Future Generation Computer Systems 93 (2019) 418–426

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Anomaly detection in wide area network meshes using two machine learning algorithms

James Zhang, Robert Gardner, Ilija Vukotic *

University of Chicago, Physics Research Center, 933 East 56th St., Chicago, IL, 60637, USA

HIGHLIGHTS

- In this paper, we describe a new method for detecting anomalous behavior over network performance data, gathered by perfSONAR, using two machine learning algorithms: Boosted Decision Trees (BDT) and Simple Feedforward Neural Network.
- The effectiveness of each algorithm was evaluated and com pared.
- Both have shown sufficient performance and sensitivity to disruptions in network performance.

ARTICLE INFO

Article history:
Received 31 January 2018
Received in revised form 12 June 2018
Accepted 14 July 2018
Available online 17 July 2018

Keywords:
Anomaly detection
Wide area network
Boosted Decision Tree
Neural network

ABSTRACT

Anomaly detection is the practice of identifying items or events that do not conform to an expected behavior or do not correlate with other items in a dataset. It has previously been applied to areas such as intrusion detection, system health monitoring, and fraud detection in credit card transactions. In this paper, we describe a new method for detecting anomalous behavior in network performance data, gathered by the Open Science Grid using perfSONAR servers. Two machine learning algorithms were studied: a Boosted Decision Tree (BDT) and a simple feedforward neural network. The effectiveness of each algorithm was evaluated and compared. Both have shown sufficient performance and sensitivity.

1. Introduction

A network anomaly is a deviation from the normal operation of the network, which is learned through observation and is signified by decreased network performance. In this paper, we are interested in four network performance measurements: throughput, packet loss rate, one-way delay (OWD), and traceroute. Throughput measures the amount of data that can be transferred over a given time interval. The interval was chosen to be 25 s and is deemed not too long to cause undue stress on a link, yet not too short to have unreliable measurements. Packet loss is the percentage of lost packets over the total number of transferred packets. To reach a sensitivity of 10^{-5} we measure it at 10 Hz and average the result in one-minute bins. One-way delay measures delay (in ms) separately for each direction of a path. Traceroute records the path and transition time between the source and destination.

Anomalies can last anywhere from one hour to multiple days (if unnoticed) and can be caused by a multitude of factors. Possibilities include: full connectivity disruption, in which all packets are lost; a device on the path is close to saturation, signified by an increase in one-way delay as packets spend more time in the buffers of the devices; a device on the path is saturated, where packets are lost as a device's buffers overflow, signified by increase in one-way delay; routing changes leading to asymmetrical paths which takes more time to reorder packets, signified by large variance in one-way delays; or dirty fibers and other problems with optics, signified by an increase in packet loss rate.

We used perfSONAR [1] data collected by the Open Science Grid. PerfSONAR is a network measurement toolkit that monitors and stores network performance data between pairs of endpoints ("links"). While perfSONAR is installed on thousands of servers, we are particularly interested in ones that are part of the WLCG (the worldwide Large Hadron Collider computing grid for the CERN LHC experiments) and the OSG (Open Science Grid) meshes. Data from all the servers is collected and transported to an Elasticsearch cluster at the University of Chicago. The cluster also collects and indexes data from the WLCG file transfer service (FTS), wide area network usage data from the WLCG job processing systems, network throughput measurements, etc. Knowledge of available network capacity on all links makes it possible for the WLCG experiments to optimize performance of their distributed computing systems:

* Corresponding author.
E-mail address: ivukotic@uchicago.edu (I. Vukotic).

https://doi.org/10.1016/j.future.2018.07.023
0167-739X/

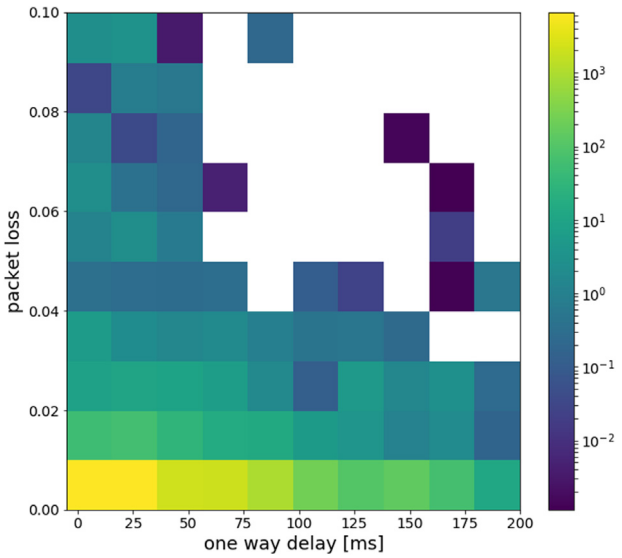

Fig. 1. The relation of packet loss (%) to transfer throughput. The histogram has one entry per link monitored.

data placement decisions, job brokering to computing centers that do not have the data to be processed but have a good connection to a center that does, etc. PerfSONAR data can also be used for capacity planning and network configuration decisions. Collected traceroute data makes it possible to do precise network tomography, thus attributing a saturation or other anomalous behavior to one or several hops.

The WLCG/OSG network mesh size is extremely large. With an ever-increasing number of links and issues in network performance, it becomes increasingly difficult to identify where, when, and why these issues arise and whether or not they are significant enough to ignore. Furthermore, due to the high variance and quantity of data collected, it becomes difficult to analyze and develop models of normal network behavior and of anomalies. Machine learning algorithms are thus favorable as they can learn what is normal behavior and subsequently what is anomalous behavior, and adapt to changes in the structure of normal data.

Ideally an optimal anomaly detection method would satisfy the following criteria: Have the capability to naturally combine disparate data features and data sources, e.g. all the links to/from a site, packet loss, OWDs, paths, throughput measurements, FTS measurements, etc.; give information on which features (or combinations of features) are causing the anomaly and alert the appropriate persons responsible for solving the issue; have tunable sensitivity, as we are not interested in short duration (order of 1 h) flukes as no action could be taken at time scales shorter than that; and perform at a practical level.

