Real time anomaly detection in network traffic time series

Sergio Martínez Tagliafico[1], Gastón García González[1], Alicia Fernández[1], Gabriel Gómez Sena[1], and José Acuña[1,2]

[1] Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Uruguay
{sematag,gastong,alicia,ggomez,acuna}@fing.edu.uy
[2] Telefónica Móviles, Uruguay
jose.acuna@telefonica.com

Abstract. Anomaly detection is a relevant field of study for many applications and contexts. In this paper we focus in on-line anomaly detection on unidimensional time series provided by different network operator equipments. We have implemented two detection methods, we have optimized them for on-line processing and we have adapted them for integration into a testbed of a well known Hadoop big data platform. We have analyzed the behavior of both methods for the particular datasets available but we also have applied the methods to a publicly available labeled datasets obtaining good results.

Keywords: Anomaly Detection · Kalman Filter · Hadoop.

1 Introduction

Detecting anomalies in the behavior of multiple variables gathered from the network infrastructure is an essential task to be able to detect failures and also to react as soon as possible to solve the issues. Although common monitoring systems can be able to report failures in hardware equipment or software services, they do not normally provide alarms when the quality of one service is being degraded. Moreover, an efficient anomaly analysis can be useful to detect performance issues, attacks to network security and fraud attempts.

Although anomalies analysis in telecommunications traffic is a mature area [2][12][5][3][9] with approaches based on statistical methods [11][8], the emergence of big data platforms which enable the processing of massive and diverse data, poses new opportunities and challenges. Particularly the development of analytics for platforms that solve the detection of anomalies of large volumes of data simultaneously is a important issue [1][4][13] and determines the need of adapting the algorithms implementations for parallel processing.

All the variables considered in this work have some kind of periodic behavior so our methods will provide a prediction based on the statistical of the past samples of the variable. Also a decision process is needed to signal which samples should be considered as anomalies. As stated, the ability to detect anomalies in

real time is a value added feature because it enables a fast reaction to reduce the service unavailability or degradation time for the customer.

The main contribution of this work is the development of an anomaly detection strategy based on stochastic modeling and the implementation on a Hadoop[3] big data platform testbed. We have also implemented a classification strategy based on Parzen Windows and we have compared both methods. For the validation process we used real data provided by a network operator and we also tested our method with publicly available labeled datasets.

In this document, we define in Section 2 the relevant types of anomalies for the specific application field and the proposed methods. In section 3 we depict some implementation details. In Section 4 we show some selected validation scenarios and finally in Section 5 we conclude and identify some possible future works.

2 Strategy for anomaly detection

2.1 Type of anomalies

We can define an "anomaly" as a set of values of a variable that are far from its normal or expected values. Therefore, we need to define a region of the feature space of the data that represents its normal behavior. Any data out of the normal region, will be consider as an anomaly.

The definition of the normal feature space can be a complex task. In some cases the normal behavior feature space may vary along the time and also it is sometimes difficult to have labeled traffic to aid the normal region definition.

Based on [5], anomalies can be classified as:

- Point Anomalies: A single value can be considered as anomalous with respect to the rest of the data.
- Contextual Anomalies: A single value is anomalous in the context of its neighbors values but in other cases can be considered normal.
- Collective Anomalies: A collection of related data values is anomalous with respect to the entire data set.

The data used for this work is non labeled unidimensional time series obtained from telecommunication infrastructure, for instance, the interface traffic from a router. The relevant anomaly types for this scenario are "Point Anomalies" and "Collective Anomalies". When we find an abnormal change of the series value respect to the expected value for that time instant, we will be in presence of a point anomaly. Besides, we can find that some values have a slight but prolonged withdrawal in time so as to be considered a collective anomaly.

For this particular context it is also relevant the ability to perform anomaly detection in real time. The scenario is that a monitoring system will produce a stream of sample values for the chosen variable. The cadence of the variable samples will depend on the monitoring system configuration, typically in the

[3] http://hadoop.apache.org/

order of minutes. The detection process will receive a streaming of values and it should produce an indication if an anomaly is detected. It is obvious that the detection process cannot take longer than the sample interval.

2.2 Point and Collective anomalies: ARIMA+Kalman models

There are a lot of techniques mainly used for anomaly detection in this scenarios [5] and our first approach is to adjust a stochastic model for the time series and define an anomaly based on whether an observation is suspicious of not being generated by this model.

