Outlier Detection in Data Streams Using OLAP Cubes

Felix Heine(✉)

Department of Computer Science, Faculty IV,
Hannover University of Applied Sciences and Arts,
Ricklinger Stadtweg 120, 30459 Hannover, Germany
felix.heine@hs-hannover.de

Abstract. Outlier detection is an important tool for many application areas. Often, data has some multidimensional structure so that it can be viewed as OLAP cubes. Exploiting this structure systematically helps to find outliers otherwise undetectable. In this paper, we propose an approach that treats streaming data as a series of OLAP cubes. We then use an offline calculated model of the cube's expected behavior to find outliers in the data stream. Furthermore, we aggregate multiple outliers found concurrently at different cells of the cube to some user-defined level in the cube. We apply our method to network data to find attacks in the data stream to show its usefulness.

1 Introduction

Outlier detection is an important basic technique used in many application areas including data quality, fraud detection, or intrusion detection in networks. In this paper, we look at outlier detection in multidimensional data. We present a novel method for outlier detection that exploits the multidimensional nature of data. It is a one-class anomaly detection approach that includes a training phase using only-normal data. The training is done offline, the detection phase can be run either online or offline. Exploiting the multidimensional structure of data helps to find outliers that are undetectable at the highest aggregation level or to collect more evidence for suspicious data by looking at various multidimensional views.

This also helps to find contextual and collective outliers (see [2]). First, aggregating the data in different ways allows for contextual models by assigning different models to different groupings. Second, collective outliers can be spotted more easily when looking at the grouping where the outlier becomes most evident. In order to define and search the groupings in a systematic way, we treat all data sources as *multidimensional data cubes*. For most data, this is a natural way to model the data. For example, network data can be grouped and aggregated e.g. using target IP ranges, source IP ranges, and protocols at different layers.

This work has partly been developed in the project IQM4HD (reference number: 01IS15053B). IQM4HD is partly funded by the German ministry of education and research (BMBF) within the research program KMU Innovativ.

M. Kirikova et al. (Eds.): ADBIS 2017, CCIS 767, pp. 29–36, 2017.
DOI: 10.1007/978-3-319-67162-8_4

The advantage of this approach is the natural treatment of collective and contextual outliers. We integrate the anomaly detection results coming from different views of the data to *collect evidence* and to build a final score for each entity of interest. The method itself is general. Domain knowledge is only needed to properly define the cube's dimensions and metrics.

The main contribution of this paper is a one-class method to find anomalies in multidimensional data cubes using models that describe the normal behavior of the data at a fine-grained level. This method is complemented by a second method that combines anomaly scores that were calculated for different data views and to relate these scores to relevant *units of inspection*. These units are application dependent and define the granularity of the generated anomaly events. For network data, a useful unit of inspection is a single network connection. As an application example, the method is applied to network data. We provide an evaluation using data from the 1998 DARPA IDS evaluation.

The next section describes the proposed method in detail on a generic level that is independent from a specific application. Transfer to a network example scenario is done in the evaluation Sect. 3. Before concluding, related work is described. Please note that we assume that the reader is familiar with basic OLAP cube terminology and iceberg cubes. For an introduction, see [4,14].

2 Anomaly Detection in Multidimensional Data

In this section, we describe our approach. The basis is time-related multidimensional data, i.e. a stream of tuples that each contain a time-stamp, multiple dimensional attributes (categorical) and multiple metric attributes (continuous). We furthermore assume to have recorded old data that well reflects normal behavior and can be used to train the system.

We split both training and test data into a sequence of cubes by accumulating data over an interval ΔT. In the training phase, this is done offline. Here, we build the models that describe the normal behavior of the data. During the online operation, we collect data until the interval is completed and then build the cube for the past interval. We then use the models to find outliers in the current time slice. As an outlier will be visible in multiple aggregation views, we furthermore integrate multiple outlier scores in a single score per unit. We now describe details for both phases.

