Delft University of Technology

Cyber-Data Analytics
CS4035

Assignment 2: Group 4

Anomaly Detection

Authors:
Navin Raj Prabhu - 4764722

Thomas Pfann - 4102371

June 4, 2019

1 Familiarization of Dataset

1.1 Types of signals

In this assignment we try to predict whether anomalies in the BATADAL data set are part of an (ongoing) attack, or should be disregarded. To find anomalies in traffic data, there are several methods suggested. We focus mostly on ARMA, Discreet Markov Models, PCA, and finally a Deep Neural Network. From the BATADAL website, we have 2 data sets, both containing measurement data from several sensors.

- Water tank levels: L_T
- Pressure: P
- Flow levels (pumps and valves): $F - PU$, $F - V$

There are 31 sensor signals, and another 13 actuator signals which show whether a valve is open or closed.

1.2 Correlation of signals and cyclic behaviour

As can be seen in 1, there are some signals that have quite a high correlation with each other (note that darker blue is a higher correlation). Some signals have been removes if they did not show any correlation what so ever across the board. From the image we can see that pressure signals tend to be correlated with water flow signals under normal circumstances.

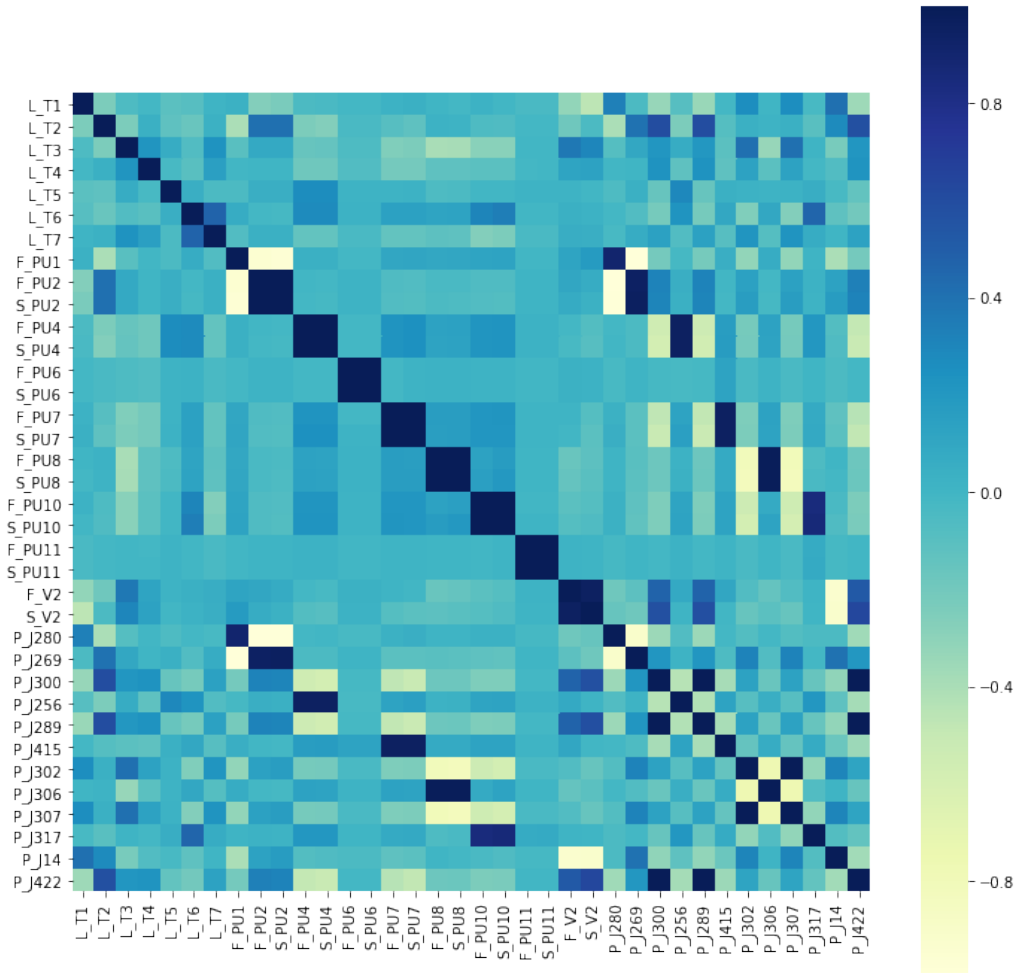
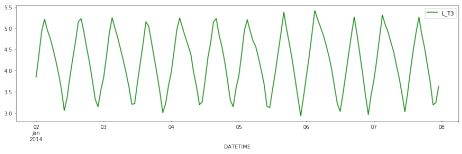
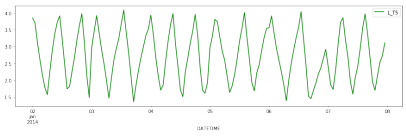


Figure 1: Correlation between signals

As seen in 2, some signals show clear cyclic behaviour.