Let be y_k a time series and $Y_k = (y_1, \ldots, y_k)$ the vector of observations which represents its evolution up to time k, the detection process used is based on obtaining the distribution of the series in time $k+1$ given its evolution Y_k. We will write this distribution as:

$$p(y_{k+1}|Y_k) = p(y_{k+1}|y_k, \ldots, y_1)$$

In this approach we use ARIMA (Autoregressive Integrated Moving Average) models represented as a state space model [7].

$$y_k = \boldsymbol{Z}\boldsymbol{x}_k + \varepsilon_k, \qquad \{\varepsilon_k\}\ iid \sim N(0, \sigma_\varepsilon^2) \tag{1}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{T}\boldsymbol{x}_k + \boldsymbol{R}\boldsymbol{\eta}_k, \qquad \{\boldsymbol{\eta}_k\}\ iid \sim N(\boldsymbol{0}, \boldsymbol{Q}) \tag{2}$$

where $\boldsymbol{Z}$, $\boldsymbol{T}$ and $\boldsymbol{R}$ are fixed matrices. Matrices are represented by bold letters in our notation.

As every distributions in this model are Gaussian, the distributions $p(y_{k+1}|Y_k)$, $p(\boldsymbol{x}_k|Y_k)$ y $p(\boldsymbol{x}_{k+1}|Y_k)$ are also Gaussian. Lets call $\hat{\boldsymbol{x}}_{k|k} = \mathsf{E}[\boldsymbol{x}_k|Y_k]$, $\hat{\boldsymbol{x}}_{k+1|k} = \mathsf{E}[\boldsymbol{x}_{k+1}|Y_k]$, $\boldsymbol{P}_{k|k} = \mathsf{Var}[\boldsymbol{x}_k|Y_k]$ y $\boldsymbol{P}_{k+1|k} = \mathsf{Var}[\boldsymbol{x}_{k+1}|Y_k]$, then

$$p(\boldsymbol{x}_k|Y_k) = N(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}) \tag{3}$$

$$p(\boldsymbol{x}_{k+1}|Y_k) = N(\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{P}_{k+1|k}) \tag{4}$$

$$p(y_{k+1}|Y_k) = N(\boldsymbol{Z}\hat{\boldsymbol{x}}_{k+1|k}, \boldsymbol{Z}\boldsymbol{P}_{k+1|k}\boldsymbol{Z}' + \sigma_\varepsilon^2) \tag{5}$$

where $\hat{\boldsymbol{x}}_{k|k}$, $\hat{\boldsymbol{x}}_{k+1|k}$, $\boldsymbol{Z}\hat{\boldsymbol{x}}_{k+1|k}$ are minimum variance linear unbiased estimators (MVLUE) of $\boldsymbol{x}_k$, $\boldsymbol{x}_{k+1}$ and y_{k+1} respectively given Y_k.

The on-line estimation of this distribution can be done by the well known Kalman Filter equations.

$$e_k = y_k - \boldsymbol{Z}\hat{\boldsymbol{x}}_{k|k-1}, \qquad F_k = \boldsymbol{Z}\boldsymbol{P}_{k|k-1}\boldsymbol{Z}' + \sigma_\varepsilon^2$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_k + \boldsymbol{P}_{k|k-1}\boldsymbol{Z}'F_k^{-1}e_k, \qquad \boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{P}_{k|k-1}\boldsymbol{Z}'F_k^{-1}\boldsymbol{Z}\boldsymbol{P}_{k|k-1}$$

$$\boldsymbol{x}_{k+1|k} = \boldsymbol{T}\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k e_k, \qquad \boldsymbol{P}_{k+1|k} = \boldsymbol{T}\boldsymbol{P}_{k|k-1}(\boldsymbol{T} - \boldsymbol{K}_k\boldsymbol{Z})' + \boldsymbol{R}\boldsymbol{Q}_k\boldsymbol{R}'$$

Using this result, we can manage to efficiently update the distributions and the state as soon as each sample arrives to the system. As can be seen, only matrix sum and product are involved. This result will be very important for on-line anomaly detection.