2.1 Training Phase

The basic idea is to build a separate model for each cell of the cube, and to check online data against this model. The training data must not contain anomalies. As a basic model, we start with a Gaussian model, however, more complex models are possible.

The model should be able to assess the outlier score for the metric values in a single cell as well as the correlation between different cells. Our basic model captures the distribution of each metric in individual cells without inter-relationships

between the metrics as a univariate Gaussian model. Thus it is easy to calculate model parameters (estimated standard deviation and estimated mean), and easy to apply (calculation of Z-scores). More complex models could fit a time series model to a cell, e.g. an autoregressive model. However, already this simple model leads to a huge number of parameters, when looking at the whole cube, thus we aim to limit the number cells where a model is created. The number of underlying data samples (size of a cell) is a good indicator to do the pruning. We use a system-wide parameter T_m as a threshold for the minimal number of time steps where the data must be present for a cell in order to build an individual model for this cell. This avoids models that are not backed by enough data and limits the size of the overall model.

Beside the behavior of individual cells, we also aim to model the relationship between cells. We capture the relationship between each cell and all of its parent cells by providing a model for every direct connection in the cube lattice. In the initial implementation, this model uses the ratio between the parent cell's metrics and the child cell's metrics. As long as we restrict the aggregation functions to summation over non-negative values, we know that the parent's metric is always larger or equal to the child's metric. This leads to a ratio between 1 and 0, that we then model using a univariate Gaussian.

2.2 Online Operation

During online operation, we collect data for each time interval (again of size ΔT) and build an iceberg cube as soon as the interval is complete. Here, the iceberg condition is a cell with a minimal count of T_i underlying base records. For each cell that appears in the iceberg cube, a model is looked up. If one is found, we calculate an outlier score and store it to an *outlier cube*. Also for all upper level cells in the cube lattice, the scores are calculated and stored.

There will be lots of cases where no fitting model is available. This happens either when the training data did not contain the test cell, or when the training data contained fewer than T_m time intervals with data for the test cell. In this case, the cell model was pruned from the model. So we cannot compute any score for this cell. However, due to the collection of scores from related cells, we are still able to compute a score for each unit of interest.

The result of the first step is an outlier cube, that contains anomaly score facts corresponding to the facts in the base cube. In the second step, we relate the anomaly scores to the units of inspection. In the case of network data, the unit of inspection is a connection, corresponding to a base cell of the cube. However, this does not need to be the case.

For each cell that corresponds to a unit of inspection we collect all related scores. These are all scores from either ancestor cells or descendent cells of the inspected cell in the outlier score cube. We store all scores in a single record that integrates all evidence that is related to the inspected unit. In this record, we have scores related to the correlation models, and there is one score per metric, both for the basic models and the correlation models.

This results in a large, however fixed number of scores for each unit. The number is fixed as it only depends on the cube model and not on the current data instances. For our sample cube, there are more than 1000 individual scores per unit. Each additional dimension would multiply this number. However, due to iceberg conditions, not every score might be present for every unit.

We have multiple options to structure this record. In order to lose as few information as possible, we could reserve one column for each potential ancestor or descendent, resulting in high-dimensional records. To make the dimensionality smaller, we map outlier scores to the distance in the cube lattice. By this, we mean the length of the shortest path from the inspected cell to the outlier score cell. Thus the cell itself has distance zero, the parent cells have distance one, direct children have distance -1, and so on. In the case of our example network cube, we have 11 different distances.