(a) Cyclic behaviour of signal - L_T3



(b) Cyclic behaviour of signal - L_T5

Figure 2: Example of cyclic behaviour in various signals

1.3 Predictability of signals using *AR*: Auto-Regression

To test the predictability of our signals by modeling them with an Autoregressive model. AR models and processes are stochastic calculations in which future values are estimated based on a weighted sum of past values. AR models and processes operate under the premise that past values have an effect on current values, which makes the statistical technique popular for analyzing nature, economics, and other time-varying processes. Multiple regression models forecast a variable using a linear combination of predictors, whereas AR models use a combination of past values of the variable. Then by computing the residual of the predicted signal and the original signal, we checked the strength of the prediction. Some signals were more easily predictable then others. An example is shown in 3

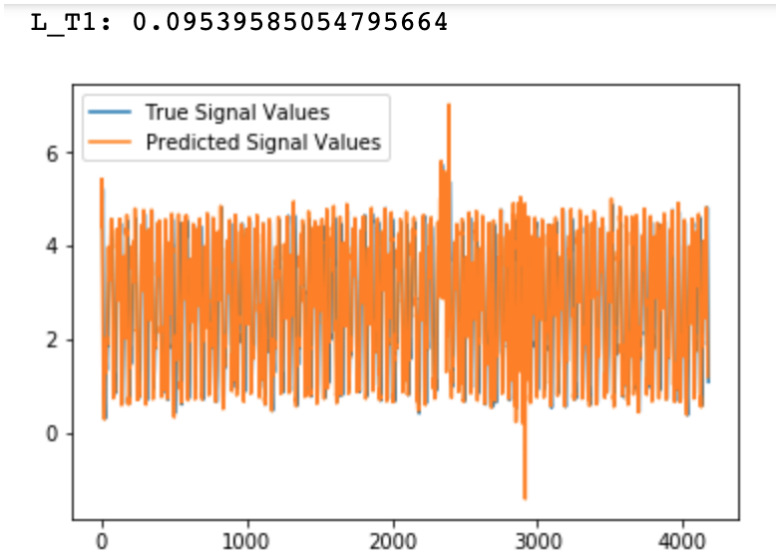


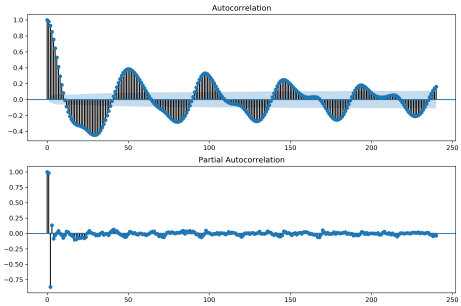
Figure 3: Prediction and actual model of L_T1, and the residual

2 *ARMA* - Auto Regressive Moving Average for Anomaly Detection

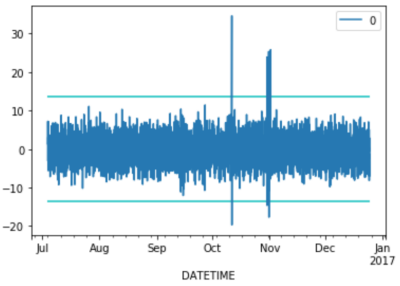
The ARMA model is used in the time-series modelling of sequential data in order to understand and, predict future values in the series. This task is achieved using an AR - Auto Regression part and an MA - Moving Average part. The AR part involves regressing the variable on its own lagged (i.e., past) values. The MA part involves modeling the error term as a linear combination of error terms occurring contemporaneously and at various times in the past. The model is usually referred to as the ARMA(p,q) model where p is the order of the AR part and q is the order of the MA part.
Categorical signals from actuators which have discrete values denoting open/close of tank were not considered for the series modelling task as ARMA models are meant for continuous sequential data and does not make sense when used to model categorical data. So, for this report all 25 continuous sensor signals were fit using the ARMA model.