1.1. Packet loss

We are particularly interested in packet loss due to its extraordinary influence on transfer throughput. As seen in Fig. 1, even a 0.02% change in packet loss can cause a 10^3 magnitude change in transfer throughput.

1.2. Related work

A number of methods exist for anomaly detection in time series data from single or multiple performance indicators. The simplest methods use amplitude and/or duration of mean and variance as indicators of anomaly significance, or some variant of ARIMA methods. Slightly more powerful are principal component analysis, wavelet analysis, Kalman filter, covariance matrix analysis, and online change-point detection methods. These methods are not able to benefit from the fact that significant anomalies simultaneously affect multiple performance metrics, often across multiple network links. Machine learning techniques have been widely used in anomaly detection [2]. Su presented a density method of using the k-nearest neighbors algorithm (kNN) to detect denial-of-service attacks [3]. Sakurada et al. published a reconstruction method of using autoencoders to detect anomalies on both artificial data generated from Lorenz system and real data from a spacecraft's telemetry data [4]. Rajasegarar et al. described a boundary method of using one-class support vector machines for anomaly detection in sensor networks [5]. A dynamically adaptive plateau-detection (APD) was used on perfSONAR data as described in [6]. Recently, Catmore proposed a split-sample classification method and other ideas for anomaly detection [7]. This paper presents two new methods based on split-sample classification and reconstruction.

In the case of anomaly detection of active network monitoring data, Calyam et al. presented a dynamically adaptive plateau-detection (APD) method based on reinforcement learning [6,8]. Zhang et al. proposed a PCA-based approach for correlated anomaly detection [9]. One of the major differences between our approach in this paper and the methods in [6,8,9] is that we use supervised boosted decision trees (BDT) and feedforward neural network machine learning algorithms whereas the authors in [6,8,9] used reinforcement learning and unsupervised PCA. Another major difference is that our method utilizes data samples in consecutive time intervals for anomaly detection model training and prediction while the methods in [6,8] rely upon the mean and standard deviation of the samples in a defined buffer to detect anomalies.

2. Datasets

We use two different datasets to test the functionality of the new machine learning algorithms as applied to network anomaly detection.

2.1. Simulated dataset

A simulation of the actual data was generated to test the functionality of our methods. We defined six time series (features) spanning a seven-day period from 08-01-2017 00:00:00 to 08-07-2017 23:59:59. Data for each time series was assigned a value between 0 and 1 and was generated for each second. Normal data was generated for each time series by generating a random value less than 0.5 and then generating random values with a normal distribution between 0.00625 and 0.05 around that number. Normal data was flagged as 0. Anomalous data was 2 or 5 standard deviations from the normal noise distribution. Anomalous data was generated six times throughout the entire time period and had a maximum duration of 4 h. Fig. 2 is an example of simulated data.

2.2. Real-world datasets

The data collected between various links measuring their one-way delay, throughput, and packet loss was used to test the functionality of our methods to see if they work on a practical level. Figs. 3 and 4 show examples of real-world data that would be analyzed.

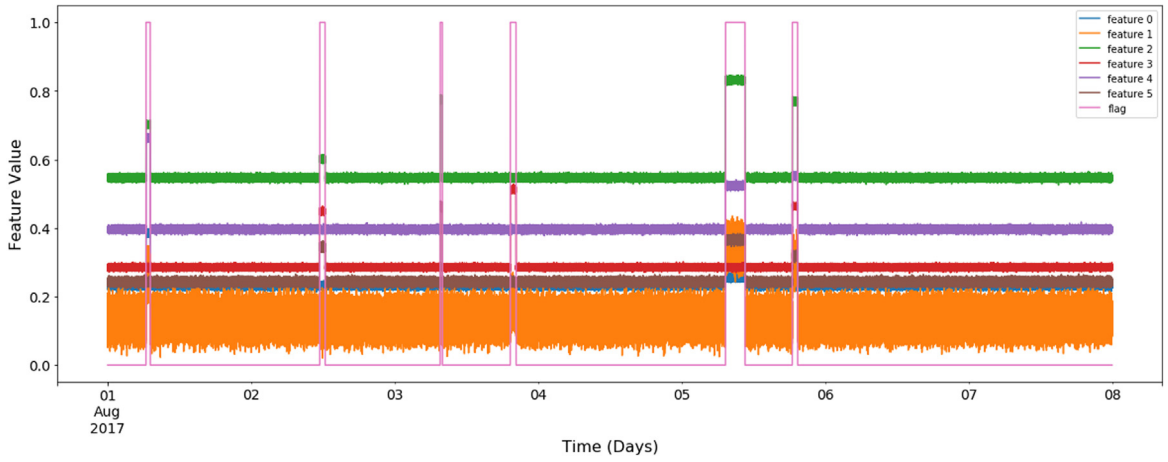

Fig. 2. Example of simulated data. Each link, all assigned a specific color, is a different feature of the dataset and the pink columns signify anomalies. The vertical axis is the value of the data point for each feature and has arbitrary units.

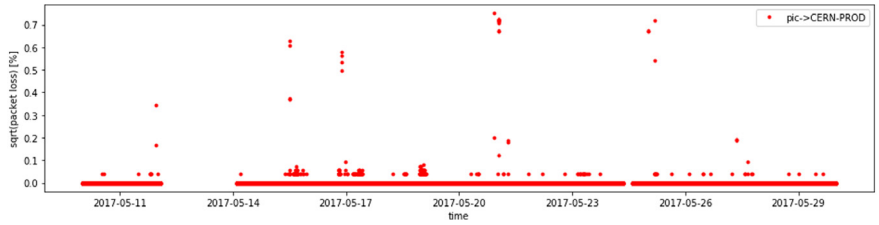

Fig. 3. Packet loss measured over a 20 day period between 2017-05-10 and 2017-05-30 for one link where the source is PIC and the destination is CERN. CERN is a Tier-0 site in Geneva, Switzerland, PIC is the Spanish Tier-1 center.

