.74452243  0.165366826  0.60751334  0.06613247 -0.03382081  0.07684850
Voltage             -0.3302400  0.13166579 -0.035840891 -0.13408040  0.91875887  0.08407414 -0.05618887
Global_intensity     0.5592875 -0.01887903  0.001319803 -0.06405429  0.13719021  0.08396741 -0.81047192
Sub_metering_1       0.2985694  0.12959647  0.728321937 -0.47855459  0.04209348  0.24025667  0.27404124
Sub_metering_2       0.2835704  0.41249803 -0.651516183 -0.42708102 -0.05656774  0.28684953  0.23891237
Sub_metering_3       0.3872371 -0.47184638 -0.093390934  0.43888546  0.24163952  0.50440068  0.33653590
```

After we find the most important PC, we look at the rotation of each feature in the PC1. The rotation of the PC shows how the variables contribute to the PCA axes. Positive values

indicate positive relationships and negative values indicate negative relationships. If a feature has a large absolute value in PC1, then it is more important to us. So we select Global_active_power and Global_intensity as the subset of the response variables for the training of multivariate Hidden Markov models on normal electricity consumption data. And here is the data plot of PCA:



Explaining the selection of response variables and observation time window

chosen for the analysis:

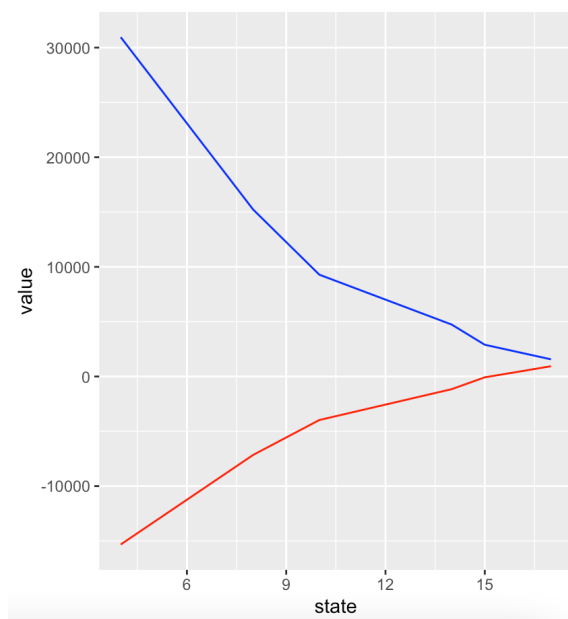
We select Global_active_power and Global_intensity as the subset of the response variables because these two contribute the most to the rotation of the PC1. The rotation of the PC shows how the variables contribute to the PCA axes. Positive values indicate positive relationships and negative values indicate negative relationships. If a feature has a large absolute value in PC1, then it is more important to us.

For the selection of the observation time window, we choose Monday 18-20. The reason is that this time window gives us a reasonable log-likelihood trend. The log-likelihood only becomes positive after state = 16. And when the state = 15, the log-likelihood is very close to

0 which means the model fits the data very well. And the BIC trend of this time window decreases all the time between state =4 and state = 24. This is also a good thing because when state = 15, the BIC is low enough to believe our model doesn't overfit the data.

2. Part2

An overview of the log-likelihood and BIC values for different numbers of HMM states to justify our final selection of model(s):












(figure for log-likelihood & BIC for different numbers of HMM states)

Overall, we found the log-likelihood increases from -15000 when the number of HMM states is 4. The BIC decreases from 30000 when the number of HMM states is 4. When the number of HMM states increases to 16, the log-likelihood keeps approaching 0 and the BIC keeps decreasing to a minimum. When the state = 15, the log-likelihood is -5 which is very good for the training set. Because the log-likelihood shows how well our HMM fits the data

and 0 means our model exactly fits the training data. And the BIC is low enough to show our model doesn't overfit the data. Since the log-likelihood becomes greater than 0 after state ≥ 16 , I choose to not test higher numbers of states. There will be a global minimum after states ≥ 16 for BIC.

Explaining the choice for partitioning the data into train data and test data:

We divide the data into a training set which is from 2006-12-16 to 2009-06-08 (15600 data points) and a testing set which is from 2009-06-08 to 2009-12-01 (3000 data points). The reason is that the training set should occupy the majority of the original data set in order to learn the model. Usually, people divide the data as 80% of the training set and 20% of the testing set. Our group followed this style and our training set occupied 83% of the total data.