Order of ARMA: For our report, In order to find the optimal p,q values of the sequential data signals, we do a grid search for each signal individually by fitting an ARMA model on the signal and evaluating the goodness of the fit in terms of the AIC - Akaike information criterion. The lower AIC value shows the better fit between the original signal and the modeled one. To perform the grid search in order to determine the bounds

of p,q values, we use the Auto-Correlation and Partial Correlation plot of one of the sensor signal data. This plot can be seen in Figure - 4a. From the image, with an arbitrary threshold of 0.5, we see for example in signals - L_T1, that the Auto-Correlation plot shows a higher correlation than the threshold for 6 lags and similarly in Partial-Correlation plot, it exceeds the threshold for 3 lags. Hence the grid search for p and q values are done in the range of 6 and 3 respectively.



(a) Auto-Correlation and Partial-Correlation - L_T1



(b) ARMA Threshold - P_J317

Figure 4: ARMA - Parameter Selection

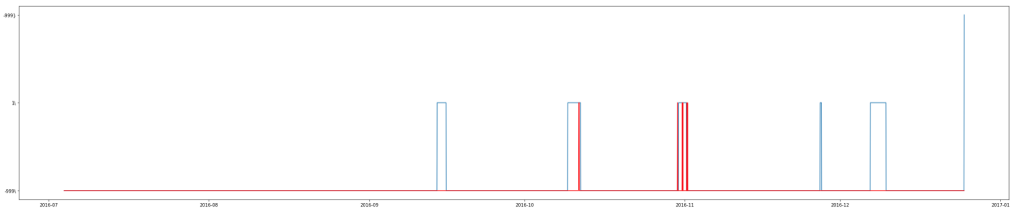


Figure 5: ARMA Predictions - P_J317

Threshold: Prediction threshold has been calculated based on residual of the ARMA model fit on the Training Dataset - 2. To be specific, the threshold for a signal is the 0.998^{th} quantile of the model's residual on the Training Dataset - 2. A sample threshold can be seen in the Figure - 4b. This task was performed on the training dataset

Anomaly Detection: Once the optimal p,q values for each signal and the residual threshold are selected using the Training Dataset - 1, each signal has its own p,q value and a threshold as well. Then anomaly detection is performed on the Training dataset - 2 using the threshold determined on each signal from Training Dataset - 1. The training dataset was partially labeled with unknown labels as -999. For the purpose of testing our ARMA model, for this report, we considered the unknown labels as normal and and only the features labeled as 1 were considered to be anomalies.

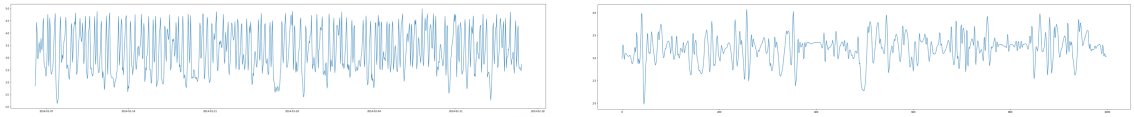
Study on detected Anomalies: The result of one of the best performing signal can be seen in Figure - 5. ARMA is only fit on continuous signal data and other discrete signal like actuator signals are not considered. The best performing signals with a precision greater than 0.5 are P_J280, P_J415, P_J302, P_J306, P_J307, P_J317. The ARMA model was not able to find any attacks in most of the signals, but there were few signals as mentioned above, which performed comparatively better but still was not able to detect more than 5 attacks out of 219 attacks. From the best performing models, we can see that ARMA tends to do better in the "P_J" - Pressure levels sensors.

3 *Discrete Markov Models* for Anomaly Detection

While the ARMA model assumes a continuous sequential data for prediction, there are other contrasting models which perform anomaly detection on discrete data, for example the *Markov Model* for anomaly detection. For this, first the continuous data should be converted to discrete sequences. Before the process of discretization, we denoise the sequential data, which is done using the *pywavelets* package [1]. The results of the denoising can be seen in Figure - 6.

