Assignment 2

Group 7

June 3, 2018

Familiarization task

After loading and examining the BATADAL dataset, we can say that we have two different types of signals, the Sensors and the Actuators. More specifically, the sensor signals are: LT1-LT7 ,FPUB1-FPUB11, FV2, PJ280, PJ269, PJ300, PJ256, PJ289, PJ415, PJ302, PJ306, PJ307, PJ317, PJ14, PJ422. The actuator signals are (SPUB1-SPUB11 and SV2). The actuators receive only values 0 and 1 which indicate weather the sensors is open receiving data from the environment or not.

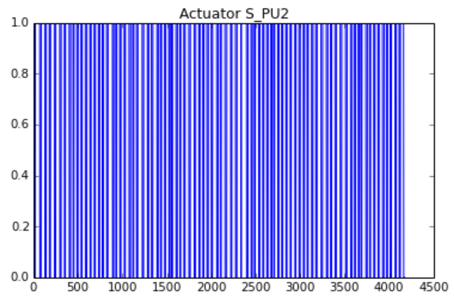

Figure 1: Visualization of example actuator data

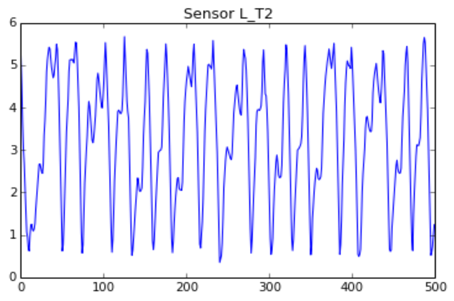

Figure 2: Visualization of example sensor data

In order to examine if the signals are correlated, we will plot a heatmap of all the signals.

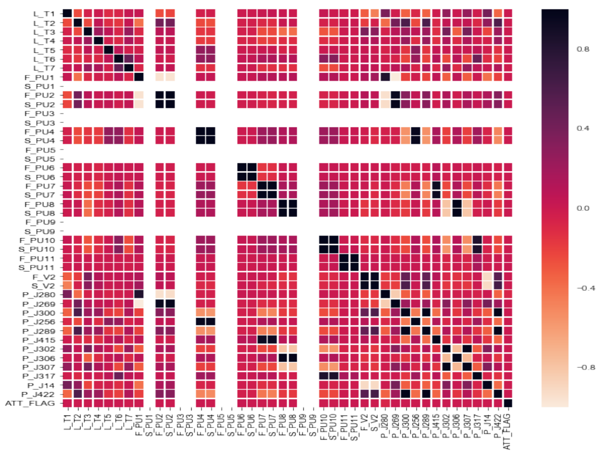

Figure 3: Heatmap to show correlation between the sensors

As we can see, some sensors are highly correlated. For instance we can see a high correlation between FPU1 and PJ269. Similarly we can see other correlations (all the black cells indicates high correlation while lighter cells show weak or non correlation between sensors and actuators).

We also want to investigate if the signals have some cyclic behavior. Even from the simple visualization of the signals, like in figure 2, we can see that there is some kind of cyclic behavior we will search it deeper by applying Fourier transform. In this way we will transfer the signal in the frequency domain. The Fourier transformations can be seen in the following figures:

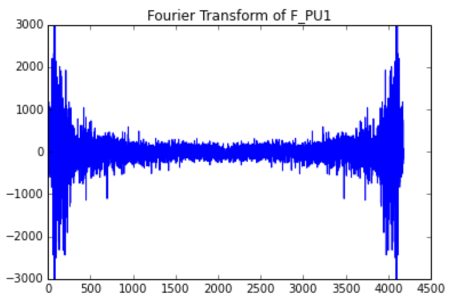

Figure 4: Fourier Transformation of F_PU1

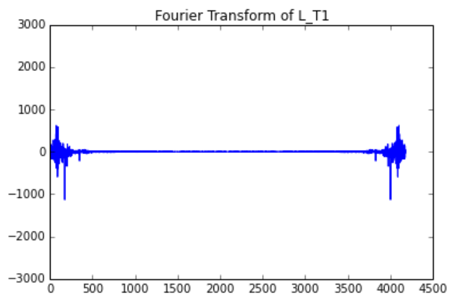

Figure 5: Fourier Transformation of L_T1

It becomes pretty obvious by these transformations that the signals have a cyclic behavior. We investigated more signals and the results can be found in the Jupyter notebook.