In case a unit lives for more than one time unit, we have multiple such records. For the final result, we build both the average score per distance over all time units and the maximum score over all time units. So the final score record for each unit looks as follows. By mind and maxd, we mean the minimum and maximum distance of any cell in the cube to the units of interest. These values only depend on the cube structure and are thus fixed for a given cube. In our example, mind is 0 and maxd is 10.

$$(\text{avgscr}_{\text{mind}}, \ldots, \text{avgscr}_{\text{maxd}}, \text{maxscr}_{\text{mind}}, \ldots, \text{maxscr}_{\text{maxd}})$$

In the final step, all scores contained in the records that were build in the last step must be integrated into a single score that is used to decide whether the specific unit is regarded to be an outlier or not. Each individual attribute of the record describes some evidence. When there is a huge score in the cell itself (distance 0), this is a clear indicator for an outlier. However, not all outliers will be visible at this layer. Some will only be visible when looking at more distant cells. In a denial of service attack, for example, the individual connection itself does not look suspicious, while the aggregated number of connections from various sources to a single target might generate an outlier score. In the other extreme, an outlier score at the apex cell is very unspecific, so it should not make every cell at the same time an outlier. So we propose to weight the scores using their distance to the relevant cell:

$$\text{score} = \sum_{d=\text{mind}}^{\text{maxd}} \frac{1}{|d|+1}(\text{avgscr}_d + \text{maxscr}_d)$$

3 Evaluation

In this section, we evaluate the method using data from the network domain. We use the well-known dataset from the 1998 DARPA IDS evaluation.[1] The goal of the evaluation is to gain insights in the quality of the solution and to explore the effects of different parameters with respect to the detection capability. Here, the DARPA data is very useful as we have ground truth available.

[1] See https://www.ll.mit.edu/ideval/data/1998data.html.

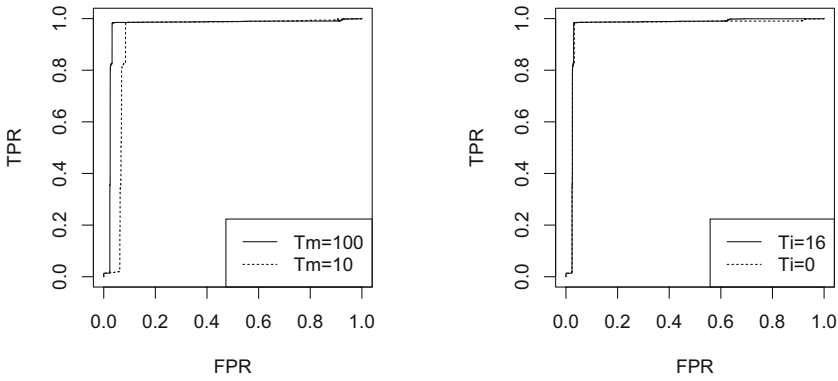

Fig. 1. Evaluation results.

First, we preprocess the training data by removing all attacks. Thus we start with training data that only contains normal data, while the test data still contains attacks *and* normal connections. We model a cube from the data including different metrics like packet count, data volume and specific metrics e.g. for capturing specific flags. The system also adds by default another fact that counts the number of base cells below the current cell in the lattice that have at least one record. For the dimensions, we first use the IP source and destination address. We build a hierarchy for these dimensions based on the /24, /16, and /8 address prefixes. IP communication will have two additional dimensions that indicate the layer 4 and layer 7 protocol fields. In layer 4, this field differentiates between UDP, TCP, ICMP, or other protocols. In layer 7, the initial target port (well known port) for a UDP or TCP connection will be used. For ICMP, the command type is used. These two protocol attributes build a protocol hierarchy.

The DARPA data contains different types of attacks. We will focus on DoS attacks, as they result in typical outlier patterns in the data that we aim to detect with our method. The main parameters are the threshold used to build the models (T_m), and the threshold used to build the current time frame's iceberg cube (T_i). First, we compare the results for model thresholds $T_m = 10$ and $T_m = 100$; see the left side of Fig. 1. The detection is much better at a larger model threshold value. This is an indicator that the models build with only a few sample time frames in the training data are too volatile. They seem to introduce too specific normal behavior so that later on different behavior is flagged wrongly as anomalous behavior. For both curves, we set $T_i = 0$.