Anomalies definitions The first stage in our anomaly detection process is to identify an abrupt change of the data values from the expected value for a given time instant. This condition can be defined as:

Definition *(Type A anomalies - Point Anomalies)*: Let be y_k a Gaussian process modeling a time series, $Y_n = (y_1, \ldots, y_n)$ the set of realizations representing the evolving up to the time instant n and $p(y_{n+1}|Y_n) = N(m_{n+1}, P_{n+1})$ the conditional distribution of y_{n+1} given its evolution up to time instant $k = n$. A point type anomaly occurs at $k = n + 1$ if

$$\|y_{n+1} - m_{n+1}\| > r * \sqrt{P_{n+1}}$$

that means at $k = n + 1$ the value is more than r standard deviations far from its expected value.

The second stage is intended to detect an slight but sustained in time shift from the expected series value.

Definition *(Type B anomalies - Collective Anomalies)*: Let be y_k a Gaussian process modeling a time series, $\{y_{n-(l-1)}, \ldots, y_n\}$ the last l observations of the series and $p(y_{k+1}|Y_k) = N(m_{k+1}, P_{k+1})$ the conditional distribution of y_{k+1} given its evolution up to the time instant k. A collective anomaly occurs at $k = n$ if

$$y_k - m_k < \sqrt{P_k} \quad \forall\ k = n - (l-1) : n$$

or

$$y_k - m_k > \sqrt{P_k} \quad \forall\ k = n - (l-1) : n$$

2.3 Point anomalies: Parzen windows

The analyzed data has an intrinsic weekly periodicity, so we have chosen the Parzen windows method[4], to measure how near is a sample from its nearest neighbors.

For that purpose, we began constructing a circular buffer with the accumulated samples of the last week indexed by its collected time-stamp. When a new test sample arrives, a cluster of an hour history of samples centered in its time-stamp is created.

For each value in the cluster, we considered a gauss function with mean value the sample and a presetted variance value. The new test sample is then evaluated with each of the cluster functions and the accumulated value is compared with a presetted threshold value. If the sum is above the threshold, it means that in the environment of the current sample there are several samples of the cluster. Otherwise it means that the current sample is far from the values of the cluster and it is labeled as an out-lier. The outliers are discarded for updating the circular history buffer.

[4] Inspired on the density estimation by Parzen windows.[6]

Method The circular buffer can be represented as $W = (x_t^n, ..., x_{t+m}^n)$, where the index t indicates the temporal position in the buffer, and the index n indicates the week where the sample belongs. At the beginning all the samples of the buffer are from the same week.

When the samples begin to arrive in real time, the current sample $\boldsymbol{x_k^n}$ is taken and a cluster w_k is generated from the samples of the circular buffer taken half hour back and half hour forward referring to the time of the current sample.

$$w_k = (x_{k-h/2}^n, ...x_{k-1}^n, x_k^{n-1}, x_{k+1}^{n-1}, ..., x_{k+h/2}^{n-1})$$

Then, for each cluster sample, a window function is used, where the value of the current sample is evaluated. In this case we will use a gauss function where the mean will be the value of the sample and the variance σ will be a presetted parameter.

$$p_k = (N(x_{k-h/2}^n, \sigma), ...N(x_{k-1}^n, \sigma), N(x_k^{n-1}, \sigma), N(x_{k+1}^{n-1}, \sigma), ..., N(x_{k+h/2}^{n-1}, \sigma))$$

All the values obtained from the evaluated functions are added and the result is compared with a threshold value U. If the sum is above the threshold it means that in the environment of the current sample there are several samples of the cluster nearby. Otherwise it means that the current sample is very far from the values of the cluster, then it is labeled as out-lier. The outliers are discarded when updating the circular buffer.

$$\sum_{i=k-h/2}^{k+h/2} p_{ki}(\boldsymbol{x_k^n}) \ < \ U. \tag{6}$$

3 Implementation highlights

The implementation was done in Python and has been integrated into a Hortonworks HDP[5] platform testbed. The on-line data ingestion was done through Apache NiFi and Apache Kafka and then the data processing was done running the python code with pySpark. The implemented software was thought with a modular architectural design in mind so as to enable an easy change of the detection algorithms.

For the modeling phase, we use the module `statsmodels.tsa.statespace`[6] and particularly the class `statsmodels.tsa.statespace.sarimax.SARIMAX`, which allow modeling by ARIMA space state models.