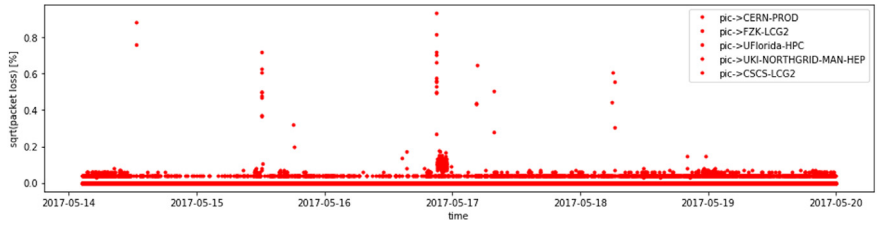

Fig. 4. Square root of packet loss (%) over a six-day period between 05-14-2017 and 05-20-2017 for 5 WLCG sites: CERN-PROD, FZK-LCG2, UFlorida-HPC, UKI-NORTHGRID-MAN-HEP, CSCS-LCG2.

3. New anomaly detection method

There are numerous algorithms for anomaly detection in time series. Among more general methods used are e.g. ARIMA [10], SVM [11], and more specific ones e.g. Bayesian inspired approaches [12]. We tested one-class support vector machine method, while it showed both high specificity and sensitivity on a small set of time series, when employed on a large space (data from more links or additional features), it suffered from the curse of dimensionality. Simultaneous change point detection method from [12] showed unacceptable computational cost at our data scale. Since we have no annotated historical data we are limited to unsupervised methods. The amount of data that must be continuously processed excludes several otherwise promising approaches. Similar to the split-sample classification method [7], in the following we compare time-dependent features in the period under examination (subject period) and in the directly preceding period (referent period). We do this by trying to train a Boosted Decision Tree (BDT) [13] or a neural network (NN) to correctly assign unlabeled samples to subject or referent periods. We expect that any significant difference in data between the two periods will be exploited by the BDT/NN. It follows that the accuracy of the classification will be proportional to the dissimilarity of the samples. We selected the referent periods of 24 h as significantly long to not be sensitive to short duration incidents but short enough to capture long-term changes in link performance. Subject period of one hour is roughly the minimal period that one could expect a human intervention to happen in case alert was received. For both approaches, we flag the reference data with zero and the subject data with one. Further, 70% of the data from both the reference and subject periods are combined and used to train the machine learning models. Training effectiveness was then tested on the remaining 30% of the data.

3.1. Boosted decision trees

A decision tree is a rule-based learning method [2] that creates a classification model, which predicts the value of a target variable by learning simple decision rules inferred from the data features. A decision tree of depth 1 is known as a decision stump.

Decision trees make a split based on the Gini impurity, a measure of how often a randomly chosen element from the dataset would be incorrectly labeled if it were labeled randomly. A higher Gini impurity suggests a less pure split.

For a set of items to be classified in n classes, where $i \in \{1, 2, \ldots, n\}$, and p_i denoting the fraction of items correctly labeled as class i in the set, the Gini impurity can be calculated as follows [14]:

$$I_g(p) = 1 - \sum_{i=1}^{n} p_i^2 \quad (1)$$

As examples of decision trees and stumps, Figs. 5 and 6 show decision trees of different depths. We apply AdaBoost [15], a boosted decision tree algorithm, to our datasets. Boosting is a family of machine learning algorithms that start from weak classifiers, i.e. classifiers that label examples slightly better than random guessing, and return a weighted vote of all of them, the result of which is much more accurate at labeling.

Boosted decision trees train a decision tree by dividing data on one of the features. Then, it validates the decision tree against the training data to find misclassified data values. Each value is assigned a weight, determining its importance. The weight of misclassified values is increased and the weight of correctly classified values is decreased, then a new tree is built. Afterwards, it uses the original tree and the newly formed tree and tests against the training data again to find misclassified values. Another tree is formed, and this process is repeated for however many weak classifiers, i.e. estimators, desired. Each tree will produce a weighted vote between 0 or 1 on whether or not something is an anomaly. The majority vote is used to determine the classification (e.g., anomaly or not) of datasets.

The trees' classification performance is used to determine its true positive rate (i.e. the number of positive events that were correctly identified as such) and false positive rate (i.e. the number of negative events wrongly categorized as positive). These rates are then used to determine an Area Under Curve (AUC) score. The AUC score is the area under the Receiver Operating Characteristic (ROC) Curve, which plots the true positive rate against the false positive rate, and is a popular metric for binary classification.

3.2. Simple feedforward neural network

With recent advances in both hardware performance and availability (GPUs), and software stack (Keras, Tensorflow), it became possible to relatively quickly train neural networks with a large number of trainable parameters. We chose to start with a neural network consisting of one input layer with as many ReLU activation neurons as the number of time series data under investigation, one hidden layer with twice as many ReLU neurons and one sigmoid activated output neuron. We selected this topology as surely sufficient to capture any effects in training data. In a future work, we will investigate the performance of a simpler, one layer network as this would simplify finding which time series contributed most to a period being flagged as anomalous. In order to increase performance, we will also optimize learning rates, try different neuron activations, optimizers, etc. Training and testing data were prepared in the same way as for the BDT model, except that training data gets shuffled after each epoch. We train for 60 epochs in batches of 256 samples using Adam optimizer [16] and as a loss function use binary cross-entropy. Training for one period takes around 20 s on a Tesla 20 K NVidia GPU and roughly three times less on the NVidia GTX 1080Ti. How well the trained network performed on the test samples is given by the binary accuracy. An anomaly is flagged based on how likely an accuracy of that magnitude is to happen by pure chance.