Now we look at what happens when we restrict our attention to cells that fulfill a certain iceberg condition. In the right graph of Fig. 1, the detection capability for DoS attacks is shown with an iceberg threshold of $T_i = 0$ to $T_i = 16$ during online operation, meaning we exclude cells with fewer than 16 packets for a given time frame. Here, the detection capability is nearly unchanged. This is quite intuitive, as outliers consist of a larger data volume and thus are typically

part of the iceberg cube. This means that restricting the computation to iceberg cubes is a good way to improve performance without losing quality.

Finally, we provide *preliminary* performance insights for the online phase. We processed two weeks of network data. The overall throughput is averaged to 30 MBit/sec. The results where measured on a commodity PC with Intel i7 processor. However, this includes heavy I/O overhead as we are currently logging many intermediate results. Thus we think it is possible to analyze data streams with much higher data rates in an online fashion.

4 Related Work

In this section, we compare our work to similar work in the area of outlier detection (see e.g. [1,2]). In [5], Le and Han describe the SUITS algorithm to find anomalies in OLAP cubes viewed as multi-dimensional time-series data. However, they focus on unsupervised outlier detection. Their notion of outliers assumes that time series from child cells normally have the same shape compared to the parent cell at another scale. Any cell that deviates from this assumption is an outlier. This is fundamentally different to our approach of taking past data as a normal reference and is only useful for offline processing. For an alternative approach to unsupervised outlier detection in OLAP cubes, see [6]. A similar outlier definition based on unsupervised analysis of an OLAP cube is given by Sarawagi et al. [11]. For each cell, the outlier score stems from the relative deviation to an "expected value", which is computed from related cells in various ways. They do not have a special processing for the time dimension. The outlier score is used in an interactive environment to guide the user to interesting cells of the cube. Also Dunstan et al. [3] describe the idea to present the outliers in reports using various groupings. This resembles our idea of collecting evidence from different related cells. Palpanas et al. [9] give a method to reconstruct a cube's base data based on marginal distributions (i.e. aggregate values) using maximum entropy estimation. By comparing the reconstructed values and the actual values, outliers can be identified.

There is also work that looks at OLAP cubes in data streams. In [4], stream cubes are defined. However, in order to reduce the number of cells, the authors use a minimal interest layer. This contradicts our idea to look also at very detailed levels when there is enough data present. Restricting the cube to a minimal interest layer might lead to overlooking important hints for outliers. In Rettig et al. [10], an infrastructure is proposed to find outliers in streaming data from the telco domain in real time. There are some similar ideas, as e.g. the cell[2] usage data has multidimensional structure including a region and an event type dimension. However, the aggregation is specific to the application example in contrast to our generic method. Furthermore, data is compared to older time steps for outlier detection and not to any model.

In [12], the authors follow an approach using OLAP cubes to detect outliers in wireless networks. Apart from a different outlier score definition,

[2] A mobile network cell, not a cube cell.

the main difference to our work is that they use domain expertise to predefine those cuboids that are explored in search for outliers. Our method explores all cuboids dynamically that contain enough data (iceberg).

There is a large body of work about outlier detection in high dimensional data, see e.g. Chap. 5 of [1]. However, as pointed out in [5], the data model of the OLAP approach is different. Outliers are defined in terms of distance to other data points in the high-dimensional space, and an important approach to finding outliers is to look at the data using different projections. This is in contrast to our method. We look at each metric (i.e. feature) individually, which means that we use one-dimensional projections in the feature space. However, we use different subsets of the data, with aggregated values of the individual metric values based on the OLAP dimensions. The only similarity is on a very high level: to find outliers, it is beneficial to look at the data from multiple different perspectives. Müller et al. [8], as an example for this line of work, explore subspace clusters and collect outlier indicators from these subspaces. However, a single outlier is always a single data row, no collective outliers can be found with this method.

5 Conclusion

This paper presents a new approach to outlier detection in a stream of multidimensional data records. The basic idea is to calculate models specific to various groupings of the data, using an offline training phase and normal-only training data. This makes the approach a one-class anomaly detection method. In the online detection phase, a cube is built from the data for a series of small time intervals. The models are used to calculate multiple outlier scores at different cells of the cube.

