Cyber Data Analytics
Assignment 2

Ahmet Güdek (4307445) Takang Kajikaw Etta Tabe (4373758)

June 4, 2018

All code used in this assignment can be found on `https://github.com/agudek/CDA/tree/master/sequential_data`

Familiarisation

The data provided by BATADAL contains sensor data for several tanks, pumps, valves and pressure nodes with hourly measurements. The datasets contain data for are 7 tanks with water levels L_T1 to L_T7, 11 pumps with flow levels and status indicators F_PU1/S_PU1 to F_PU11/S_PU11, 1 valve with flow value F_V2 and status indicator S_V2, and 12 pressure node values P_Jxxx. The datasets also contain a ATT_FLAG column which can be used as label to distinguish between attack and no-attack scenarios.

The BATADAL website also shows the positioning of tanks, pumps and valves[1]. From this we can already stipulate that there is some correlation between sensor data, but to be certain we created a correlation matrix. This matrix in figure 1 shows high correlation between several sensor values. We see especially high correlation between state and flow values of pump sensors, which makes sense, as water only flows through a pump when it is turned on. To find some cyclic behaviour, we plotted all sensor data of the second data set in one large figure and added red colour to the points flagged as under attack. This figure 7 can be found in the appendix and shows that there is some cyclic behaviour, but it is difficult to distinguish whether the repetitions results in the exact same data points. The data points under the red areas also show that during an attack some sensors seem to show behaviour different to the original cyclic behaviour, which can be used to detect such attacks in the future.

Finally, we attempted time series prediction per sensor using a decision tree regressor after normalisation and sliding window preprocessing steps. We then calculated the mean absolute errors and mean squared errors to asses the predictions. These values can be found in table 1 in the appendix. From this we can see that most sensors can relatively easily be predicted to a reasonable degree as the largest mean absolute error is 0.25.

ARMA

In this section we detect anomalies using ARMA. In order to use ARMA to capture the underlying distribution of training data, it is important to determine its order. This order is expressed by parameters 'p' (the number of lag observations included in the model, also called the lag order.) and 'q' (the number of times that the raw observations are differenced, also called the degree of differencing), which can be obtained from the autocorrelation graph of each time series.

The autocorrelation graph for the signal L_T1, obtained by plotting the autocorrelation Function(correlogram or autocorrelation plot) is shown in Figure 2 where:
x-axis is the number of lags, y-axis is the autocorrelations and shaded blue region is the 95% confidence band.
In order to get a less noisy, easy to read plots which provide more insights, the number of lags on the X axis was reduced to 48 as shown in Figure 2.These were measurements from the 48 hours of 2014-April-06 and 2014-April-07(shown in Figure 2b). Figure 2b shows that the first 5 autocorrelations are out of the confidence band and show a strong positive correlation with each other, with the 5th autocorrelation being really close to the confidence band and with a not so large magnitude compared to the previous 4 (so we discard it). From this observation, we get a value of 4 for the 'q' parameter for signal L_T1. To get the 'p' parameter, a **partial autocorrelation function graph is plotted** as shown in Figure 2a. We observed that the first 2 autocorrelations are out of the confidence