The final task of Familiarization is to check how difficult it is to predict the next value of a timeseries given the previous ones. There are many algorithms that can cope with this problem, however we used on one of the most simple but also really efficient. We run the Persistence model for one signal, the F_PU1. In the following figure we see how the test and train error change for the 100 first entries of the data. The error is not high, thus we can conclude that it is not always hard to predict the next value. Of course it also depends only the data and the spikes they contain. For smoother timeseries the results will be better, since this method depends on the previous value to predict the next one.

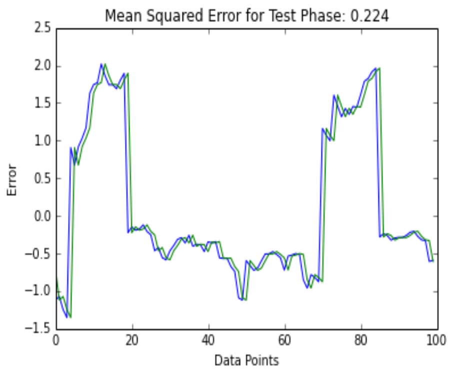

Figure 6: Mean Squared Error for Persistence Algorithm

ARMA task

In this task we used the ARMA model to detect anomalies in our data. Our initial step is to load the datasets (training dataset with only normal data, training dataset with normal and attack data and test data). Then, we created autocorrelation plots and residual errors for some sensors (we arbitrarily chose LT1, FPU6). As training set was used the normal dataset and as testing the second training dataset. The procedure of generating the ARMA models is as follows:

First we check if the signal of the sensor is stationary. In order to check it, we use the Augmented Dickey-Fuller test. This is necessary since in our code, we use the ARIMA model. Thus in order to determine the 'd' parameter value(0 or 1), we have to know if the signal is stationary. Then we used the Autocorrelation plots and the AIC statistics to find the optimal order (values p,q) for the ARMA model. Having the optimal order, we create an ARMA model with the training set and then, using the parameters of this model, we create the ARMA model for the test set. Finally having the residual of the predict model, we set a threshold in order to keep these samples that will be characterized as attacks. The threshold is based on the std of the training set residuals. Specifically, we set the threshold equal to 2*std and we keep that residuals from the predict model that were larger than this threshold. Thus these samples were the anomalies that were detected.

In the following we present the Autocorrelation and residual errors of the two sensors mentioned before.