Discretization: For this particular task we use the SAX - Symbolic Aggregate Approximation adopted from [2], [3]. In Piece-wise Aggregate Aggregation based SAX , first, time series is divided into smaller equal sized

windows. Data that falls into this window is represented by the window's mean value. Then similar mean values from this data is labeled with a letter from an alphabet. While SAX is one of the first discretization methods designed especially for time series data, the temporal aspect of the data is only taken into account by the preprocessing step of per- forming the PAA. The window size and the alphabet size create a tradeoff between efficiency and approximation accuracy [4]. Since SAX was developed especially for time series and comes with several parameters (eg: window size) to encourage flexibility and tradeoff SAX was chosen for the discretization task. The parameters - window size and the alphabet size were chosen on visual inspection of the Figure - 7. Post few trail and error simulations, the window size was chosen as 3 and alphabet size was chosen as 3 The results of the discretization can be seen in Figure - 7.



(a) Noisy Sequential Signal

(b) De-Noised Sequential Signal

Figure 6: De-Noising Sequential signal - L_T7

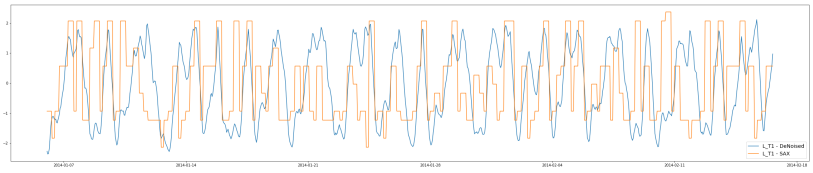


Figure 7: SAX Discretization - L_T1

Anomaly Detection: Post the discretization of signals, the discrete data is used to model a Markov Chain Model. This markov model is a model based temporal prediction model, where the model is based on the transition matrix built for each of the signals based on the frequency of transition between discrete values in the discretized sequential data. Once the transition matrix is formed for the signals, it can be used to predict for new time steps. To detect an anomaly here we have to determine a threshold which is determined based on the transition model, window size and the past values. This decision threshold is based on a minimum likelihood approach [5]. Post prediction of each signal, a majority vote combiner was used to combine predictions of signals only for which the precision was greater than 0.5. The value of 0.5 was determined after a tuning method which involved simulation of the above explained tasks for values ranging from 0.2 to 0.7.

Study on detected Anomalies:

The resulting prediction of the Discrete Markov Chain Model can be seen in Figure - 8. We see that the model is able to detect few attacks. Similar to ARMA, discrete model is also fit only on continuous signals. The combiner of prediction is applied on signals with a precision and recall greater than 0.7 and 0.3 respectively, similar to the above ARMA model. The best performing sensors were "L_T1" and "L_T6". We see that Discrete Markov Chain Models are perform well by detecting more attacks in the "L_T" - Water level signals sensors. The combiner of the best signals were able to detect 81 attacks out of 219 attacks in total, which is more than that of ARMA.

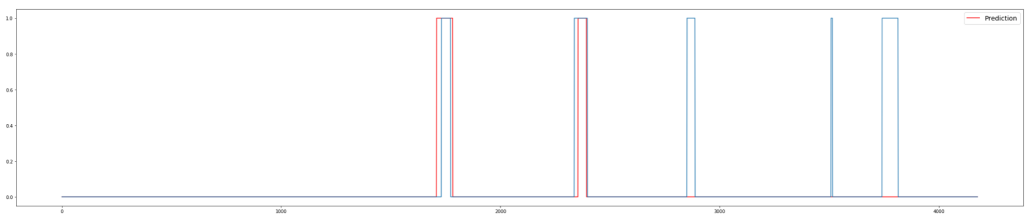
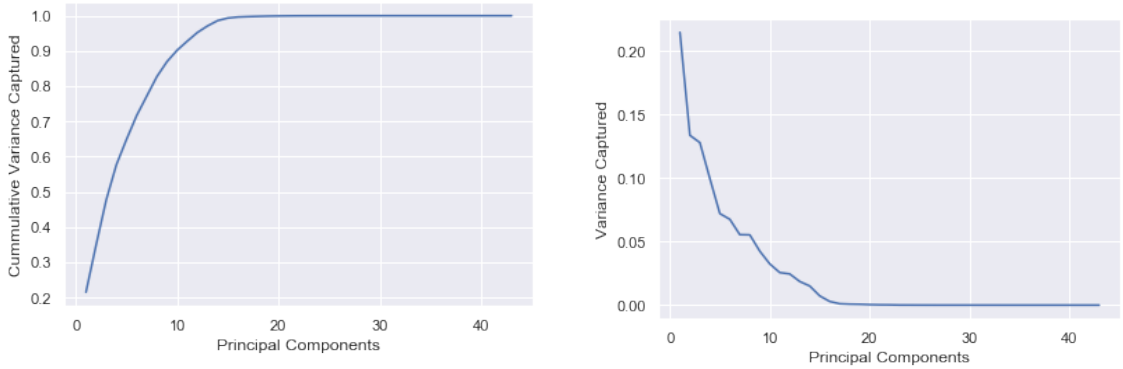