The anomaly detection algorithm is implemented in the module `anomaliasKF`[7]. The two main components of the anomaly detection process for real time detection are implemented in the class `AnomalyDetector.py` helped with the `pykalman`[8] module for the Kalman Filter equations.

[5] https://hortonworks.com/products/data-platforms/hdp/
[6] (http://www.statsmodels.org/dev/statespace.html)
[7] https://iie.fing.edu.uy/ sematag/anomalias/
[8] https://pykalman.github.io/

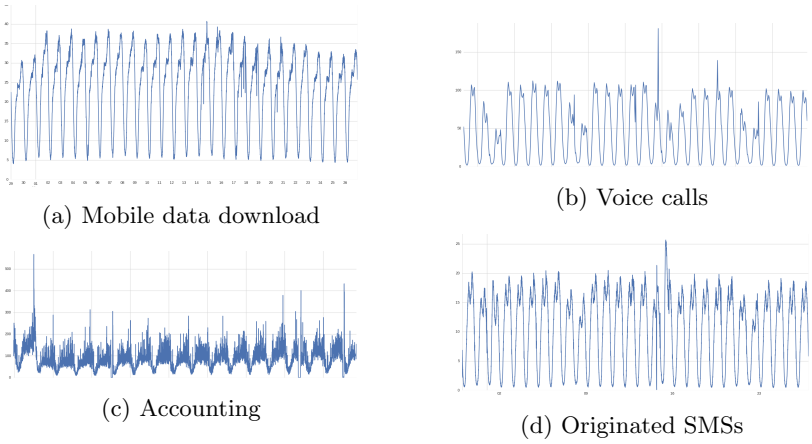
4 Experiments and results

4.1 Dataset characterization

The available time series for evaluating the proposed methods come from our partner, a mobile operator in Uruguay. Table 1 show the main characteristics of the series and figure 1 illustrate the general behavior of each one. For each series, we use the data of the first days to adjust the stochastic model and the rest of the data was used to test the method.

Series	Name	Sample frequency	Duration	Train set
1	Mobile_Data_Downlink_Bytes	1 sample each 5 minutes	28 days	First 7 days
2	Voice_Calls_Originated	1 sample each 1 hour	28 days	First 7 days
3	Accounting_Mobile_Data	1 sample each 5 minutes	18 days	First 4 days
4	SMS_Originated	1 sample each 5 minutes	28 days	First 7 days

Table 1: Time series used for validation



(a) Mobile data download

(b) Voice calls

(c) Accounting

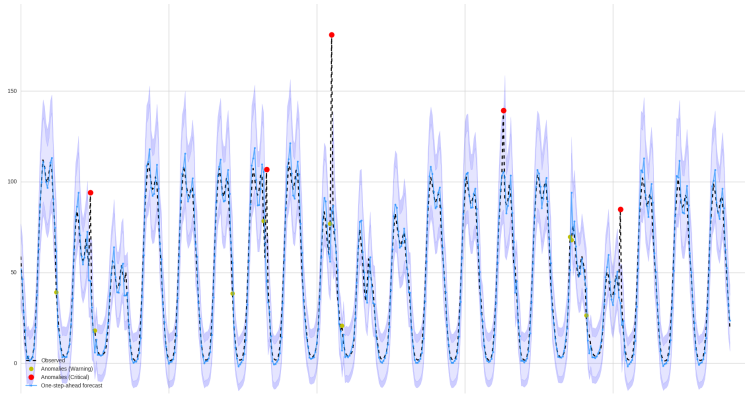
(d) Originated SMSs

Fig. 1: Global view of the time series analyzed

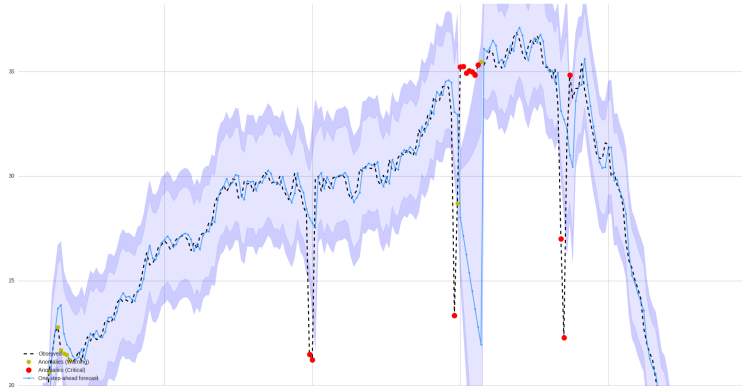
Among the referred real datasets and to improve the evaluation of the proposed strategy in other types of time series, we have tested our approach in two time series from the Numenta Anomaly Benchmark (NAB)[10]: the hourly demand for New York City taxis and the real time traffic data (occupancy) from the Twin Cities Metro area in Minnesota.