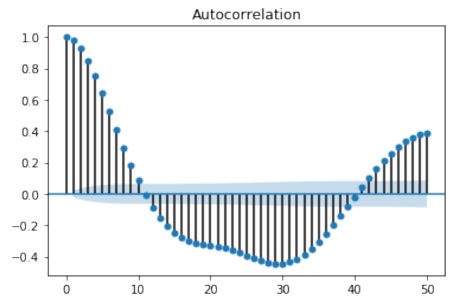

Figure 7: Autocorrelation for LT1 sensor

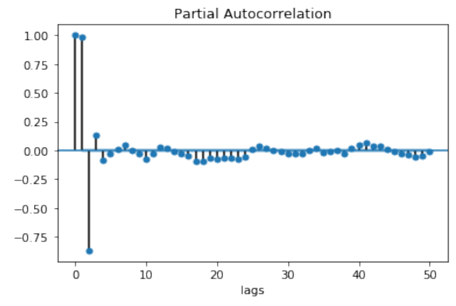

Figure 8: Partial Autocorrelation for LT1 sensor

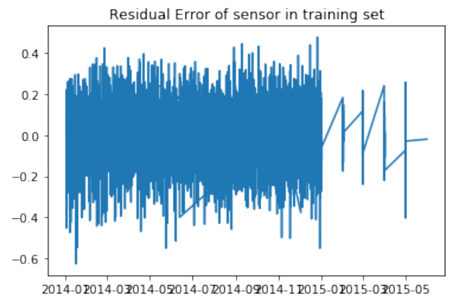

Figure 9: Residual errors for LT1 sensor in training set

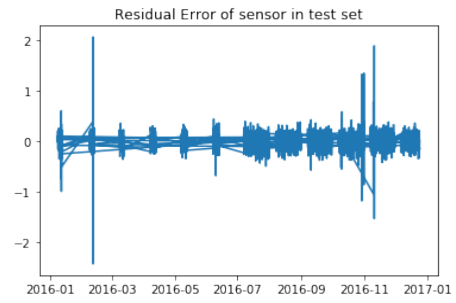

Figure 10: Residual errors for LT1 sensor in test set

Figures for more sensors and results can be found in the Jupyter notebook. The second part of that task is to create ARMA models for almost all sensors of the dataset. The results for the individual sensor can be found in the Jupyter Notebook, they were quite good. This part was executed two times. The first time we used as training and for tuning the parameters the normal data and for testing the second training dataset. For the second execution we used as training the normal data of course, for tuning the parameters the second train dataset, and for testing the test dataset.

Finally we combined all individual models in order to create a single anomaly detection method. The method that we used was to characterize as anomalies these samples that have been characterized as anomalies from at least two individual models (models for each sensor). The results for this part can be found in table 2.

As fas as the sensor that modeled effectively using ARMA concerns, we can say that the sensors with the better performance(accuracy,recall,precision) have been modeled more effectively. From the results in the notebook file, it is clear which are these sensors.

It is worthy to mention that using ARMA we can detect a specific type of anomalies more accurate. Specifically, we can accurately detect sudden anomalies that they do not last for long time. This is happen since ARMA is based on previous point to predict the next. Thus is the anomaly last for long time, the system will consider it, after a while, as a normal signal.

Discrete models task

In this task we create discrete models in order to detect anomalies in our time series. Thus we use the Symbolic Aggregate approXimation algorithm in order to transform our time series into strings. Specifically, the procedure for generating the discrete models for each sensor is:

Normalize the values of each sensor. Then we use the Piecewise Aggregate Approximation algorithm to split our data into fixed window size. Specifically we reduce our data into 116 pieces. After that steps we are ready to use the SAX algorithm to transform these pieces (data) into letter and finally create the final string by concatenating these letter. The last part is to generate tri-grams from this string. After doing that in both training and test set, we count the occurrences of tri-grams in both datasets and we distinguish the anomalies by detecting the tri-grams that appeared in test set (both attack and normal data) and did not appear in training set (normal). In the following image we present how the SAX algorithm works.

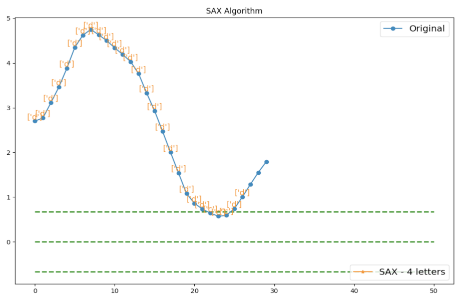

Figure 11: SAX algorithm

The results for the individual sensor can be found in the Jupyter Notebook, they were quite good. This part again was executed two times. The first time we used as training the normal data and for testing the second training dataset. For the second execution we used as training the normal data of course, and for testing the test dataset.

Same as in ARMA, the sensors with the better performance (as presented in the results in the notebook file) have been modeled more effectively. The anomalies that we can detect using discrete models are the opposite from the ones that ARMA can detect. Specifically, discrete models can detect accurate anomalies which last for long time.

PCA task

In order to conduct PCA we first do some preprocessing like, deleting the labels from the train set and making the signals have zero mean (normalization). Another important step is to remove the abnormalies from the normal data since may influence the principal components. In figure 13 we see these abnormalies. The next think we have to do is tune the parameters. There are many parameters that have to be tuned like the number of components that capture a big percentage of the variance and the threshold. Based on the research done in [1] we found that even though the network has 44 signals, the vast majority of variance in each timeseries can be well captured by 14 or 15 principal components as shown in figure 12.