The outlier scores in these cubes are then related to interesting real-world entities in order to collect evidence for the outlierness of these entities. If entities live for multiple time steps, further evidence is collected from subsequent cubes. The collected scores are condensed into a single outlier score for each entity.

To make the computation efficient and to avoid volatile models, only cube cells with enough training data will be included in the cube model. During online operation, a cube cell without a model is ignored. However, due to evidence from other cells, outliers can still be found. Furthermore, test data cube cells with too few data can also be ignored, using an iceberg cube approach.

An evaluation based on an network data scenario shows that the method is able to distinguish between normal data and attacks very well. The evaluation also shows the effects of different parameter settings, with the conclusion that using appropriate thresholds for the model is indeed important to avoid spurious models. Restricting the attention to the iceberg cube during processing does not lower the detection quality.

References

1. Aggarwal, C.C.: Outlier Analysis, 1st edn. Springer, New York (2013). doi:10.1007/978-1-4614-6396-2
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**(3), 15:1–15:58 (2009). http://doi.acm.org/10.1145/1541880.1541882
3. Dunstan, N., Despi, I., Watson, C.: Anomalies in multidimensional contexts. WIT Transa. Inform. Commun. Technol. **42**, 173 (2009). http://www.witpress.com/elibrary/wit-transactions-on-information-and-communication-technologies/42/19978
4. Han, J., Chen, Y., Dong, G., Pei, J., Wah, B.W., Wang, J., Cai, Y.D.: Stream cube: an architecture for multi-dimensional analysis of data streams. Distrib. Parallel Databases **18**(2), 173–197 (2005). http://link.springer.com/article/10.1007/s10619-005-3296-1
5. Li, X., Han, J.: Mining approximate top-k subspace anomalies in multi-dimensional time-series data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 447–458. VLDB Endowment (2007)
6. Lin, S., Brown, D.E.: Outlier-based Data Association: Combining OLAP and Data Mining. Department of Systems and Information Engineering University of Virginia, Charlottesville, VA 22904 (2002). http://web.sys.virginia.edu/files/tech_papers/2002/sie-020011.pdf
7. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: DARPA Information Survivability Conference and Exposition, DISCEX 2000, Proceedings, vol. 2, pp. 12–26 (2000)
8. Müller, E., Assent, I., Iglesias, P., Mülle, Y., Böhm, K.: Outlier ranking via subspace analysis in multiple views of the data. In: 2012 IEEE 12th International Conference on Data Mining, pp. 529–538. IEEE (2012). http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6413873
9. Palpanas, T., Koudas, N., Mendelzon, A.: Using datacube aggregates for approximate querying and deviation detection. IEEE Trans. Knowl. Data Eng. **17**(11), 1465–1477 (2005)
10. Rettig, L., Khayati, M., Cudré-Mauroux, P., Piórkowski, M.: Online anomaly detection over Big Data streams. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 1113–1122 (2015)
11. Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-driven exploration of OLAP data cubes. In: Schek, H.-J., Alonso, G., Saltor, F., Ramos, I. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 168–182. Springer, Heidelberg (1998). doi:10.1007/BFb0100984
12. Sithirasenan, E., Muthukkumarasamy, V.: Substantiating anomalies in wireless networks using group outlier scores. J. Softw. **6**(4), 678–689 (2011)
13. Thatte, G., Mitra, U., Heidemann, J.: Parametric methods for anomaly detection in aggregate traffic. IEEE/ACM Trans. Networking **19**(2), 512–525 (2011)
14. Xin, D., Han, J., Li, X., Wah, B.W.: Star-cubing: Computing iceberg cubes by top-down and bottom-up integration. In: Proceedings of the 29th International Conference on Very Large Data Bases, vol. 29, pp. 476–487. VLDB Endowment (2003)