4. Experimental results

4.1. Boosted decision trees

In this section, we study the effectiveness of BDT in detecting unusual events in network performance data. We primarily analyze packet loss and one-way delay. We chose not to analyze traceroute and throughput, because throughput is measured too infrequently to be included as a parameter and traceroute is not just a variable, it consists of a list of nodes, so it cannot be trivially included. In the future, we plan on calculating the hash from it and using that as a feature. 50 estimators were used on the testing data. Both accuracy and AUC score were determined, and if the AUC score was above a certain threshold then the result was determined to be an anomaly.

4.1.1. Decision stumps vs. decision trees

We tested the effectiveness of both boosted decision trees utilizing trees of depth 6 vs. that of boosted decision trees utilizing decision stumps on our simulated data. It can be seen in Figs. 7 and 8 that using decision stumps versus using decision trees produced roughly equal results. Both methods were able to identify anomalies of one hour in length. It is interesting to note that both methods were not able to detect the first anomaly. This is most likely because those anomalies occurred before the first day (our method uses a day's worth of data as training data), and as such did not have enough training data previously to identify them.

Though the two results were very comparable, the decision stumps took on average 2.6 seconds per hour worth of data to run, whereas decision trees took 12.2 seconds per hour worth of data, more than four times longer. We determined that decision stumps applied to BDT were a more practical and efficient method for anomaly detection and were thus used for the remainder of the experiment.

4.1.2. Anomaly duration vs. degree of anomalous behavior

We wanted to know which factors had the greatest effect on whether an interval was determined to be anomalous: the magnitude of the offset between an anomaly from normal data, the duration of anomalous behavior, or the number of time series affected. We used the simulated data described in Section 2.1. Six anomalies were generated on the same set of normal data. Fig. 9 shows a visual representation of generated anomalies. The start of each anomaly was separated by 24 h. We fluctuated the duration of anomalous behavior, the number of features affected, and the anomaly offset for each one. The degree of anomalous behavior encompasses both the number of features affected and the anomaly offset.

Data considered to have affected many features affected three times as many as data that was not (3 features vs 1). Data with a long duration of anomalous behavior was three times longer than data (3 h vs 1). We generated offsets (anomalies) of two amplitudes: small — 2σ and large — 5σ shift.

Normal data had an AUC score and accuracy of roughly 0.50 and 0.96 respectively. The threshold for anomalous behavior was defined as any AUC score above 0.55. From the results shown in Table 1 and Fig. 9, we see that the change in the anomaly offset had the most significant effect on whether an anomaly was determined to be so.

Holding the number of features affected and the duration of anomalous behavior constant, an increase in anomaly offset increased the AUC score by 0.379, a roughly 63.91% increase, and increased the accuracy by 0.032. The number of features affected had a less significant effect. Holding the anomaly offset and anomaly duration constant, an increase in the number of features affected increased the AUC score by 0.267, a roughly 45.03% increase. The

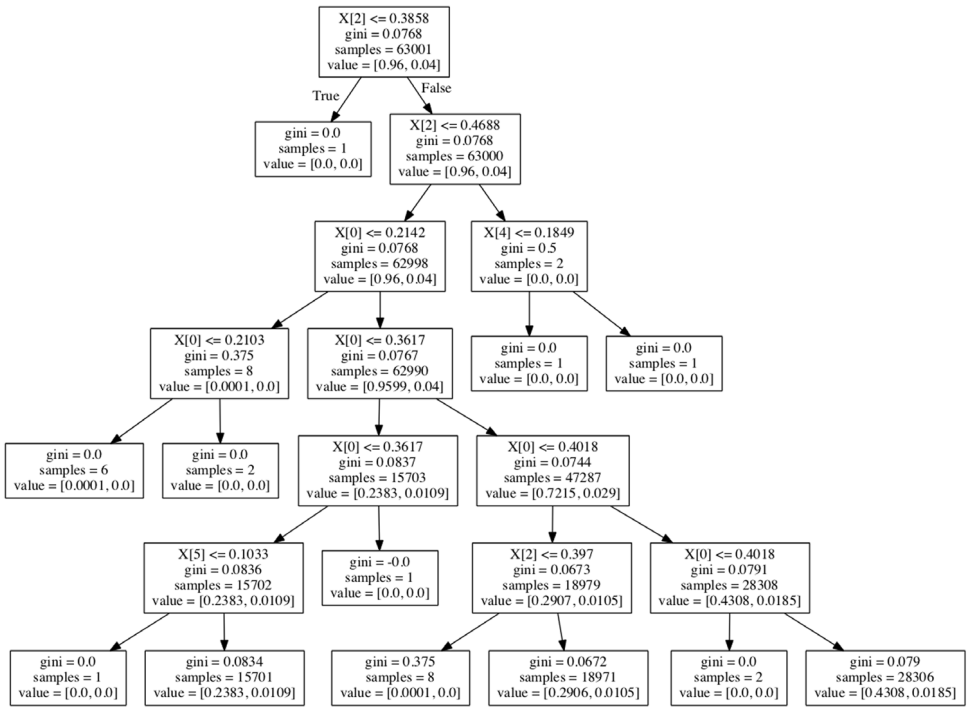

Fig. 5. An example of a decision tree of depth 6.

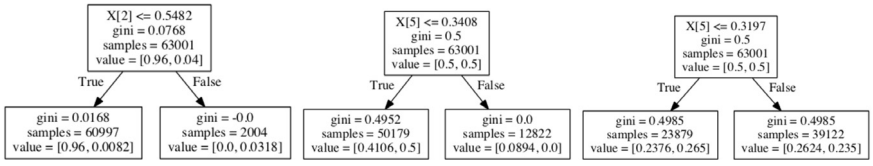

Fig. 6. An example of three different decision stumps or decision trees of depth 1. Taking the first stump as an example, the tree splits the data on feature 2 with a Gini impurity of 0.0768 taking 63001 pieces of sample data. It was able to separate the data whereby the right column has a Gini impurity of 0.0, signifying that it was perfectly categorized.