Figure 8: Discrete Markov Chain - Prediction

4 *PCA* - Principal Component Analysis for Anomaly Detection

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

PCA To normalize the data, we scaled it so that it would have zero mean and unit variance. We than applied PCA to it and selected the amount of components needed to keep 99% of the variance in the data (see fig 9a and fig 9b for results). The method used returned that the number of components to use to retain 99% variance is 14.



(a) Priciple Components vs. Cummulative Variance Captured

(b) Principal Components vs. Variance Captured

Figure 9: Principle Component Analysis

Anomaly detection By calculating the residuals as stated in section 5.1 of the paper [6], and finding the projection of the test data in the anomalous subspace, we create a threshold classifier with a threshold found through trial and error with cross-validation. We previously checked another threshold technique, but we were looking for low FP, so we have to adjust the threshold accordingly.

Study on detected anomalies: The resulting prediction of anomalies using PCA was actually quite strong. As seen in figure 10. The image shows the residual plot with the best found threshold, the predicted attack, and the actual attack from top to bottom. We were able to detect 87 of 219 attacks, with only 38 false positives. A mayor drawback however, is that after applying PCA, we can no longer tell which sensors or actuators are performing the best in the model. By using PCA, we were able to spot volume anomalies, and removing them from the training data would be best, since we only want to model the normal behaviour. However, in practice we found the models were performing worse if we removed them from the training data. This could either be because our anomaly detection is flawed, or because they actually do play a role in the prediction of attacks. More research on this is needed.

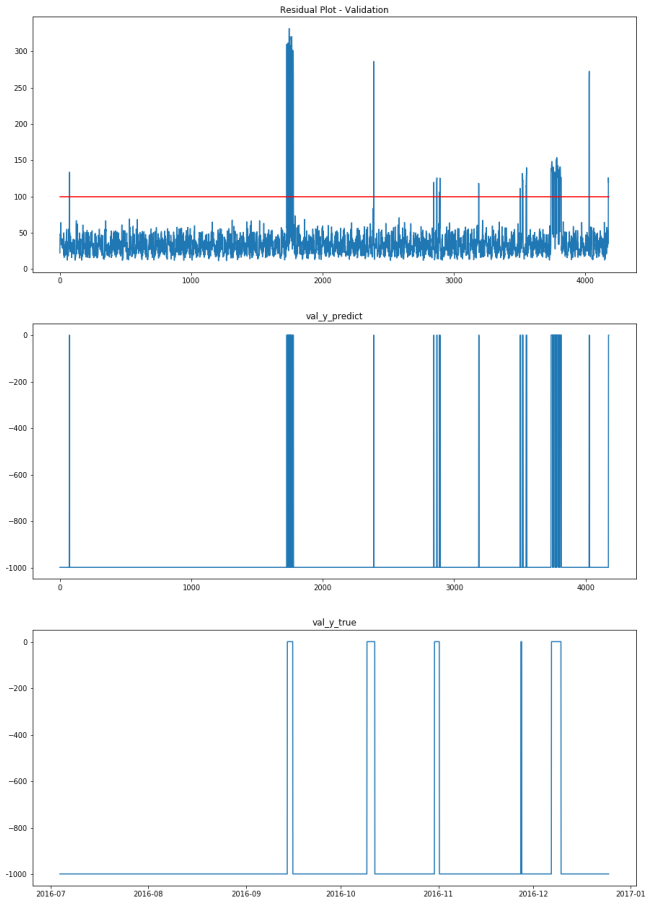


Figure 10: PCA predictions

5 *Comparison* of Anomaly Detection methods

	TP	TN	FP	FN	Recal	Precision	Accuracy
ARMA	3954	5	4	214	55.55	2.28	94.66
Discrete Markov	3926	81	32	138	36.98	71.68	95.93
PCA	3920	87	38	132	39.73	69.60	95.93

