Delft University of Technology

Cyber Data Analytics
CS4035

Lab 2

Authors:
Josefin Ulfenborg (4989988)
Wouter Zirkzee (4398858)
June 4, 2019

Full code and data files are available at https://github.com/wouterz/cyber-data-analytics/

1 Familiarization

The signals that are present in the BATADAL data set [1] are water level for tank X (L_TX), switch status for pump X (S_PUX), flow of pump X (F_PUX), switch and flow status of valve X (S/F_VX) and pressure for one of the junctions (P_JXXX).

Every S_PUX together with F_PUX (for the same X), are correlated, if switch flag is 0 (pump is off) then the flow will also be 0. Furthermore, we also found correlation between pump 1 and pump 2, as seen in Figure 1 and Figure 2, where the top figure is for non-attacks, and the one to the bottom is for attacks. What can be observed is how these two pumps continuously alternate in the sense that when the flow of pump 1 starts to decrease, pump 2 gets switched on. This figure shows the relationship between them for the first 1000 rows of training data set 2, but as this appears to be a cyclic behaviour it will most likely follow the same trend for the full data set.

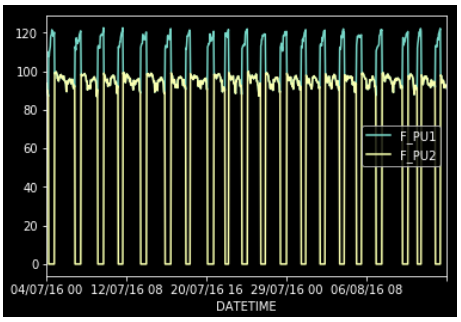

Figure 1: Flow of pump 1 and pump 2 for non-attacks.

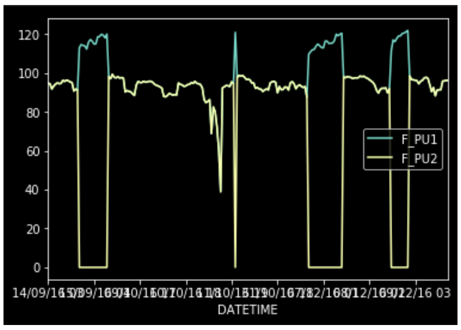

Figure 2: Flow of pump 1 and pump 2 for attacks.

Another correlation we observed was between the same two pumps, as well as water level for tank 1, as seen in Figure 3 and Figure 4. Again the top figure visualizes the data for non-attacks, and the bottom one for attacks. As with the previous figure, this plot shows the relationship between the signals for the first 1000 rows. We can now observe how the water level for tank 1 decreases as the flow of pump 2 gets switched off. Thus, it appears that the water level of the tank is more connected to pump 2, and not to 1.

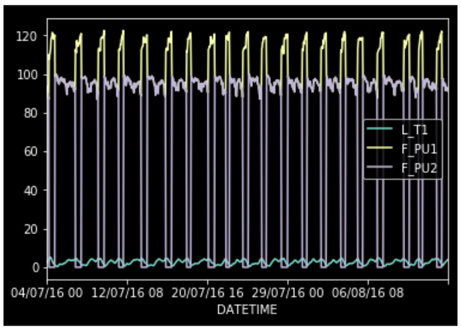

Figure 3: Flow of pump 1 and pump 2, together with water level for tank 1 for non-attacks.

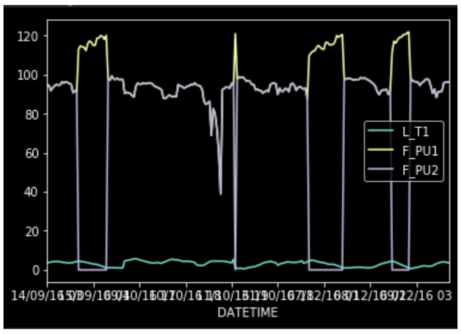

Figure 4: Flow of pump 1 and pump 2, together with water level for tank 1 for attacks.

Moreover, for the predicting phase we used AR, but since all three data sets had three different time stamps, we only used the second training data set for this phase. We split the data set into a training and testing part, as to only predict on the testing data. The performance of prediction can be seen in Figure 5.