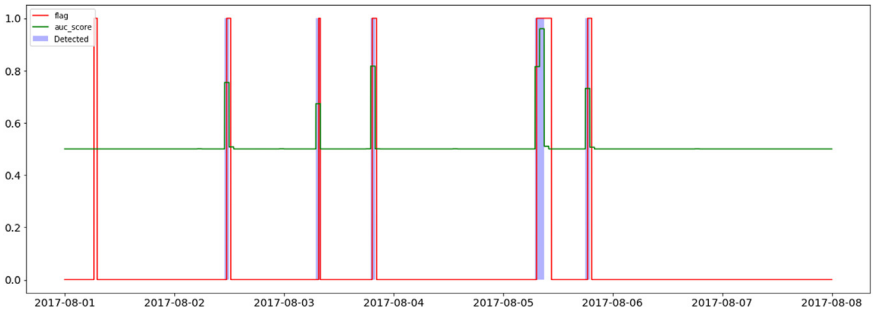

Fig. 7. Results for BDT using decision stumps. The anomaly detection was performed on simulated data of a period of 7 days. Red lines indicate where anomalous data was generated, blue shading indicates where the algorithm predicted an anomaly was, and the green is the AUC score for each hour. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
The AUC scores of six different simulated anomalies all having some combination of anomaly duration, amplitude, and a number of features affected. Anomaly numbers come in the order of when the anomaly was generated. Results above our 0.55 cut level are shown in bold letters.

Anomaly	Offset [σ]	Features affected	Duration [h]	AUC score | Accuracy			
				Hour before	Hour 1	Hour 2	Hour 3
1	2	1	1	0.499 | 0.959	**0.593** | **0.964**		
2	2	1	3	0.500 | 0.960	0.525 | 0.959	0.511 | 0.959	0.501 | 0.959
3	2	3	1	0.499 | 0.959	**0.860** | **0.982**		
4	5	1	1	0.499 | 0.959	**0.972** | **0.996**		
5	5	1	3	0.500 | 0.960	**0.693** | **0.960**	**0.702** | **0.959**	0.502 | 0.959
6	5	3	1	0.500 | 0.960	**0.999** | **0.999**		

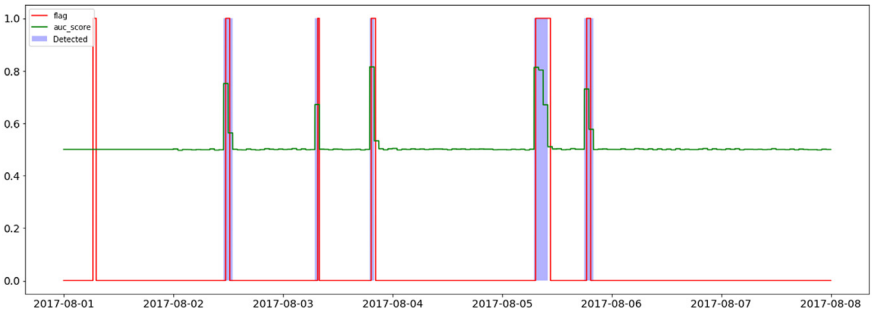

Fig. 8. Results for BDT using decision trees of depth 6.

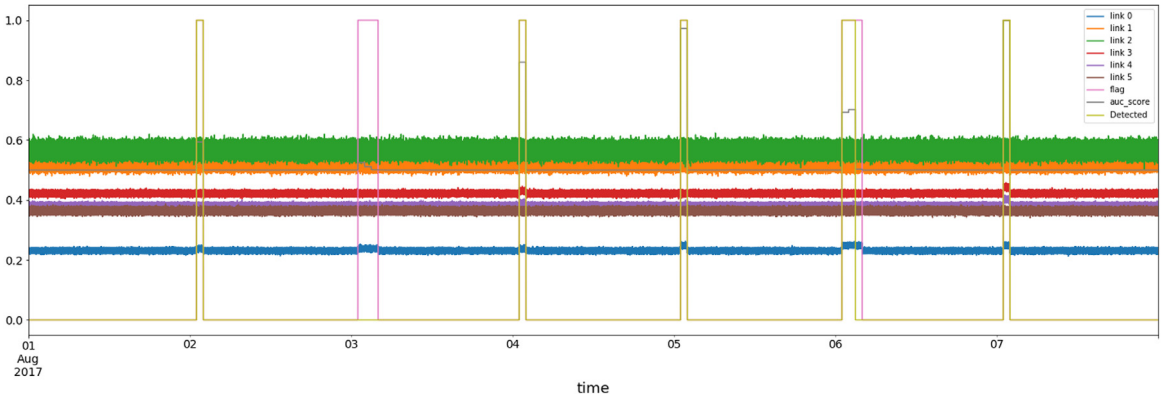

Fig. 9. Six anomalies generated over a 7 day period. Each anomaly has some combination of anomaly duration, amplitude, and a number of features affected.