[1] `https://batadal.net/images/fig1.png`

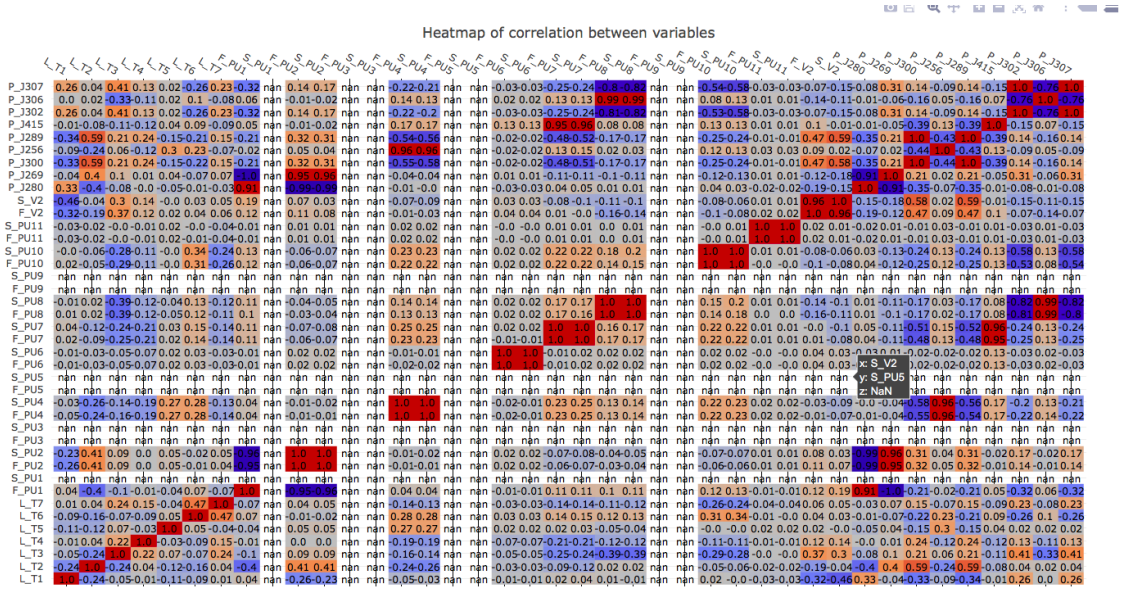
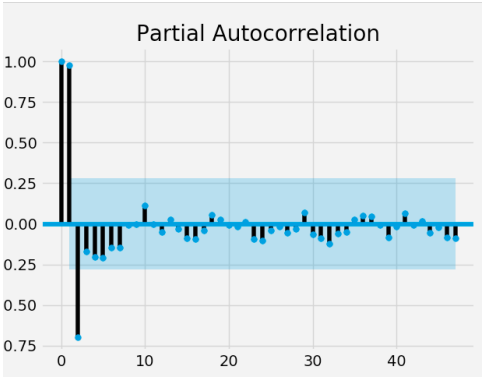
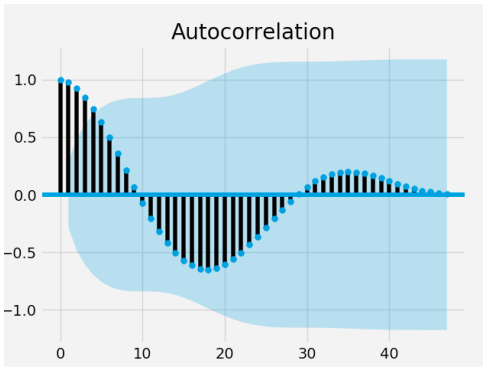


Figure 1: Correlation matrix for all sensors in the training set

bands and show positive correlation. From this observation, we get a value of 2 for the 'p' parameter for signal L_T1. These plots can be obtained when the class **plotting.py** on our github is run.



(a) partial autocorrelation of signal L_T1



(b) correlogram of signal L_T1

Figure 2: Autocorrelation plot with fewer Lags for signal L_T1

Akaike information criterion (AIC) is used to determine the parameters of the ARMA model. Different Arma models can be fitted with different parameters but the best fit parameters are those with the lowest AIC values. We wrote a method in python which calculates the best fit value for the p and q parameters. For signals L_T1 to L_T7 (which we chose to consider for the ARMA analysis section), the optimal p and q values can be found at [2].

[2] https://github.com/agudek/CDA/blob/master/sequential_data/aic_min_orders.csv

The upper and lower threshold selected for identifying anomalies are calculated using the formula:

$$\begin{aligned} Upper\ bound\ threshold &= mean + 3 * standard\ deviation \\ Lower\ bound\ threshold &= mean - 3 * standard\ deviation \end{aligned}$$

It is worth mentioning that the mean and standard deviations used are from the training dataset.