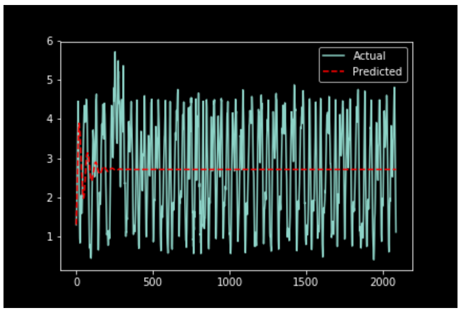

Figure 5: Predicting the water tank 1 level using AR.

This is our prediction by simply running AR without any further pre-processing. The results start very promising, however predicting too far ahead predictions seem to converge towards the mean. So for this scenario we believe predicting the next values in a time series on a short term basis is relatively easy, but gets harder as the time gap grows.

2 ARMA

The five signals we will learn ARMA models from are the level of the water tanks 1-5. We used the auto-correlation plots from the training data in order to decide the order of p and q For all tanks we chose $q = 1$. We chose $p = 10$, for tank 1, and $p = 6$, $p = 4$, $p = 4$, $p = 3$ for tank 2-4 respectively. The auto-correlation plot and the first residual graph for L_T1 can be seen in Figure 6, where the auto-correlation function reaches the upper confidence interval for the first time at approximately $x = 10$. The four remaining residual graphs can be seen in Figure 7 and Figure 8 for tanks 2-5 respectively.

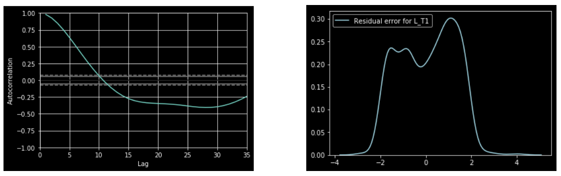

Figure 6: Auto-correlation plot for L_T1 and its residual graph.

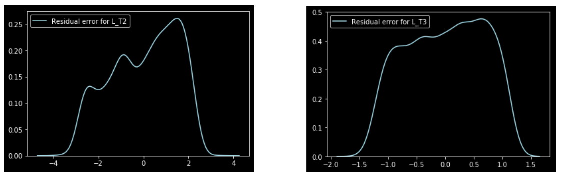

Figure 7: Residual graphs for tanks two and three.

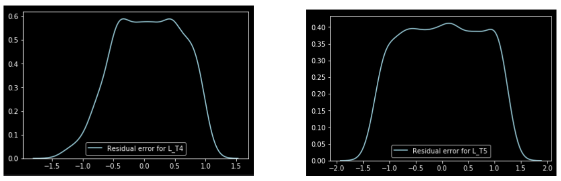

Figure 8: Residual graphs for tanks four and five.

Figure 9 visualizes how well ARMA detects anomalies for the level of water tank 1.

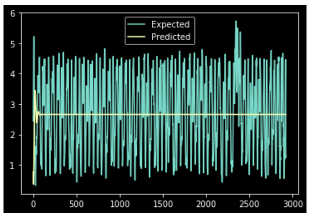

Figure 9: Anomaly detection using ARMA.

Unfortunately, when using ARMA to predict we end up with very poor results, as the prediction converges towards the mean. Around the x value 2400 there seems to be a clear anomaly, nevertheless it is not detected by ARMA.

3 Discrete models

For the discretization task we chose again to look at the level of water tank 1 (L_T1). We made the data points symbolic by binning them into five different categories: very low, low, medium, high and very high. For this task we used the *cut* function from *Pandas*, which set the threshold for the different categories automatically. This method was chosen because it seems relevant to be able to label the level of the water in the tank this way, and it is easy to classify new values that come in as well.

The discretization of the data can be seen in Figure 10, and in Figure 11 is a similar visualization of where the anomalies lie. From the top figure we can identify the peak in the middle as an anomaly on water tank 1, which can also be identified in the discretization image.