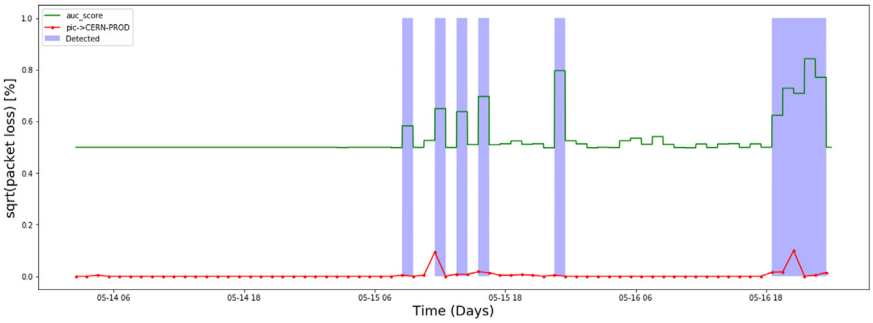

Fig. 10. Boosted decision tree as applied to one-way delay and packet loss data between PIC and CERN-PROD. The light blue columns signify areas where the algorithm detected anomalous activity, using an AUC threshold of 0.55.

duration of anomalous behavior had the smallest effect on determining an anomaly. Holding the other two variables constant, an increase in the duration of anomalous behavior decreased the AUC score by 0.068, a roughly 11.47% change. The results suggest that the extent to which an anomaly is considered as such is more dependent on the anomaly offset and the features affected, that is, the degree of anomalous behavior, more than the anomaly duration.

4.1.3. Application on real data

We tested the effectiveness of using a boosted decision tree on both the packet loss and one-way delay data over a 4-day time span between two sites: PIC and CERN-PROD. The AUC threshold was set to 0.55. From Fig. 10, it is evident that the AUC threshold is too high, as it detects anomalies far too often to be practical.

For each anomaly that was detected, we were able to generate both the feature importance of each time series and an ROC curve,

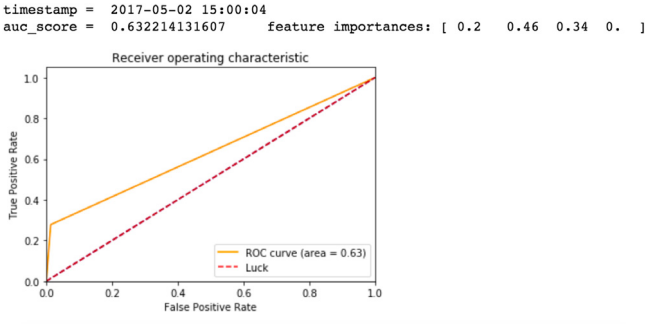

Fig. 11. ROC curve of the first anomaly detected and the relative importance of its features. By looking at the feature importance, we see that feature 2 had the greatest influence on the AUC score, then feature 3, feature 1, and finally feature 4, which did not factor in at all in generating the anomaly.

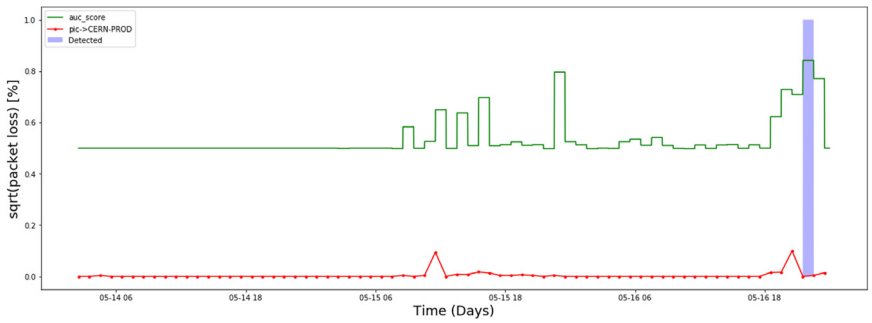

Fig. 12. Boosted decision tree as applied to one-way delay and packet loss data between PIC and CERN-PROD in a 30-day period. The AUC threshold was 0.80.

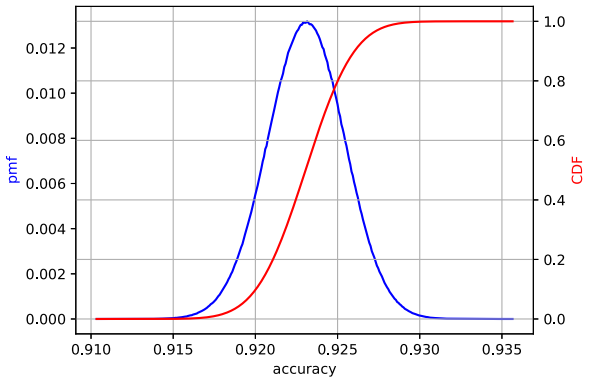

Fig. 13. Distribution of accuracies obtained by chance (binomial distribution) corresponding to 12 h referent and 1 h subject intervals with one sample per second ($p = 0.92308$). Accuracy larger than 0.9285 can happen with $<1\%$ chance.

thus allowing us to tell which time serie(s) caused the anomaly. Fig. 11 shows an example of said importance and curve.

Due to the high frequency at which anomalies were detected, the AUC threshold of 0.55 was determined to be impractical. After adjusting the AUC score to 0.8, the result as seen in Fig. 12, became much more practical to be used.

4.2. Simple feedforward neural network

In this preliminary study, we used two hidden layers with twice as many ReLU neurons as time series and a single sigmoid output neuron. This results in a network with 2521 trainable parameters. The result of the test is a binary classification accuracy — defined as a ratio of correctly labeled and total number of samples. The distribution of accuracies that can be expected by chance depends only on the number of samples in referent and subject intervals and can be seen in Fig. 13.

4.3. Application on simulated data

We used the simulated data described in Section 2.1. A 12 h period prior to the 1 h subject period was used as a reference. This choice gives an accuracy of 0.923 by pure chance, and we consider anomaly detected if the calculated accuracy has less than 1% chance of appearing randomly (0.9285). From the results shown in Table 2 and Fig. 14, we see that only the last three anomalies (with offsets of 5σ) have been detected, and only in the first hour of anomaly appearance. Accuracy levels during the first three anomalies and during the last two hours of the fourth anomaly were not simply under the threshold but actually exactly equal to the pure chance accuracy. As it can be seen from Fig. 15, loss and accuracy change in a stepwise manner and in case backpropagation optimization does not find any minima, the result will be equal to chance. This does not limit the applicability of the method as we are anyhow not interested in small effect anomalies affecting single time series, or repeated identification of the same anomaly.