▶ df	1556444 obs. of 9 variables	
▶ df8	15600 obs. of 9 variables	
▶ dfanomaly1	6120 obs. of 9 variables	
▶ dfanomaly2	6120 obs. of 9 variables	
▶ dfanomaly3	6120 obs. of 9 variables	
▶ dfcompare	5 obs. of 2 variables	
▶ dfM	Large mids (22 elements, 92.7 MB)	
▶ dfplot	7 obs. of 3 variables	
▶ dfctest	3000 obs. of 9 variables	

(figure for the training set and the testing set)

(df8 is the train set and dfctest is the test set)

Comparing the normalized training log-likelihood and test log-likelihood in order to show how good our model fits the data:

When normalizing our log-likelihood for the training and testing we chose to divide the log-likelihood of each by the number of days. We went with this option since we already filled our data with predicted values (made by mice()). This allowed us to choose 130 days for the train and 25 days for the test.

	models	logLike
1	train	-0.5831291
2	test	-31.1088868

(figure for the log of the training set and the testing set)

We can see that the log-likelihood of the test set is higher than the log-likelihood of the train set. But the difference is expected because we only trained our model based on the train data. And the log-likelihood of the test set is also small enough, that we can tell our model is not overfit. If our model is overfitted, then the test data set would have a very small log-likelihood. And if our model was underfitted, then it would also have a very small log-likelihood.

3. Part3

Illustrating anomalies in the three different data sets with injected anomalies:

	models	logLike
1	train	-0.5831291
2	test	-31.1088868
3	anomaly1	-3287.6447006
4	anomaly2	-3287.6446961
5	anomaly3	NaN

(figure for the log of all the sets)

After we apply the HMM which was trained before onto those three new data sets. We compared the normalized log-likelihood of them. We find that the new data set 1 and the

new data set 2 have anomalously large log-likelihood. The normalized log-likelihood of these two sets are both **100** times larger than the normalized log-likelihood of the test set. Which is enough to say they are anomaly data. And the third new data set even can't be applied using our HMM. Then I checked the mean of the Global active power between the train set and this third data set.

```
> mean(dfanomaly3$Global_active_power)
[1] -1.895239e-17
> mean(df8$Global_active_power)
[1] 0.1424111
```

You can tell the mean of the Global active power of the third data set is 10^{16} times larger than the mean of the Global active power of the train set. Obviously, our model can't fit this anomaly data set because our model was trained on the train set. Which makes DataWithAnomalies3 the most anomalous data. DataWithAnomalies1 and DataWithAnomalies2 are similar considering the level of anomalous.

As a result, the rank of anomalous is the following:

DataWithAnomalies3 >> DataWithAnomalies1 >= DataWithAnomalies2

4. Conclusion

Accomplishment:

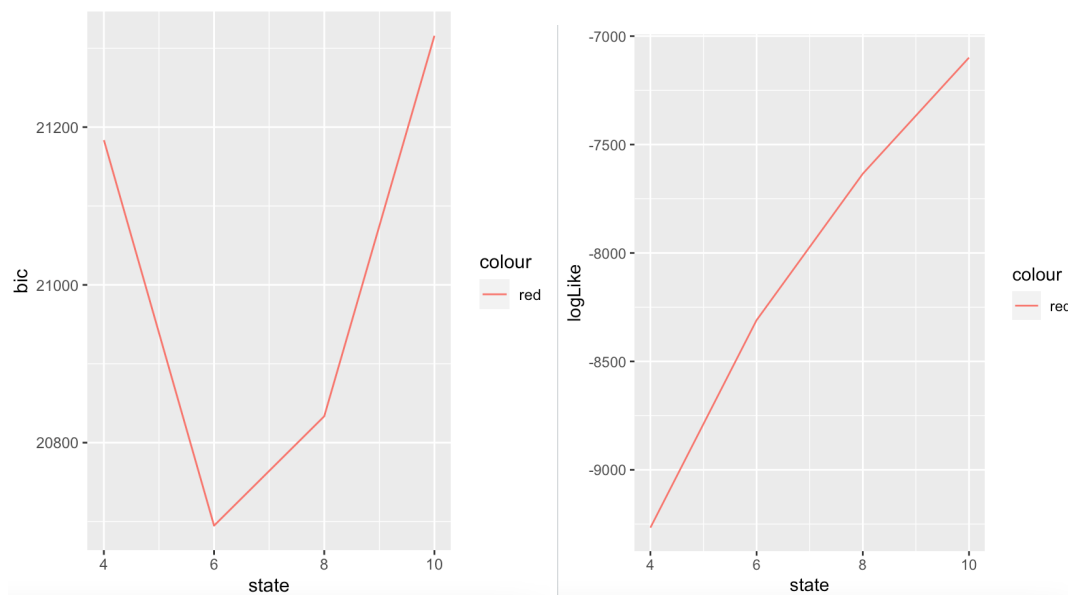
We successfully learned the parameters of interest of the hidden Markov model by training the model with the training set and evaluating the performance of different family lists, time-slots, and the number of states.