Figure 3 presents typical anomalies detected by ARMA models. Figure 3a shows the anomalies for signal L_T1. The red lines are the upper and lower threshold bounds, the orange plots are the predicted normal behavior and blue is the attack behavior. It shows that there was an attack detected in signal L_T1 around the 09- October and 11-October 2016 which shows the upwards spike above the upper threshold bound. Another attack was experienced around the 29-10 and 03-11 2016. These precisely match the attacks mentioned on the BATADAL website [3]. The Arma detected anomaly figures for the L_T2, L_T3, L_T4, L_T5, L_T6, L_T7 can be found on our github page [4]. From the figures, ARMA did a good job on detecting anomalies on most of the sensors except L_T4 and L_T7, where a large number of false positives were seen. This might be explained by the fact that these sensors show the most irregularities during normal operation as can be seen in figure 7, which can also be seen in the residuals of these sensors. These residuals, which can be found on GitHub, show much larger values than those of the other sensors. While the residuals of L_T1 in figure 3b are around between -0.5 and 0.5, L_T7 easily reaches the range of -2.0 to 1.5.

From this we see that ARMA is strong in predicting time series that do not show too much variation from its normal operation or do not depend on the change in time, i.e. not seasonal. The residuals can be used to score the predictions as residual values closer to 0 mean that the prediction is more accurate. This tells us that from figure 3b, the predictions of observation 0 are not accurate while the rest almost seem to be.

(a) Arma detected Anomaly for signal L_T1

(b) Residual for L_T1

Figure 3: Anomalies detected by ARMA model and Residual for L_T1

the prediction is more accurate.

Discrete models

We performed a simple percentile discretisation method on the sensor data where we labelled the data points with low(0), middle(1) and high(2) depending on their value after normalisation. We chose to make the splitting decisions at 0.33 and 0.67, as these values seemed to capture the least amount of noise after viewing the sensor data

[3] https://batadal.net/images/Attacks_TrainingDataset2.png

[4] https://github.com/agudek/CDA/tree/master/sequential_data/armaResults

visualisation from task 1. We then performed a sliding window method with length 6. This value was chosen after experimentation with the second training dataset and proved to create the best balance between true positives and false positives.

As a sequential data mining method we first attempted to use sequence alignment with kNN, but the distance calculation was extremely slow to perform even for a subset of a single column of the test data. So we eventually chose to implement n-grams. For this part, we chose not to use data of all sensors, but only those that show the most disturbance during attacks as can be seen in figure 7. We chose those to be F_PU1, F_PU6, F_PU7, F_PU11, P_J317 and P_J14. We calculated state transitions for all training sets and then combined the first and second training sets to decide border values, and combined the first f training set with the test set to test the method. An alarm was raised when a state transition occurred only once or twice. By comparing the the alarm timestamps with the list of attacks performed on the simulation provided by BATADAL[5][6], we saw that we were able to detect all but one attack in the second training set and all attacks in the testing set. However, we did obtain a large number of false positives as well, with 213 out of the 320 alarms raised fell outside the attack hours.

PCA

In this section, we performed anomaly based detection using PCA. For this task, we normalized the threshold value and the training dataset values between 0 and 1. To get the optimal threshold, we played around with the values starting at 0.01 and increasing it with 0.01 every time to get the optimal threshold. For this task, we got an optimal threshold of 0.09 which had an optimal number of true positives to false negative ratio.