4.3.1. Performance on actual data

When using this method on actual data we use a referent interval of 24 h to average possible anomalies in referent data over longer periods. Data is measured packet loss between CERN and 20 other sites (from 73 that we have values measured for) covering a four day period. There is one data point per minute. Missing data was filled with zero values. Fig. 16. shows results obtained by training the network for 100 epochs for each interval, using a batch size of 10, and having 70:30 split between training and testing data. The mean of the chance accuracy distribution is at 0.96 and chance accuracy of more than 0.97 has less than 1% probability.

5. Conclusion

We presented two new methods of detecting network performance anomaly based on split-sample classification: AdaBoost and simple feedforward neural network. Both methods were first

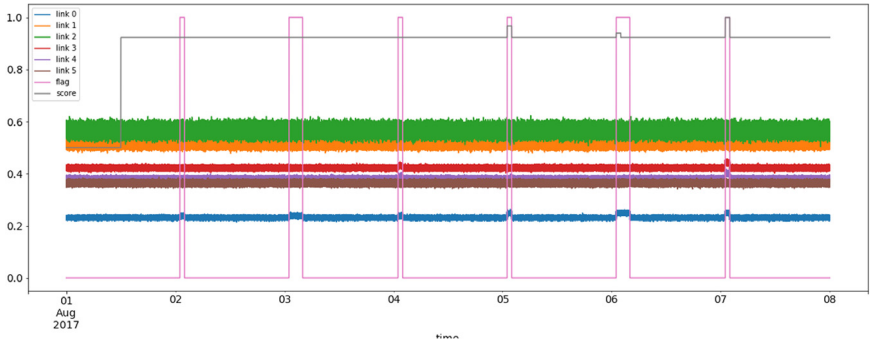

Fig. 14. Six anomalies generated over a 7 day period. Anomalies are described in Table 2. Only the last three anomalies have been detected.

Table 2
Accuracy for six different simulated anomalies, each having some combination of anomaly duration, amplitude, and number of features affected. Anomaly numbers come in the order of time the anomaly was generated. Results above our 0.928 cut level are shown in bold letters.

Anomaly	Offset [σ]	Features affected	Duration [h]	Accuracy			
				Hour before	Hour 1	Hour 2	Hour 3
1	2	1	1	0.923	0.923	0.923	
2	2	1	3	0.923	0.923	0.923	0.923
3	2	3	1	0.923	0.923		
4	5	1	1	0.923	**0.942**		
5	5	1	3	0.923	**0.991**	0.923	0.923
6	5	3	1	0.923	**0.999**		

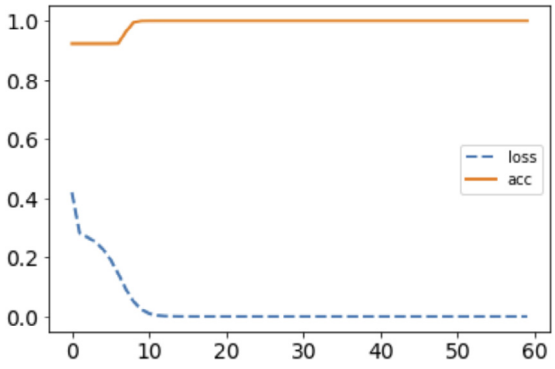

Fig. 15. Loss and accuracy for each of 60 training epochs. Anomaly threshold was already reached at epoch 15. One possible optimization is early training termination as soon as the threshold has been reached.

tested on simulated datasets to check their sensitivity with respect to duration and amplitude of anomaly. The boosted decision tree method proved to be very fast (4 s evaluation per one hour of sampled data tested) and detected all the simulated anomalies. An added benefit of this method is that it directly returns an ordered list of series according to their contribution to the anomaly being flagged. With an appropriately selected AUC threshold it is possible to tune the desired sensitivity/false positive level. The simple neural network model used was not hyper-parameter optimized and the one network tried proved less sensitive to short and low amplitude changes. Given that we are looking for the most significant anomalies this is a good feature. While the evaluation is slower at 20 seconds per hour of data tested, it is still fast enough to be of practical use. A more significant issue is that it requires a GPU for processing. Since this is a three-layer network it is difficult to get information on the importance of different time series to the resulting decision. While results on the actual data are encouraging, before using it in a production environment different network configurations should be tested (two layers, fewer neurons per layer, etc.) and hyper-parameter tuned.

6. Future work

Once the simple neural network model is fully optimized, a set of data should be independently annotated by several people. The annotations should be compared and a self consistent set produced. Both methods described should be evaluated against it. A new set of tests should be done, this time training models on multiple features per link (e.g. OWD and Packetloss). The NN model should be created, such that it not only detects anomaly but also classifies it into one of the few classes (e.g. overloaded link, unstable path, etc.). Finally, brand new methods should be tested: Data smashing [17], LSTM Deep neural networks [18], etc.

Acknowledgments

This work was supported in part by the National Science Foundation, United States grants: NSF PHY-1148698 The Open Science Grid and NSF PHY11-19200 U.S. ATLAS Operations: Discovery and Measurement at the Energy Frontier.