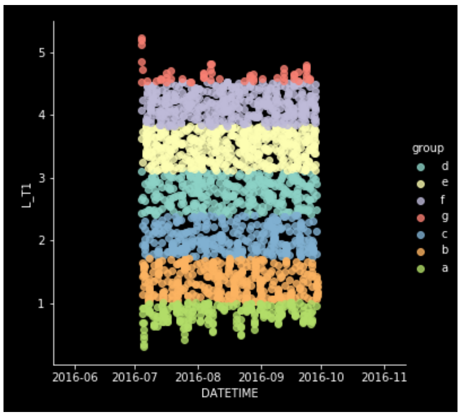

Figure 10: Visualization of discretization of the level of water tank 1 data training data.

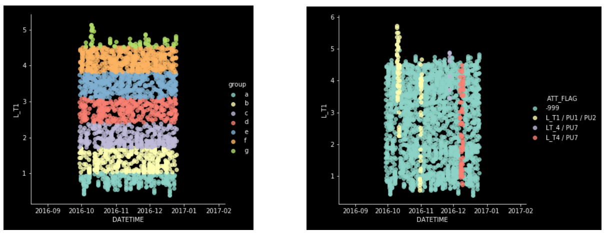

Figure 11: Visualization of discretization of the level of water tank 1 data test data and a plot of the attacks, yellow attacks influence tank 1.

4 PCA

By plotting the amount of variance explained against the number of components (Figure 12), it can be seen that around six components are able to explain most of the variance and is a good choice for the amount of PCA components to consider.

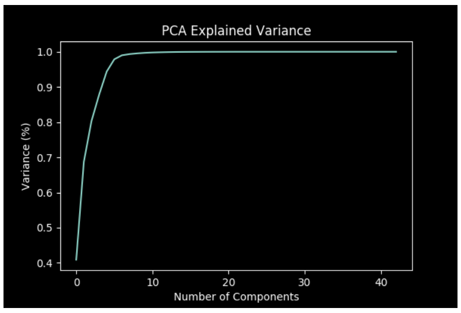

Figure 12: Explained variance against the number of components used by PCA.

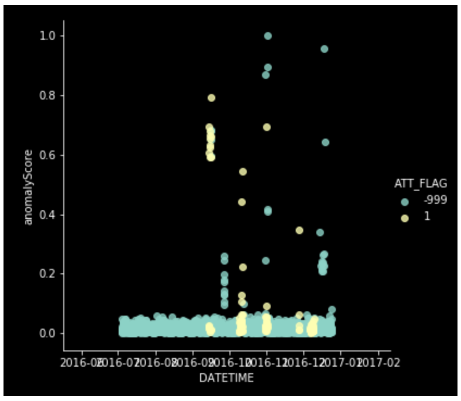

Figure 13: Anomaly scores obtained by reconstructing the data using 8 PCA components.

After regenerating the signal using the components we compute an anomaly score based on the distance between the original and the regenerated signal. Figure 13 shows the attack and non-attack datapoints and their respective anomaly score. A good choice for a decision boundary would thus be around an anomaly score of 0.4.

5 Comparison

In order to perform a comparison between the PCA, ARMA and discretized model we computed the precision and recall for for each of the models. We chose this method as it is good for the skewed data. The results in Figure 14 show that ARMA obtains the highest precision, but the discretized model obtained the highest. However as ARMA converges to the mean it can only detect outliers, the same holds for the discretization mode where PCA would also be able to detect more context related anomalies.

	ARMA	Discretized	PCA
Precision	0.6667	0.2098	0.4166
Recall	0.1443	0.1340	0.0515

Figure 14: Precision/recall comparison between the different models.

6 Bonus

To perform anomaly detection using a deep neural network we used PyTorch to create an implementation of a Long Short Term Memory (LSTM) neural network. This is a special kind of neural network, more capable of learning long term dependencies than a regular neural network making it well suited to be used with time series data. This model has been setup using two hidden layers, as this generally used for data which is not linearly separable. To select the amount of nodes in the hidden layer we experimented with different amounts to try and obtain a good fit on the timeseries without either under- or overfitting. In the end we used 200 neurons in the hidden layer, and decided on the amount of epochs to train by computing the loss per epoch. Indicated by Figure 15 around 35 epochs should provide enough training for the model. Through trial and error 36 epochs showed to perform best.