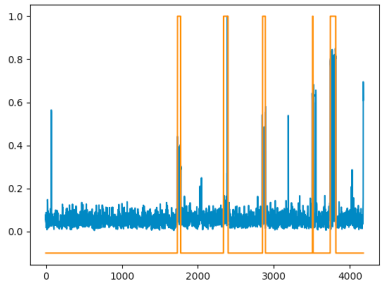


Figure 4: Time series for PCA residual on training dataset 2

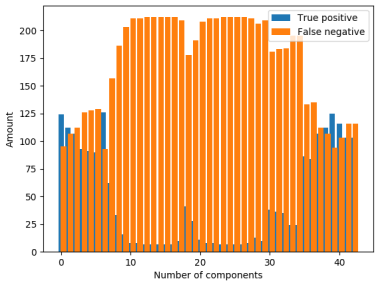


Figure 5: TP and False Negative plot for n components

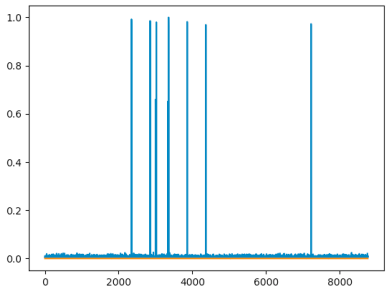


Figure 6: Time series for PCA residual on training dataset 1

Figures 4 and 6 show the time series for the PCA residual on the training dataset 2 and dataset 1 respectively. We considered the residuals per row, also known as the square prediction error (SPE). The orange line in the plots is the SPE per observation while the blue lines are the observations. In both figures, all observations with large SPE values are far off the model plane. This is due to the fact that that particular observation is inconsistent with the model. These observations are anomalies or attacks. In Figure 6, which is based on training dataset 1, we notice that the SPE per observations stays low for all observations (the orange line is horizontal throughout). This is because the dataset contains no attacks. In Figure 6, 5 observations are seen to have a relatively high SPE value, which indicate detected anomalies or attacks. From this, we conclude that some anomalies are detected in training dataset 2 but not a lot.

To complete the PCA task, the number of components had to be chosen to be considered by the model. In order to have the optimal n, we looped through all the 44 components and for each component, we checked the ratio of true positive and false negative and we chose the best n to be 7. This gives a total of 126 True positives and 93

[5] https://batadal.net/images/Attacks_TrainingDataset2.png
[6] https://batadal.net/images/Attacks_TestDataset.png

false negatives. All other n values deviated from these numbers. The graph for all n components can be seen in Figure 5.

PCA works well when detecting single point anomalies at a given point in time irrespective of its past. If the values of features are interdependent and the anomaly does not depend on a change in time, then it is more practical to use PCA.

Comparison

In this section, we evaluate and compare the results of the models presented. In order to do that, a couple of metrics are utilized. The first metric used is the test point-wise precision and recall scores which show the number of correctly identified and detected anomalies. It also shows how many points which are classified as anomalous are in fact anomalous. The table below shows the comparisons for the 3 methods.

	Precision	Recall	F1
ARMA	0.21	0.45	0.30
Discrete	0.28	0.22	0.24
PCA	0.25	0.58	0.35

From the above table, Low precision value indicates presence of many false positives, however high recall value ensures that most of the anomalous objects will be correctly detected.