Table 1: Test point-wise Comparison - (TP, TN, FP, FN, Recall, Precision and Accuracy)

The Table - 1 presents the performance of the above explained modes - ARMA, Discrete Markov Model and PCA in terms of Confusion Matrix, Recal, Precision and Accuracy for a test point-wise comparison. The inferences we derive from the comparison table in terms of anomaly detection are,

1. The ARMA model performs the worst by detecting not more than 5 attacks. Though it produces only 4 false alarms (FP), it fails in the actual task of anomaly detection. The ARMA model also has several parameters like the p, q values which are different for each sensor signal and this requires a time consuming task of parameter tuning. These drawbacks make the ARMA model time consuming and inefficient for anomaly detection. One advantage of ARMA model is the simplicity and the ability of model to explain the best performing sensors.

2. The Discrete Markov Chain model performs better than the ARMA model by detecting 81 attacks with a precision of 71.68. It is also important to note that though the number of false alarms has increased to 32, the fraction of FP with respect to TP is still small. The discrete model similar to ARMA, explains the best performing sensor signals. The Discrete Markov Chain model requires several pre-processing of data (De-Noise, Discretization) which also comes with several tune-able parameters, this increases the complexity of the model and requires the analyst to choose these highly influential parameters optimally.

3. The PCA model performs the best by detecting 87 attacks, which is not a great improvement from the Discrete model. The PCA also does not explain the best performing signals as it samples the top sensors with the maximum variance. But the advantage of PCA can be seen from Figures - 8, 10 and Table - 2, here we see that PCA has detected at-least 1 anomaly in 5 windows of attack (It has detected at-least one attack in all the 5 actual windows of attacks labeled in the dataset) while the models ARMA and the discrete models were able to detect only 2 out of 5 regions.

	# of regions detected at least 1 anomaly in
ARMA	2
Discrete Markov	2
PCA	5

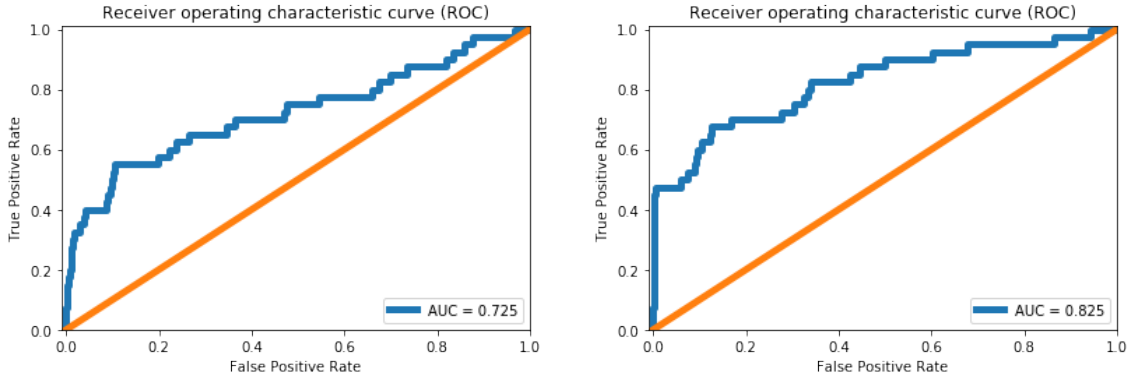
Table 2: Number of attacks detected (at least 1 anomaly) per detection method

Since our previous point-wise comparison concluded that the Discrete Markov model and PCA model differed very little, we decided to check how many attacks were detected by counting in how many attacks the model found at least one anomaly. We Found that PCA finds 5 attacks, while Discrete Markov only finds 2. However, the FP in the Discrete Markov model were around the attacks, while the FP of the PCA could be found in random places, far away from attacks. If we created a weight for the FP by distance of the closest attack, the PCA would perform worse. If we would however put a weight on the number of undetected attacks, the Discrete Markov model would perform worse. So depending on the need between detecting more attacks (PCA), or having no FP far from an attack (Discrete Markov) either of these two models would perform best.