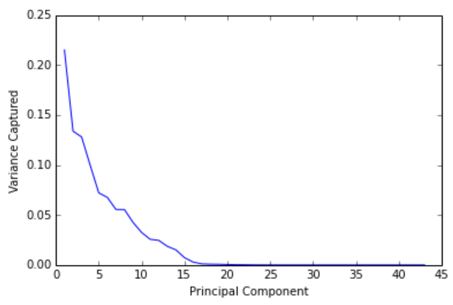

Figure 12: Fraction of total signal variance captured by each principal component

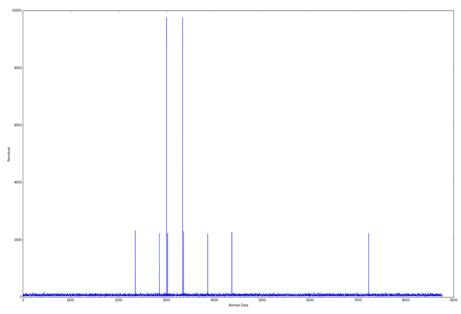

Figure 13: Residuals for train data and found threshold

We have also implemented a function that returns the number of components needed. Next we try to optimize the threshold using a statistical test for the residual vector (Q-statistic) [2]. However this method does not work properly for our cases and produces many false positives as can be seen in figure 13. Thus we manually set the threshold to a value that will improve our results. Lastly we tested the PCA model using the evaluation dataset, which contained also attacks. Using the graphical representations we investigated how different thresholds perform. In table 1 the metrics can be found.

PCA model is able to detect point-wise anomalies but it has the drawback that it is not able to find the corresponding sensor.

Comparison task

Firstly we will evaluate the systems based on some accuracy, precision, recall and Fscore. The table 1, 2 and 3 provide our results for the second train set (BATADAL_dataset04.csv). We have to make clear that for the comparison of the models we have combined all the individual models created by ARMA to create a global anomaly detection method, the same procedure has also been followed for SAX.

Metric	value
TP	88
FP	38
Accuracy	95.95%
Precision	69.84%
Recall	40.18%
F_score	51.01%

Table 1: Metrics for PCA

Metric	value
TP	9
FP	23
Accuracy	94.42%
Precision	28.12%
Recall	4.11%
F_score	7.17%

Table 2: Metrics for ARMA

Metric	value
TP	112
FP	320
Accuracy	89.77%
Precision	25.93%
Recall	51.14%
F_score	34.41%

Table 3: Metrics for SAX

From the above tables we notice that the best combination of accuracy and precision is obtained by PCA. Also the Fscore of PCA is the highest, even though the recall is 10% less than the recall of SAX. Recall can be considered of the most important metrics in our case since we don't want to miss any dangerous attacks in the water system. What was unexpected was the low value of recall and fscore of the ARMA model. Just 4% of the items that should have been marked as attacks were actually marked. We investigate further how the models perform using the actual test set. ================================**TEEEEeeeeeeeEEEST**==============

In general we have made these observations about the models we used. For ARMA model, it analyzes multiple signals and then combines their predictions. The method is definetely applicable to real life problems and we achieved pretty good results for both datasets. However there are some drawbacks we would like to mention: You need to select stationary signals and build models for every signal, which is time-consuming. Also, in the training phase, it takes a lot of time to model all those signals.

As far as the PCA model is concerned, it reconstructs data into principle component space and calculate the residual between original data and reconstructed data. Due to the fact that top few principal components capture variability in normal data while outliers have variability in the smallest component, we can use the residual and set a threshold to detect the anomaly. It have several advantages and disadvantages: Some of the advantages of the model is that it can be operated in an unsupervised mode. Also, it is suitable for high dimensional data since it is a dimensionality reduction methods. Last but not least it is simpler and faster than ARMA. A major disadvantage is that PCA is based on the assumption that anomalies and normal instances are distinguishable in the reduced space.

The discrete model we chose was SAX. SAX may be more complicated in its implementation, however it was extremely faster than ARMA, even if you have to analyze each signal separately. Also SAX may distinguish anomalies that are not that distinguishable form the normal data (the assumption that makes PCA weak).

As a conclusion we recommend using PCA. We are actually more familiar with this technique and we managed to tune the parameters in a way that improved our results a lot. As we mentioned above PCA provided the highest Fscore, Accuracy and Precision, however the recall was not that high. As a alternative we would recommend SAX. This technique seems really promising and with further investigation would definitely yield to better results.

Another method we chose was overlap-based false and true positives method. This metric will show us how similar the models are between them. That will also be helpful for the bonus task, where we want to combine the models to improve our performance. Models that are not very similar may detect different anomalies and it would be interesting to investigate how their combination would perform.