Based on the above results, we see that PCA per

Appendix

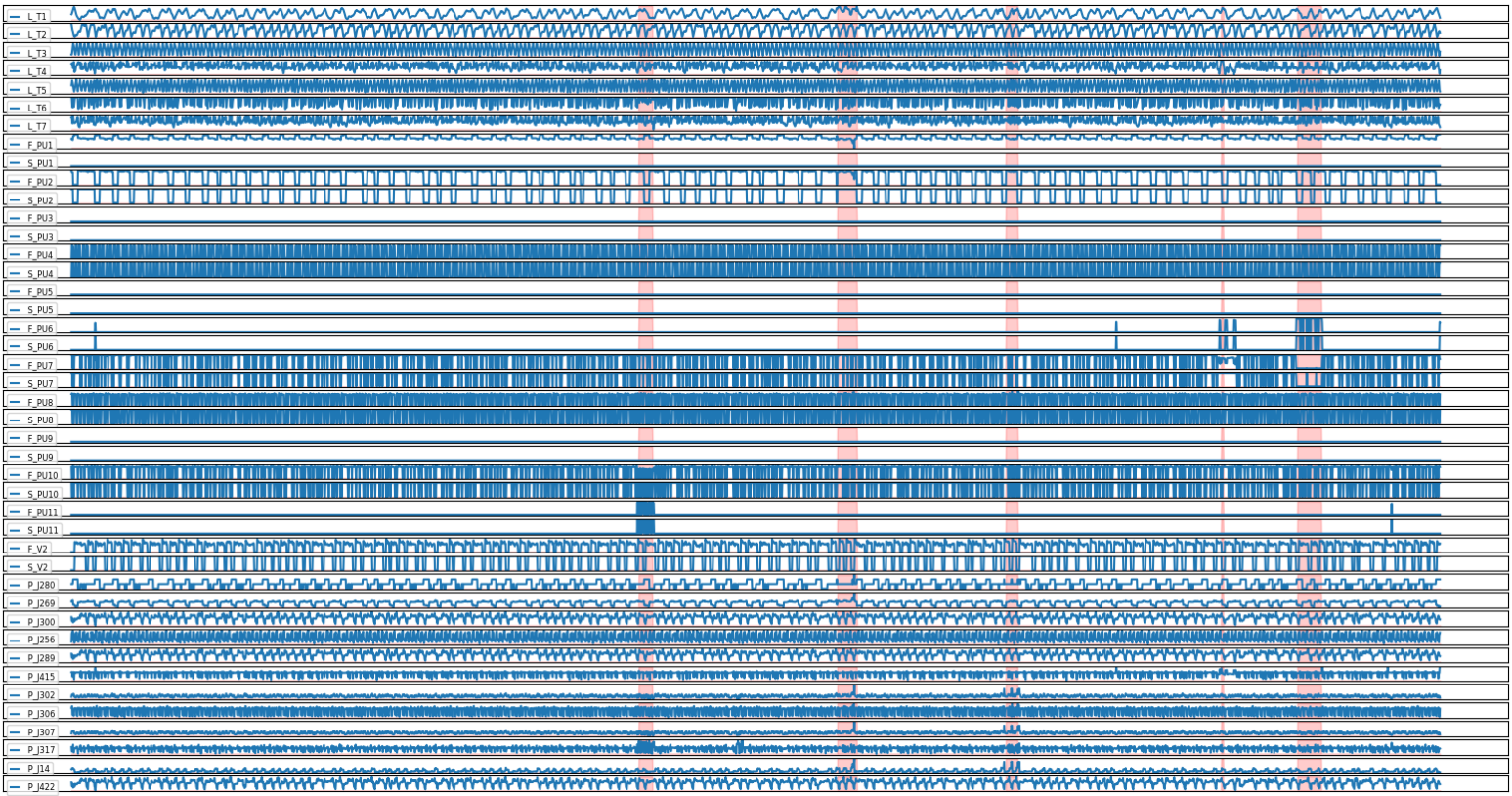


Figure 7: Sensor data of all sensors for around 2000 hours. There is some repeating behaviour and clear correlation between some sensors. The red areas indicate an attack.

SENSOR	MAE	MSE	SENSOR	MAE	MSE	SENSOR	MAE	MSE
L_T1	0.022223	0.000953	F_PU5	0	0	S_V2	0.034524	0.016044
L_T2	0.028325	0.001737	S_PU5	0	0	P_J280	0.024676	0.007195
L_T3	0.024081	0.001	F_PU6	0.00081	0.000001	P_J269	0.025246	0.001343
L_T4	0.100579	0.019963	S_PU6	0.000832	0.000001	P_J300	0.052389	0.005359
L_T5	0.045001	0.003001	F_PU7	0.193627	0.142074	P_J256	0.046919	0.009555
L_T6	0.078554	0.021339	S_PU7	0.246449	0.176561	P_J289	0.060291	0.008174
L_T7	0.08957	0.014528	F_PU8	0.111388	0.080821	P_J415	0.069046	0.019316
F_PU1	0.033895	0.008289	S_PU8	0.101953	0.070436	P_J302	0.094552	0.020629
S_PU1	0	0	F_PU9	0	0	P_J306	0.077051	0.033649
F_PU2	0.014013	0.003818	S_PU9	0	0	P_J307	0.104184	0.026168
S_PU2	0.046567	0.037897	F_PU10	0.11352	0.068766	P_J317	0.052581	0.008299
F_PU3	0	0	S_PU10	0.135863	0.070603	P_J14	0.060982	0.014944
S_PU3	0	0	F_PU11	0.000234	0	P_J422	0.05425	0.007199
F_PU4	0.070859	0.055151	S_PU11	0.000234	0			
S_PU4	0.04983	0.040001	F_V2	0.064459	0.019252			

Table 1: Performance of time series prediction using decision tree regression. (lower is better)