Appendix

All the codes and test data can be found in this repository: https://github.com/ATLAS-Analytics/AnomalyDetection

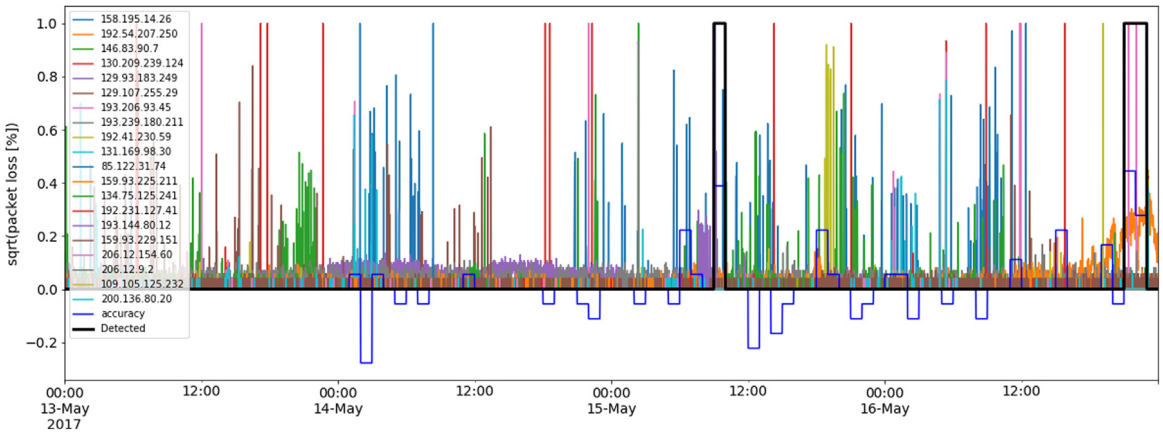

Fig. 16. Packet loss between CERN and 20 other sites (here given as an IP address of their PerfSONAR node). Shown is the square root of the value to better visualize small values. The blue line shows relative binary classification accuracy change from chance (in percents). It is calculated as a (*measured* − *chance*)/(1 − *chance*). Thick black line (Detected) marks intervals flagged as anomalous. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

[1] B. Tierney, J. Metzger, J. Boote, E. Boyd, A. Brown, R. Carlson, M. Zekauskas, J. Zurawski, M. Swany, M. Grigoriev, perfsonar: Instantiating a global network measurement framework.

[2] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3) (2009) 15:1–15:58. http://dx.doi.org/10.1145/1541880.1541882.

[3] M.-Y. Su, Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers, Expert Syst. Appl. 38 (4) (2011) 3492–3498. http://dx.doi.org/10.1016/j.eswa.2010.08.137.

[4] M. Sakurada, T. Yairi, Anomaly detection using autoencoders with nonlinear dimensionality reduction, in: Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis, MLSDA'14, ACM, New York, NY, USA, 2014, pp. 4:4–4:11. http://dx.doi.org/10.1145/2689746.2689747.

[5] S. Rajasegarar, C. Leckie, J.C. Bezdek, M. Palaniswami, Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks, Trans. Info. for. Sec. 5 (3) (2010) 518–533. http://dx.doi.org/10.1109/TIFS.2010.2051543.

[6] P. Calyam, J. Pu, W. Mandrawa, A. Krishnamurthy, Ontimedetect: Dynamic network anomaly notification in perfsonar deployments, in: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2010, pp. 328–337. http://dx.doi.org/10.1109/MASCOTS.2010.41.

[7] J. Catmore, Ideas on anomaly detection for data quality monitoring. (2016). URL http://bit.ly/2GtohJ7.

[8] P. Calyam, M. Dhanapalan, M. Sridharan, A. Krishnamurthy, R. Ramnath, Topology-aware correlated network anomaly event detection and diagnosis, J. Netw. Syst. Manage. 22 (2) (2014) 208–234. http://dx.doi.org/10.1007/s10922-013-9286-0.

[9] Y. Zhang, P. Calyam, S. Debroy, M. Sridharan, Pca-based network-wide correlated anomaly event detection and diagnosis, in: 2015 11th International Conference on the Design of Reliable Communication Networks (DRCN), 2015, pp. 149–156. http://dx.doi.org/10.1109/DRCN.2015.7149006.

[10] H.Z. Moayedi, M.A. Masnadi-Shirazi, Arima model for network traffic prediction and anomaly detection, in: 2008 International Symposium on Information Technology, Vol. 4, 2008, pp. 1–6. http://dx.doi.org/10.1109/ITSIM.2008.4631947.

[11] J. Ma, S. Perkins, Time-series novelty detection using one-class support vector machines, in: Proceedings of the International Joint Conference on Neural Networks, 2003. Vol. 3, 2003, pp. 1741–1745 vol.3. http://dx.doi.org/10.1109/IJCNN.2003.1223670.

[12] N.R. Zhang, D.O. Siegmund, Model selection for high-dimensional, multi-sequence change-point problems, Statist. Sinica 22 (4) (2012) 1507–1538.

[13] J.H. Friedman, Stochastic gradient boosting, Comput. Statist. Data Anal. 38 (4) (2002) 367–378 nonlinear Methods and Data Mining.

[14] D. Coppersmith, S.J. Hong, J.R. Hosking, Partitioning nominal attributes in decision trees, Data Min. Knowl. Discov. 3 (2) (1999) 197–217. http://dx.doi.org/10.1023/A:1009869804967.

[15] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. System Sci. 55 (1) (1997) 119–139. http://dx.doi.org/10.1006/jcss.1997.1504.

[16] D.P. Kingma, J. Ba, Adam A Method for Stochastic Optimization, arXiv e-prints arxiv:1412.6980.

[17] I. Chattopadhyay, H. Lipson, Data smashing, CoRR abs/1401.0742 arXiv:1401.0742. URL http://arxiv.org/abs/1401.0742.

[18] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.

Robert Gardner is Research Professor of Physics in the Enrico Fermi Institute and Senior Fellow in the Computation Institute at the University of Chicago. He directs the Midwest Tier2 Center for the ATLAS experiment at the CERN Large Hadron Collider and is the integration program manager for the U.S. ATLAS Collaboration's Computing Facilities, which includes the Tier1 center at Brookhaven National Laboratory and ten university Tier2 sites. He leads the Open Science Grid user support team, is co-principal investigator of VC3: Virtual Clusters for Community Computation, a DOE ASCR award to deploy virtual cluster systems over diverse HPC centers, and is the PI of NSF CIF21 DIBBs: EI: SLATE and the Mobility of Capability.