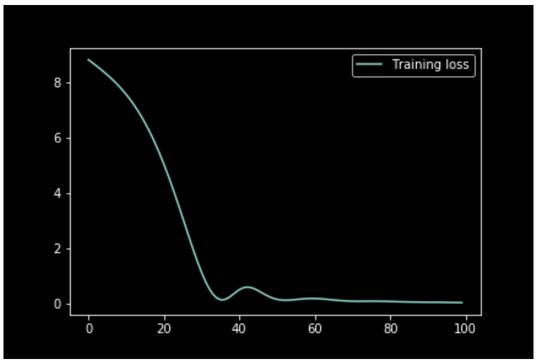

Figure 15: Training loss using 2 hidden layers with 200 neurons.

This amount of training also showed a decent fit on the normal data, Figure 16, and a slightly larger deviation during an attack Figure 17. Computing an anomaly score by distance squared from the predicted value the anomaly score, shown in Figure 18, the purple dots indicate an attack affecting tank 1. Since this is the data we used to predict, we only care about this attach and it seems to quite separable. The beginning and ending of the attack are still hard to detect as they are too similar to the regular data.

The performance, in terms of precision and recall, has shown to be good with the highest overall precision and comparable recall (Table 19). As deep learning networks can be tedious to setup and tweak to get the required results, we would not recommend it as we feel other models such as ARMA or PCA can be able to obtain similar and more understandable results in the end.

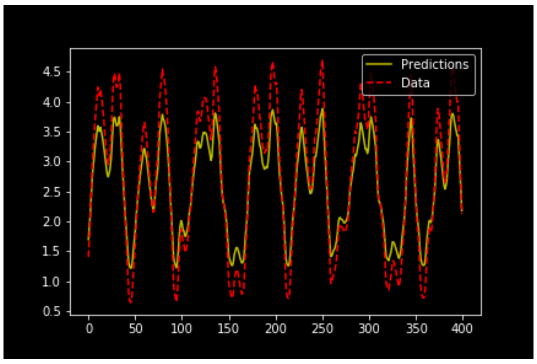

Figure 16: Deep learning fit on regular data of the level of tank 1.

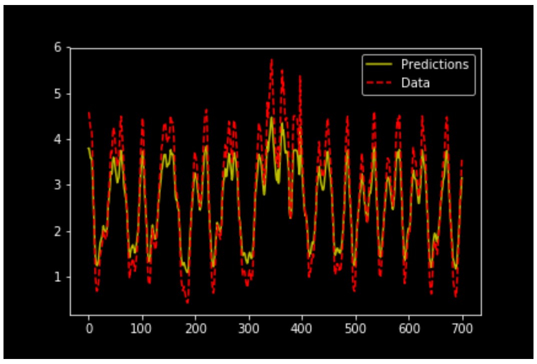

Figure 17: Deep learning fit on data containing an attack affecting the level of tank 1.

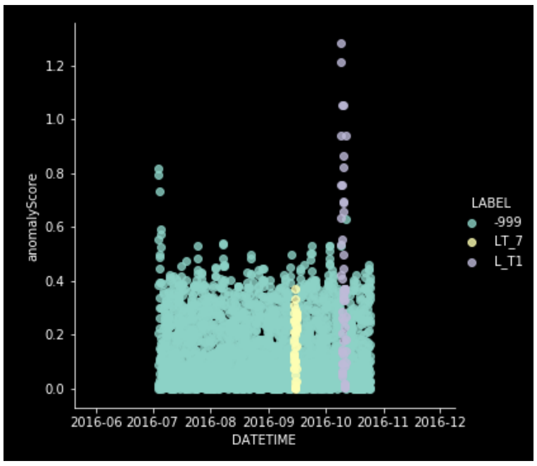

Figure 18: Anomaly score computed using the deep learning fitted prediction.

	PyTorch
Precision	0.7857
Recall	0.1134

Figure 19: Precision and recall obtained using PyTorch LSTM model.