We successfully capture the appropriate hidden Markov model by comparing the log-likelihood of the training set and the testing set to avoid over and underfitting.

We successfully detected the three anomaly data sets by using the hidden Markov model we computed from the observed data sets and by comparing the log-likelihood between them.

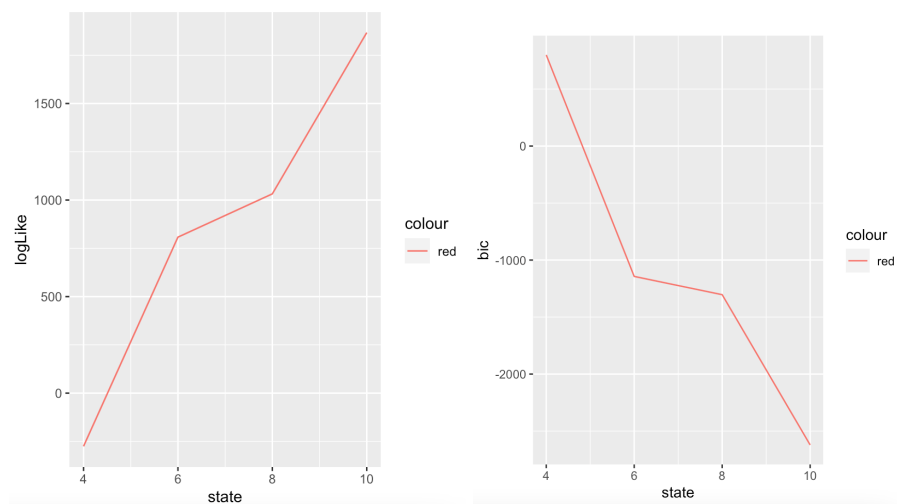
Lessons learned (the major problems encountered):

Lesson 1 - choosing the family list: I created all four kinds of lists: `list(gaussian(), multinomial("identity"))`, `list(gaussian(), gaussian())`, `list(multinomial("identity"), gaussian())` and `list(multinomial("identity"), multinomial("identity"))` to test on a time window. The most suitable list is actually the `list(gaussian(), multinomial("identity"))`, here is the plot:



You can tell there is a global minimum for BIC and the log-likelihood is negative for all states. However, when we use `list(gaussian(), multinomial("identity"))`, `setpars(training model, getpars(new data))` will produce an error. We tried different time windows, different portions of the train set, and different numbers of states. None of these changes solved the error. So we had to use `list(gaussian(), gaussian())`. The benefit is that the gaussian runs much faster than the multinomial.

Lesson 2 - choosing the time window: Initially, I chose 9-11 AM Monday, and here is the plot:



The log-likelihood starts to become greater than 0 very early. When state = 5, the log-likelihood becomes about 250. So we gave up this time window and tried many other time windows. Some of them still have positive log-likelihood very early on and some of them have very low log-likelihood even when state = 24. When we found 18-20 Monday, we deemed the log-likelihood and BIC trend to be ideal.

Lesson 3 - Using mice() to fill the NA values: Initially, I created a function to replace the NA values with their mean. But later, I realized there is a risk that the

mean can be an outlier. In the end, I used mice() function with the Predictive Mean Matching method to replace NA values.

Lesson 4 - Detecting the third data set: At first, we get real value log-likelihood for the other two data sets but NaN value for the data set 3. Which means that the data set 3 can't converge using the model. I'm wondering if it's because our model uses a high number of states which is 15 or maybe the family list we use. I tried using different numbers of states and different family lists. None of them solved the problem. Then I realized that data set 3 has a very large mean compared to the train set (10^{16} times larger). It turns out the problem is related to the magnitude of the data set 3. Since scaling can help with the gradient descent and the gradient descent is an algorithm used to find the local minimum of certain functions, the speed of convergence will be improved. After I scaled the data set 3 (just to try and understand), the train HMM could converge and it gave an anomalous big log-likelihood.

Contributions:

Yihao: 75% (the whole code and report, solving unexpected code errors and theoretical stuff. Attending office hour.)

Jordan: 25% (testing for time windows, Code for anomaly comparisons and test data comparisons, editing and partial writing of the report. Attending office hour.)

Wesam: 0% (not participant with the group at all. Did nothing to the code and report, only became concern about our report until April 3)

5. References

Visser, I., & Speekenbrink, M. (2021, May 12). Package 'depmixs4' . Retrieved April 3, 2022, from <https://cran.r-project.org/web/packages/depmixS4/depmixS4.pdf>