TP	FP	Accuracy	Precision	Recall	Fscore
203	589	85.52%	25.63%	92.69%	40.16%

Table 4: Results for the 1st Approach of Combining Models

TP	FP	Accuracy	Precision	Recall	Fscore
86	17	96.41%	83.50%	39.27%	53.42%

Table 5: Results for the 2nd Approach of Combining Models

To evaluate the models using this method we consider one models labels as the true ones and another's labels as the predicted ones. We have 3 pair-wise comparisons, since we have 3 models.

True Labels	Predicted Labels	TP	FP	Accuracy	Precision	Recall	Fscore
ARMA	PCA	12	114	96.79%	9.52%	37.5%	15.19%
SAX	ARMA	7	425	89.23%	1.62%	21.88%	3.02%
PCA	SAX	66	60	89.80%	52.38%	15.28%	23.66%

From the results above we notice that the highest similarity between two models is between PCA and ARMA. In more details these two models gave the same labels to 4043 objects and different to just 134. Some kind of similarity do also indicate all the other models, as can be seen by the high accuracy obtained. We notice that the best precision is obtained by the overlap between PCA and SAX, these two models result to much less FP than the other pairs. ARMA and SAX seem to be the most different models, since we see that just 7 entries where identified as attacks by both the systems.

Bonus task

There is much research done on how to combine individual model, for anomaly detection, in order to improve the results. Some of the papers we studied were [3], [4], [5] and we tried to implement two different approaches.

The *first approach* was based on the importance of isolating anomalies from normal traffic variation. Thus this approach first uses PCA to locate these anomalies in the normal data and remove them. Then we decided to use SAX since, as we have mentioned in the Comparison section, seems to be different from PCA and also resulted in really high recall which is very important for anomaly detection, to avoid attacks. The code is commented in Jupyter notebook, but the results can be seen also here:

It is obvious that the most strong point of this approach is the recall. 92% of the attacks were identified by our model. However we see that the system is pretty conservative and marks many more entries as attack than the real ones.

The *second approach* is based on the majority voting. We receive the predictions of each model separately and then we conduct majority voting to decide if each entry is actually anomalous or not. Thus if at least two of the models have marked an entry as anomalous the final result will be anomalous otherwise it will be normal. The performance we obtained was pretty impressive, as can be seen in table 5. In this case we have a system that is not that conservative, meaning that it does not focus on predicting each and every one of the anomalies, but has really high accuracy, so it does not label many normal data as anomalous.

The second approach combines all three models and actually get the best components of each of them. We get the high accuracy of PCA and ARMA, with high precision and also influenced positively by the high recall of SAX.

We provide two solutions, the first one is able to identify almost every anomaly in the data, yielding to many false alarms (safe choice) and one that may not detect everything, but what it detects is correct.

Code

The codes for this assignment can be found in https://github.com/Stavroulakis/CyberDataAnalytics_CS4035/tree/master/Assignment_2

References

[1] Anukool Lakhina, Mark Crovella, and Christophe Diot. 2004. Diagnosing network-wide traffic anomalies. SIGCOMM Comput. Commun. Rev. 34, 4 (August 2004), 219-230. DOI: https://doi.org/10.1145/1030194.1015492

[2] Edward Jackson, J & Mudholkar, Govind. (1979). Control Procedures for Residuals Associated With Principal Component Analysis. Technometrics. 21. 341-349. 10.1080/00401706.1979.10489779.

[3] Augustin Soule, Kavé Salamatian, and Nina Taft. 2005. Combining filtering and statistical methods for anomaly detection. In Proceedings of the 5th ACM SIGCOMM conference on Internet measurement (IMC '05). USENIX Association, Berkeley, CA, USA, 31-31.

[4] Zare Moayedi, H & Masnadi-Shirazi, M.A.. (2008). ARIMA model for network traffic prediction and anomaly detection. Proceedings of the International Symposium on Information Technology. 4. 1 - 6. 10.1109/ITSIM.2008.4631947.

[5] H. Zare Moayedi and M. A. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," 2008 International Symposium on Information Technology, Kuala Lumpur, Malaysia, 2008, pp. 1-6. doi: 10.1109/ITSIM.2008.4631947