4.2 ARIMA+Kalman results

The proposed approach was applied to detect anomalies for the four time series referred in Table 1. Figure 2 shows some anomalies (in red) detected on the series. In all cases anomalies were detected in zones where the series have an obvious abnormal behavior. Moreover, no relevant false alarms were generated, only some cases were generated immediately after a true anomaly was detected as shown in figure 2b.



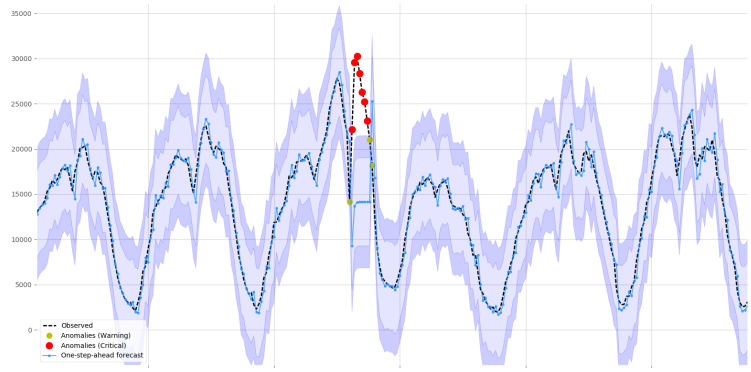
(a) Voice



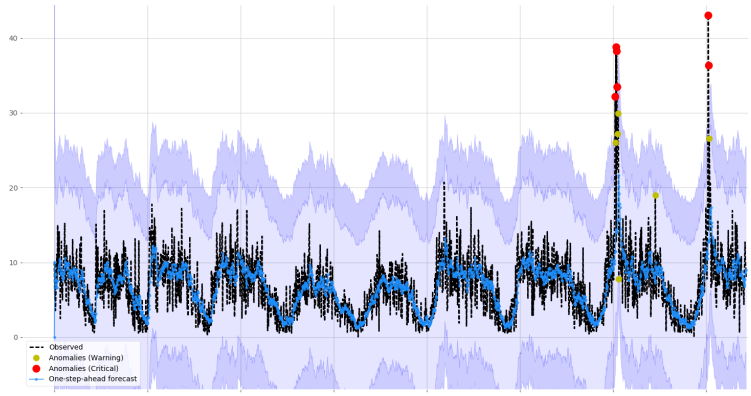
(b) Mobile data download

Fig. 2: Anomaly detection results for Mobile data download and Voice data

We have obtained similar results for the NAB time series, as can be seen in figure 3. In all cases, low false alarms were generated and labeled anomalies were completely detected. Also good performance is achieved for others NAB time series.



(a) NYX taxi demand



(b) Minnesota Metro occupancy

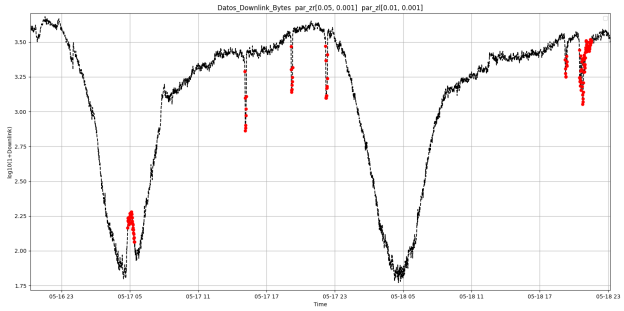
Fig. 3: Anomaly detection results for NYC taxi demand and Minnesota Metro occupancy