6 BONUS: *Deep Neural Network* for Anomaly Detection

We tried multiple set ups with different values for the number of epochs, batch size, encoding dimensions, and learning rate. Then we would adjust the threshold to where the recall and precision would meet in the precision and recall figure. We struggled to get an AUC higher then 0.74-0.75. However, doing some research we found a publication [7], which states that by using a window size of 12 hours, their AutoEncoder for Event Detection (AEED) got up to an AUC of 0.95, which is a substanial increase of our previous three models.

From what we read in [7], the tanh activator seemed to work better then the original relu activator from the given autoencoder. Also, smaller match sized were used. This created a significant improvement in performance as seen in the ROC curves below.

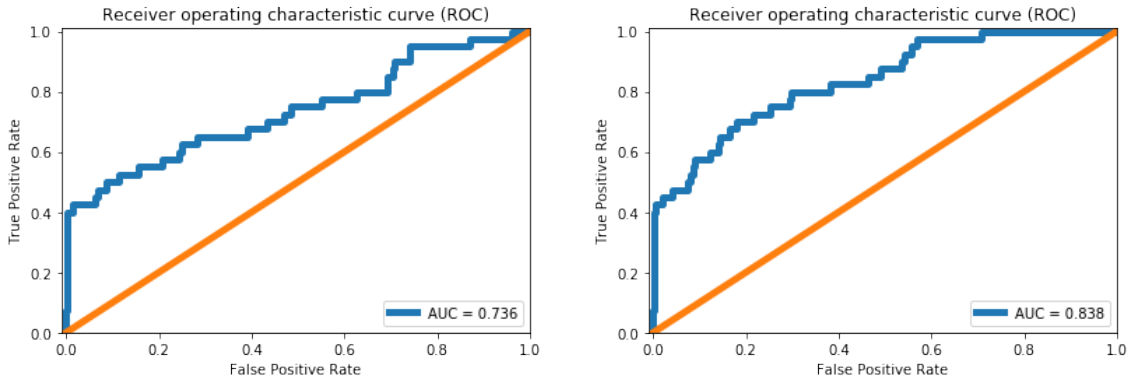


(a) ROC curve of relu activated autoencoder

(b) ROC curve of tanh activated autoencoder

Figure 11: ROC curves (test suit = 3)

We see a big increase in AUC from the tanh activated autoencoder. Also from the literature we decided to add additional hidden layer in both the tanh en relu activated autoencoders, to see if we got better performance. The results can be seen below.



(a) ROC curve of relu activated autoencoder

(b) ROC curve of tanh activated autoencoder

Figure 12: ROC curves (test suit = 3)

Again we can see in increase in the AUC metric for both activators, with the performance of the tanh activated autoencoder having a slightly bigger jump. Seeing as that we were able to get such descent performance in a short time, and the results from [7], where they managed to get an AUC of 0.953, using deep learning to predict an attack such as these on water infrastructure seem to be very viable.

7 Instructions to run the Code

The code for anomaly detection was built in Python-2.7 using Jupyter Notebooks. Follow the following steps to run the code,

1. Download the submission folder which contains the code.
2. Use Jupyter Notebooks, Python 2.7 to test the code.
3. For Familiarization Task - Run data_familiar.ipynb.
4. To test ARMA Task - Run arma_book.ipynb.
5. To test Discrete Task - Run discrete_book.ipynb.
6. To test PCA Task - Run pca_book.ipynb.
7. To test Neural Network Task, the bonus - Run autoenc_book.ipynb.

8 Bibliography

References

G. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and A. , "Pywavelets: A python package for wavelet analysis," *Journal of Open Source Software*, vol. 4, p. 1237, 04 2019.

Nathan Hoffman, "Python implementation of symbolic aggregate approximation," 2018. [Online]. Available: https://github.com/nphoff/saxpy

J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ser. DMKD '03. New York, NY, USA: ACM, 2003, pp. 2–11. [Online]. Available: http://doi.acm.org/10.1145/882082.882086

N. Ye, "A markov chain model of temporal behavior for anomaly detection," in *In Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, 2000, pp. 171–174.

I. Teodorescu, "Maximum likelihood estimation for markov chains," 05 2009.

A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in *ACM SIGCOMM computer communication review*, vol. 34, no. 4. ACM, 2004, pp. 219–230.

R. Taormina and S. Galelli, "Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems," *Journal of Water Resources Planning and Management*, vol. 144, no. 10, p. 04018065, 2018.