4.3 Parzen Windows results

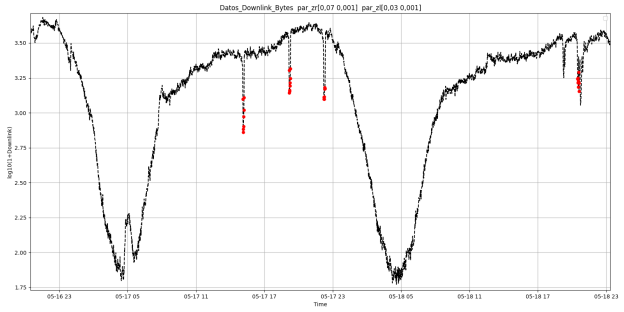
The implementation of the Parzen Windows method was also done in Python and in this section we are presenting the results for the mobile data download series. As explained in section 2.3 the method has two parameters, the window width σ and the decision threshold U. In this type of series during the night and part of the morning the difference in values between consecutive samples is larger than during the rest of the day, due to the rapid fall of the activity at the end of the day and the rapid reactivation of the activity by the morning. Then it is convenient to use different parameters for these two scenarios. When the difference in values between consecutive samples is large, it is better to use a larger window width (σ). For this test we have used this parameters:

- $\sigma_{night} = 0.05$, $U = 0.001$. For the night and the early morning.

– $\sigma_{day} = 0.01$, $U = 0.001$. For the rest of the day.



(a) Parzen method applied to mobile data download. Parameters: $\sigma_{night} = 0.05$, $\sigma_{day} = 0.01$ and $U = 0.001$.



(b) Parzen method applied to mobile data download with a wider window. Parameters: $\sigma_{night} = 0.07$, $\sigma_{day} = 0.03$ and $U = 0.001$.

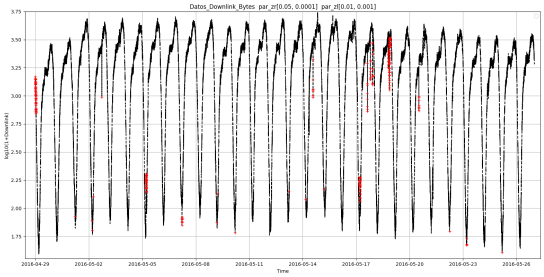
Fig. 4: Anomaly detection results for mobile data download

As shown in Figure 4a the outliers (in red), correspond to the detected point type anomalies. The parameters were adjusted until an acceptable result was obtained. In Figure 4b the effect of a wider window is shown ($\sigma_{night} = 0.07$, and $\sigma_{day} = 0.03$).

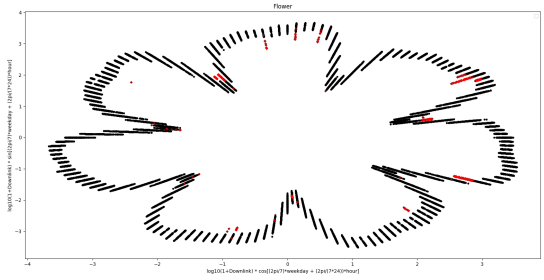
As can be seen, the amount of red samples decreased because the model is more tolerant when the window width increases.

Another way to visualize the data is the one shown in Figure 5b. This representation is quite helpful to find anomalies at a glance and to depict the weekly evolution of the series.

Figure 5a shows a series corresponding to one month of data collected and classified. In figure 5b you can see the same series represented as the flower. An



(a) Series corresponding to one month of data.



(b) The same series represented in a flower of a week.

Fig. 5: Flower representation

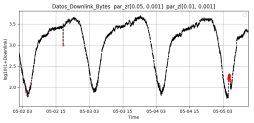
entire turn of the flower represents a week's time, in figure 5b are the four weeks corresponding to the month of figure 5a. In figure 6 shows an example for three day of the week.

5 Conclusions and future work

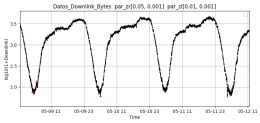
We have implemented an algorithm for real time anomaly detection using ARIMA models and Kalman filtering, obtaining good performance results for our operator partner time series. Both point an contextual anomalies can be detected. The approach was also tested with publicly available labeled time series showing good results. The use of Kalman filtering enable us the use of the algorithm for real time anomalies detection.

We have also implemented a Parzen Windows oriented method for point anomalies which is simple and requires very few calculation resources. We have also obtained interesting detection results.

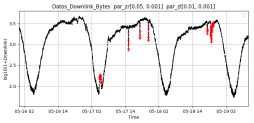
Both methods were implemented and integrated as a module into a hadoop platform sandbox, enabling the later integration to the operator production systems.



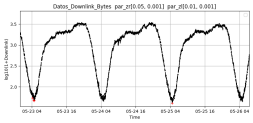
(a) First Mon, Tue and Wed of the month



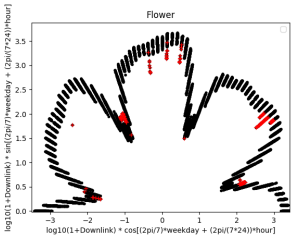
(b) Second Mon, Tue and Wed



(c) Third Mon, Tue and Wed



(d) Fourth Mon, Tue and Wed



(e) Flower representation of the four weeks data

Fig. 6: An example of how the data is represented with the flower

Based on the experience and results of this work, we have found some relevant point to work on.

First of all, the anomaly definition is a relevant issue that condition the detection process. For now, we have worked with a Gaussian model of the series but an hypothesis test over the data distribution characteristics can be faced.

Regarding the ARIMA+Kalman Filter implementation we used ARIMA state space models but other state space model families can be explored, for instance structural models or dynamic factor models[9]. Moreover, we want to work on some kind of automatic learning for the model in the training phase. We have also shown that after an anomaly has been detected the subsequent prediction is not good, so we want to improve the detection for this anomaly stage, perhaps introducing robust Kalman filtering.

Regarding Parzen Windows method, we need to improve the parameters adjusting for the different stages of the series values. We want to introduce some kind of automatic adjusting depending on some statistical properties instead of setting them manually.

[9] http://www.statsmodels.org/dev/statespace.html)

Acknowledgements

This work was partially supported by Telefónica Móviles (Uruguay) and the Groups Program of the Comisión Sectorial de Investigación Científica, Universidad de la República (Uruguay). The authors are thankful to both institutions.

The authors would like to thank Pedro Casas for its good advice at the beginning of the project.

References

1. Bär, A., Finamore, A., Casas, P., Golab, L., Mellia, M.: Large-scale network traffic monitoring with dbstream, a system for rolling big data analysis. In: Big Data (Big Data), 2014 IEEE International Conference on. pp. 165–170. IEEE (2014)
2. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials **16**(1), 303–336 (FebJan 2014). https://doi.org/10.1109/surv.2013.052213.00046, http://dx.doi.org/10.1109/surv.2013.052213.00046
3. Brutlag, J.D.: Aberrant behavior detection in time series for network monitoring. In: LISA. vol. 14, pp. 139–146 (2000)
4. Casas, P., Soro, F., Vanerio, J., Settanni, G., D'Alconzo, A.: Network security and anomaly detection with big-dama, a big data analytics framework (2017)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (Jul 2009). https://doi.org/10.1145/1541880.1541882, http://doi.acm.org/10.1145/1541880.1541882
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience, second edition edn. (2000)
7. Durbin, J., Koopman, S.J.: Time series analysis by state space methods, vol. 38. Oxford University Press (2012)
8. Knorn, F., Leith, D.J.: Adaptive kalman filtering for anomaly detection in software appliances. In: INFOCOM Workshops 2008, IEEE. pp. 1–6. IEEE (2008)
9. Lakhina, A., Crovella, M., Diot, C.: Diagnosing network-wide traffic anomalies. In: ACM SIGCOMM Computer Communication Review. vol. 34, pp. 219–230. ACM (2004)
10. Lavin, A., Ahmad, S.: Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In: Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on. pp. 38–44. IEEE (2015)
11. Mazel, J., Casas, P., Labit, Y., Owezarski, P.: Sub-space clustering, inter-clustering results association & anomaly correlation for unsupervised network anomaly detection. In: Proceedings of the 7th International Conference on Network and Services Management. pp. 73–80. International Federation for Information Processing (2011)
12. Soule, A., Salamatian, K., Taft, N.: Combining filtering and statistical methods for anomaly detection. In: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement. pp. 31–31. IMC '05, USENIX Association, Berkeley, CA, USA (2005), http://dl.acm.org/citation.cfm?id=1251086.1251117
13. Vanerio, J., Casas, P.: Ensemble-learning approaches for network security and anomaly detection. In: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks. pp. 1–6. ACM (2017)